



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 Antoine MAILLARD pour m'avoir accompagner les premières semaines et pour être resté entièrement à mon écoute. Merci à Ziad AKL pour son accueil, son suivi régulier et son écoute tout au long de ma présence à SAP. Merci à Christophe DOLIMONT pour ses conseils avisés et son suivi sur les tests auto que j'ai implémenté. Merci à Fabien COAT pour son soutien et son suivi sur le plugin

Sommaire

Remerciements	ii
Sommaire	vi
Introduction	1
1 Présentation de l'entreprise	3
2 Software tester	7
3 Migration Jira/Java Correction WorkBench	25
4 Plugin Jenkins	31
Conclusion	43
A Les niveaux de test	45
B Résultats quotidiens des tests	47
C Tous les tests implémentés	51
D Les bases pour comprendre Jenkins	57
E Maven	59
F Résultats quotidiens des tests	61
G Maven	63
H Fiche d'information de SAP	65
Table des figures	70

Introduction

Ce contrat de professionnalisation à été une expérience exceptionnellement riche tant par ce que j'ai appris à faire ou à être que par les nombreuses connaissances que j'ai pu accumuler.

Mon contrat était divisé en deux périodes, la 1^{ère} s'étend du 15 juillet au 30 septembre 2014 et la seconde du 15 février au 31 septembre. Ces longues périodes de travail ont permis une immersion complète de le monde de l'entreprise.

La 1^{ème} 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

SAP, qui signifie Systems Application and Products in data processing, a été formé en 1972 par 5 anciens employés d'IBM et ont inventé l'ERP¹ ou PGI en Français². Son siège est à Walldorf, en Allemagne. SAP conçoit et vend des logiciels de gestion et de reporting à destination des entreprises et des professionnels. SAP est le premier éditeur de logiciels en Europe (devant Dassault Systèmes, Sage et Software AG)³ et le quatrième mondial (derrière Microsoft, IBM et Oracle)⁴. Ses principaux concurrents sont les autres éditeurs d'ERP et de logiciels de gestion.

company leadership <http://www.sap.com/corporate-en/about/our-company/our-boards/executive-board.html>

SAP mène une stratégie d'acquisition⁵ des leaders des technologies de l'information, la croissance organique étant le moteur de cette stratégie, tout en continuant à investir dans le développement de ses propres produits et dans les technologies innovatrices. Cette stratégie permet à SAP de mieux soutenir l'innovation en la dirigeant vers les partenaires possédant les domaines d'expertise requis. SAP perdure dans ce sens pour acquérir les technologies stratégiques cibles pour couvrir un marché toujours plus grands et pour satisfaire toujours plus les besoins de ses clients.

La structure actuelle est le fruit de la fusion entre SAP France et Business Objects

1. Entreprise Ressource Planning
2. Logiciel de gestion intégré
3. Source : <http://www.journaldunet.com/solutions/dsi/truffle-100-europe-2014.shtml>
4. Source : <http://www.journaldunet.com/solutions/ssii/classement-mondial-editeur-2010/editeur-logiciel-classement-mondial.shtml>
5. Source : <http://www.sap.com/corporate-en/about/investors/newsandreports/acquisitions/index.html>

dont la date légale est le 1er janvier 2010, l'annonce ayant été faite le 7 octobre 2007. Précédemment, Business Objects avait absorbé une autre entité, CARTESIS. Les rachats continus Sybase en 2010, SuccessFactors en 2011, Ariba et Syclo en 2012, hybris en 2013, FieldGlass et pour finir Concur en 2014.

D'autres petites entreprises (60/100 salariés) sont en cours d'intégration : Sybase (au 01/04/2013), Success Factors et Ariba ; comme de grandes entreprises : Crystal object et Concur⁶.

Le chiffre d'affaire de l'année fiscale 2014 est de 17,56 milliards €.

SAP fourni plus de 293 000 clients venant de 190 pays et regroupe plus de 74 400 employés de plus de 130 pays⁷. Pour plus d'information à ce sujet consulter l'annexe H page 65⁸.

1.1.1 Les produits SAP

1.1.2 SAP en France

En France, SAP a implémenté des bureaux commerciaux à Lyon, Nantes, Toulouse, Strasbourg, Bordeaux, Aix, Sophia Antipolis et Caen. Leur centre de recherche (SAP Labs) était anciennement à Levallois-Perret (Paris). Paris habitait trois sites SAP différents, ils étaient présents à la défense (principalement pour les ressources humaines), à Courbevoie (principalement du développement) et à Levallois (Développement de solution et SAP Labs). Ces trois centres ont été rassemblés en un seul et la fusion s'est terminée mi-avril 2015 quand les locaux de Levallois-Perret front de Seine ont été vidés. Aujourd'hui c'est la tour SAP, à Levallois-Perret, de 22 étages qui accueille les employés des trois sites précédents pour un total de plus de 1500 personnes.

!!! photo!!!!

1.2 Contexte

Après la fusion avec Business Object, SAP a sorti sa version 4.0 de WebI. Celle-ci, se voulant innovante et complète, a grandement déçu le client car le logiciel comportait beaucoup de bugs et certaines fonctionnalités n'étaient pas implémentées. Ceci s'explique d'une part par le fait que beaucoup trop de fonctionnalités ont été planifiées. Et d'autre part parce que les différentes équipes travaillaient en tunnel, que le trop grand nombre de dépendances rendait très difficile l'intégration complète du produit et que la date de sortie du logiciel ne pouvait pas être repoussée.

6. Source : <http://pro.clubic.com/it-business/progiciel/sap/actualite-728255-sap-concur.html>

7. Source : <http://www.sap.com/corporate-en/about/our-company/index.html>

8. http://www.sap.com/bin/sapcom/en_us/downloadasset.2015-07-jul-22-01.SAP-Corporate-Fact-Sheet-en-2015-07-22-pdf.html

La réaction de SAP France pour ne pas réitérer cette erreur a donné lieu aux 8 initiatives qualité, leurs permettant ainsi d'assurer la qualité de leurs produits.

1.2.1 Les 8 initiatives qualité

1.3 Présentation de l'équipe

L'équipe dans laquelle j'ai été intégré est la suivante!!!! diagramme

1.3.1 Son histoire

puis au moment de la 4.1 sp6 Initialement Raphaël est le chef de toutes les équipes de test (QPA-QPE, fonctionnels, performance, automatique, ...). Mais il est devenu le manager du groupe WebI et, parallèlement, les équipes de tests devenaient de plus en plus nombreuses. Ne pouvant plus manager les équipes de test, celle-ci ont été divisées en deux catégories principales et sont à la charge de plusieurs managers. D'un côté il y a les tests fonctionnels et les tests effectués « à la main », de l'autre il y a les « autres ». Autrement dit les tests automatiques, les web services, les performances et les projets en avance sur le reste.

L'équipe dans laquelle j'ai été intégré est « P&I BIT BI Suite Paris ST Transversal team »⁹. Elle a été créée en juin 2014, un peu plus d'un mois avant mon arrivée.

1.3.2 Sa mission

1.3.3 Les membres

Diagramme présentant l'équipe

1.4 Ingénieur logiciel à SAP

Chaque ingénieur se voit attribuer un poste de travail (PC, bureau, ...) et un identifiant, le mien étant I310911, lui permettant de se connecter sur le portail interne à SAP ainsi que sur les différents outils.

- perforce - jenkins astec

!! ici les outils utilisés par les employés SAP!!! portal : jam, wiki Les tickets : IT ticket, hrTicket, Ticket CSS

9. Product and Innovation BIT Business Intelligence Suite Paris Software Tester Transversal Team

Chapitre 2

Software tester

2.1 Généralités sur le test

Pour répondre à la question « Qu'est-ce qu'un test ? » je dirais que c'est un code vérifiant le bon fonctionnement d'un code. Il existe de nombreux niveaux de test : unitaire, fonctionnel, d'intégration, de performance, UI, Les tests sont très importants dans un projet logiciel, ils permettent de détecter des bugs et de s'assurer du bon fonctionnement du logiciel. La mise en place de ces tests est facilité par un Framework, il en existe de nombreux mais de manière générale un Framework est un ensemble de bibliothèques et de normes (de modélisation, d'architecture, ...) ¹.

2.1.1 Description des différents types de test

Le test unitaire

Un test unitaire vérifie le bon fonctionnement d'une petite portion de code. En règle générale il ne s'agit que de tester que la valeur retournée par une méthode est la bonne. Mais il s'agit aussi de valider le fonctionnement de la méthode aux limites et de vérifier la cohérence de ses résultats. Ces tests sont censés être les plus nombreux pour couvrir le maximum de cas possible, on appelle cela le test coverage (ou couverture de test).

Le test fonctionnel

Le test fonctionnel est axé sur l'interaction des différentes méthodes. Ces tests permettent de vérifier le bon fonctionnement des spécifications du produits.

1. Source : <https://boulaich.wordpress.com/2009/07/08/framework-generalite/>

Le test d'intégration

Le test d'intégration est au test fonctionnel ce que le test fonctionnel est au test unitaire. Le test d'intégration s'assure du bon fonctionnement du logiciel en interaction avec d'autres logiciels.

Le test de performance

Le test de performance s'assure qu'une action est effectuée dans le temps imparti. Les actions testées sont au minimum celles qui composent les workflows nominaux, tant séparément qu'ensemble.

Le test de stabilité

Le test de stabilité permet de tester qu'un comportement est systématique, autrement dit qu'une même action implique toujours le même résultat.

Le test de scalabilité

Le test de scalabilité permet de s'assurer que le logiciel peut évoluer de manière à pouvoir offrir les mêmes performances dans un contexte d'utilisation qui a évolué. Par exemple dans le cas où la charge d'utilisation double, est-il possible de modifier l'architecture du logiciel afin que les utilisateurs profitent des mêmes performances ?

Le test de validation de plateforme

Ce type de test est consacré aux logiciels destinés à être utilisés sur plusieurs plateformes. Est-il possible de l'installer sur plusieurs OS ? Les comportements sont-ils les mêmes ?

2.2 Le test à SAP

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 est responsable de la bonne mise en œuvre de l'initiative qualité « 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 produits, possédant chacun plusieurs branches, et pour 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 47 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 9 pour le process complet)

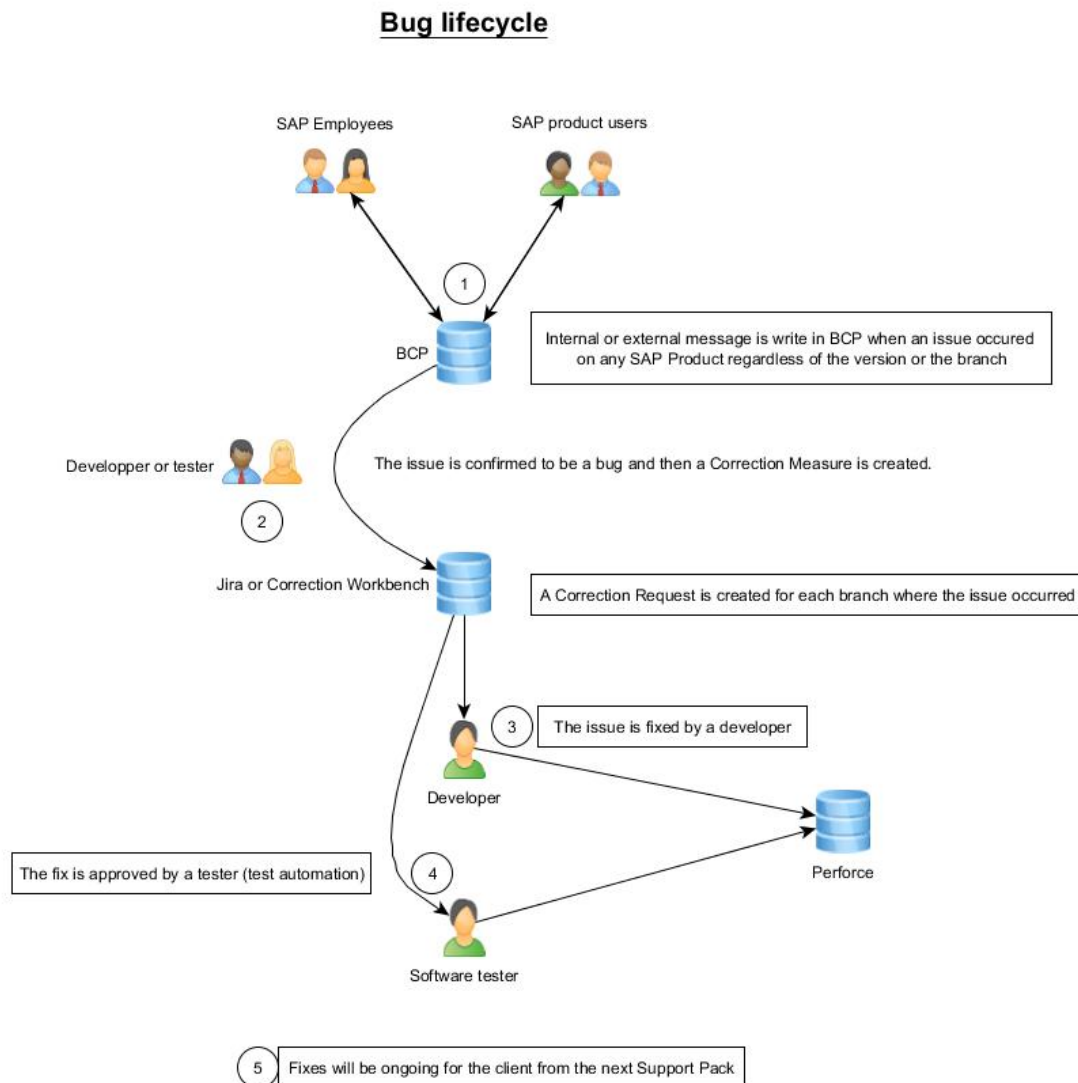


FIGURE 2.1 – Le process de test

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

Testeur fût ma première mission lors de mon arrivé à SAP et je le suis resté sur toute la première période de mon contrat, c'est-à-dire 2 mois et demis. Avant de commencer à implémenter des tests j'ai été accueilli par mes collègues et on m'a fourni le matériel nécessaire au bon déroulement de mes missions.

Mes 1^{er} 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 du workspace perforce) à l'aide de Christophe DOLIMONT
- Au terme de la 1^{ère} semaine, j'ai pu commencer à étudier le framework de test en me basant sur les tests déjà existants

C'est au cours de la deuxième semaine que j'ai pu implémenter mes 1^{er} tests.

2.5 Synthèse des 1^{ères} mission

Tout au long de cette mission, ou plutôt ces missions (un test achevé laisse toujours la place à un autre), j'ai travaillé à implémenter des codes java permettant de tester automatiquement le comportement de WebI au niveau SDK.

L'objectif principal de cette mission était d'être, à terme, capable d'implémenter seul et dans les plus brefs délais un test automatique. La grande difficulté de début de cette mission a été de comprendre, d'une part, la logique de test du framework de test, et d'autre part les différentes choses que celui-ci me permettait de faire. J'ai distingué les quelques points suivants qui sont pour moi l'essentiel qu'un testeur doit maîtriser :

- 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 wid, une queriespec ou rien du tout
- Les tests ne se basant pas sur les mêmes versions du produits, les JARs ne sont jamais les mêmes
- Où trouver les fichiers nécessaires

- Respect absolu des conventions de nommage

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 document WebI alors que l'autre permet d'appliquer un certains nombre des modifications sans pour autant le comparer systématique à la fin.

La différence en terme de consommation de temps est très importante, puisque 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 finalement commit l'implémentation. Voyons ceci plus en détail.

Tests statiques

Globalement, le test statique :

- ne fait que comparer un document par rapport à une référence
- ne demande que très peu de connaissances technique, que ce soit en Java ou sur WebI

Principe du test statique

Lors de l'exécution d'un test statique, un fichier est généré à partir d'un wid. Le format du fichier ainsi généré (txt_2, 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.

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
  import org.junit.Test;

11
  import tests.exported.annotations.B0Test;
13 import tests.exported.annotations.B0TestPlan;
  import extensions.toolbox.WIStaticTestPlanDefinition;

15
```

2. type personnalisé interne à SAP

```

@BOTestPlan(Type = TestPlanType.FEATURE_VERIFICATION, Suite = Suite.AURORA,
    Feat = Features_REBEAN.WI)
17 public class CM_{cmNumber} extends WISstaticTestPlanDefinition{
    private static String _scriptId = "CM_{cmNumber}_{short
19 text}";
    public CM_{cmNumber} () {
21     super(_scriptId);
    }
23
    @Test
25     @BOTest(Objective = "Test CM_{cmNumber}_{short
text}' functionality", Severity = Severity.CRITICAL, Author = "", Mode = Mode.
    PROD, Type = Type.FUNCTIONAL)
27     public void CM_{cmNumber}_{short text}" () {
        logNumericResults(launchTC(getTestcase("CM_{cmNumber}_{short text}")));
29     }
    }

```

Une fois ceci fait il est nécessaire de générer le fichier de référence. Cela se fait en exécutant le test une 1^{ère} fois en mettant l'option `savingtxt`³ du `script.xml` à `true` (voir figure 2.2 page 12)

```

<SCRIPT ID="CM_835743_2014_CalcIssue">
    <NAME>webi_customers_issues</NAME>
    <TYPE>WI Static</TYPE>
    <SAVINGCHART>true</SAVINGCHART>
    <SAVINGTXT_>false</SAVINGTXT_>
    <SAVINGTXT>false</SAVINGTXT>
    <SAVINGDOC>false</SAVINGDOC>
</SCRIPT>

```

FIGURE 2.2 – Contenu du `script.xml` pour générer une référence

À cette étape l'exécution du test échouera puisqu'il n'y a pas de référence. Le fichier de référence ainsi généré doit être copier depuis le répertoire de résultat puis collé dans le dossier où sont stockés les fichiers de ressources (voir la figure 2.9 page 20 pour avoir une illustration des différents fichiers et dossiers dans l'architecture du framework de test).

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

3. Pour générer une ressource `.txt`, `savingdoc` pour générer un `doc`, etc.

évidemment possible mais pas systématique, il suffit souvent de tester la valeur ou l'existence d'un objet.

Dans point de vue général 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 appel l'une des méthodes « derrière » ce bouton, reproduisant ainsi le comportement au niveau SDK 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 :

Les fichiers de log

Ces fichiers contiennent des informations telles que les exceptions levées ou tout autre informations que va générer le logger

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_{cmNumb

text}}\n
 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 2.3 page 15, et le docu-
 ment généré est enregistré dans My Documents/My Favorites/Personnals Docu-
 ments, voir l'illustration 2.4 page 16

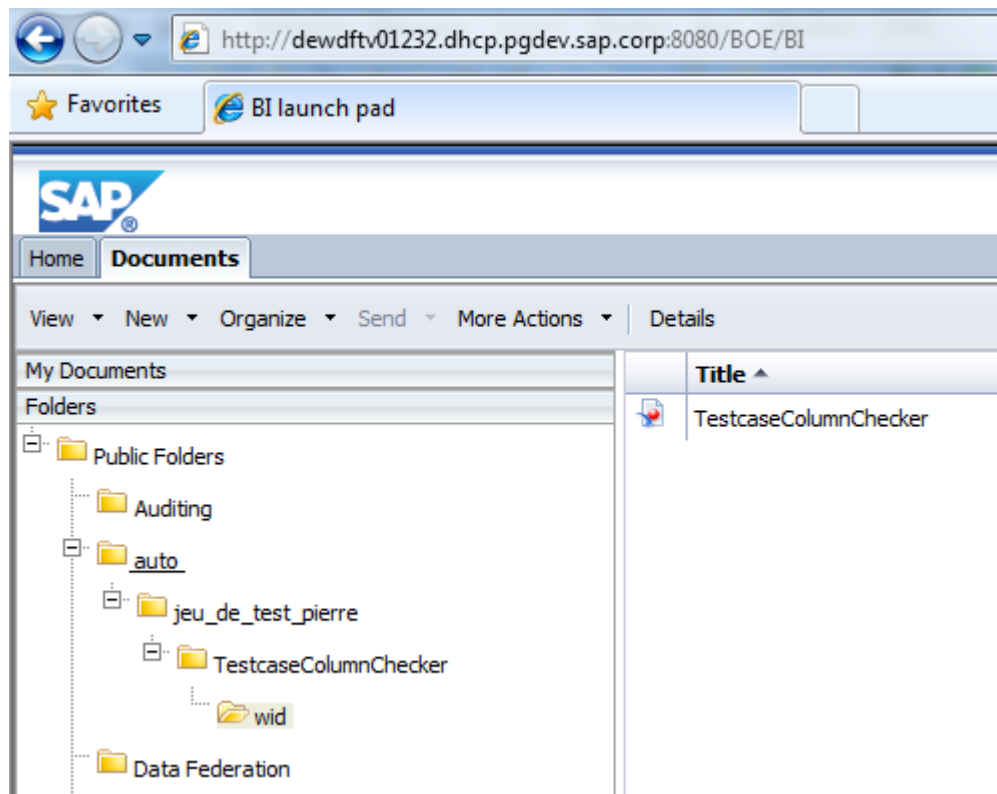


FIGURE 2.3 – Capture d'écran de WebI de l'arborescence du document de référence

Les fichiers ressources

Les .wid sont enregistrés en local dans le répertoire
 ... \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
 text}}\n
 voir une illustration de cette arborescence figure 2.5 page 16

Les logs du serveur tomcat

Disponible dans le dossier d'installation du serveur ces logs sont trop verbeux
 pour pouvoir être utilisés systématiquement. Mais ils restent une précieuse source
 d'informations et sont utiles dans certains cas.

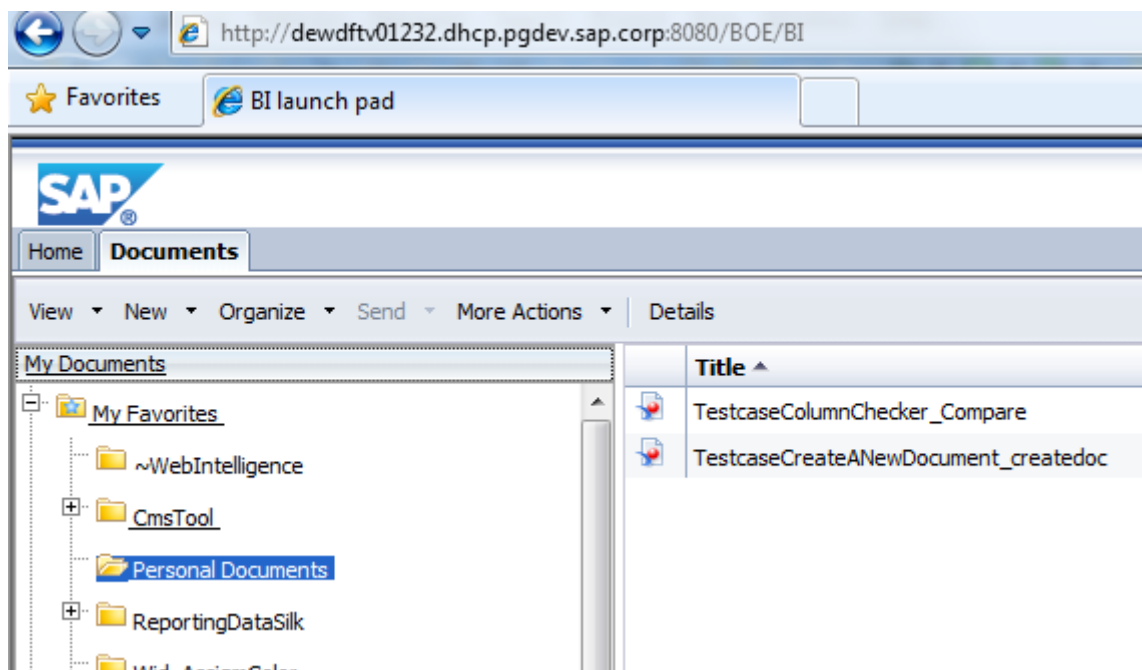


FIGURE 2.4 – Capture d'écran de WebI de l'arborescence du document généré

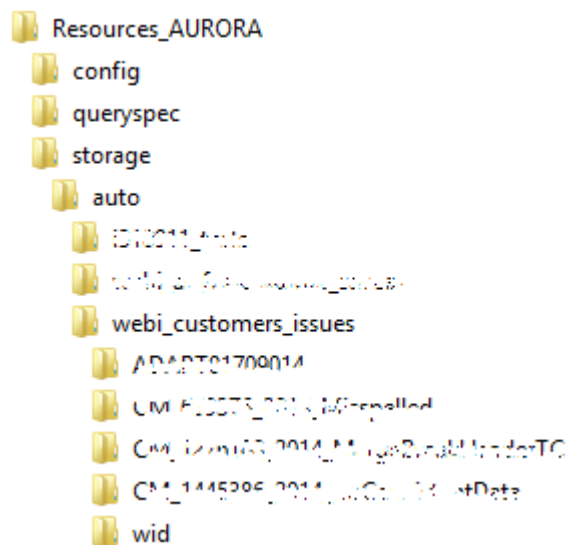


FIGURE 2.5 – Capture d'écran de WebI de l'arborescence des références locales

2.5.2 De l'étude à l'intégration

D'abord, les demandes de tests automatique ne sont pas toujours réalisables au niveau SDK. Il convient donc de vérifier, en 1^{er} lieu la faisabilité du test. Une fois que l'on sais

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 (voir figure 2.6 page 17), les informations nous concernant sont :

- L'identifiant du defect (utilisé par les conventions de nommage)
- La description détaillée du bug
- Le workflow provoquant le bug
- La/Les branche(s) sur laquelle/lesquelles il a été décelé

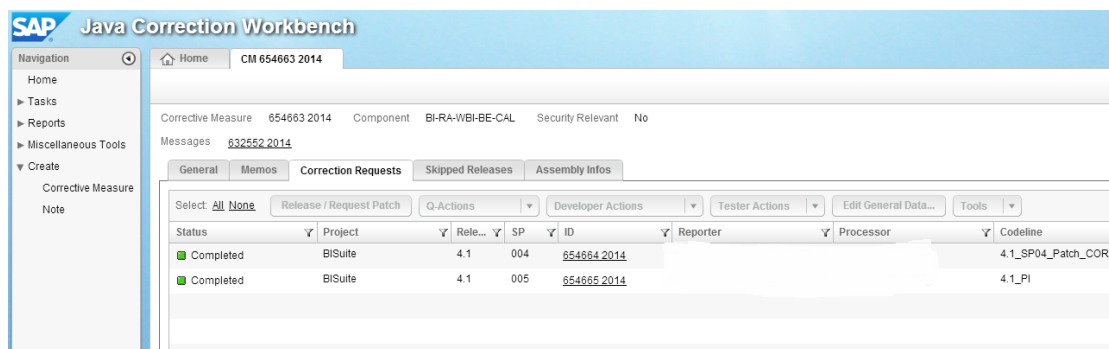


FIGURE 2.6 – Écran de JCWB propre à une CM

Ceci fait, il est intéressant d'aller étudier le code qui a été modifié pour corriger le bug. Cela permet de se faire une idée des objets provoquant le bug ainsi que les méthodes corrigées dont il faut maintenant tester le bon fonctionnement. Pour cela il suffit d'aller dans Perforce et de rechercher la modification effectuée. Un clic droit sur la changelist en question offre la possibilité d'utiliser la fonction « diff against previous revision » du SCM pour obtenir la liste exhaustive des fichiers modifiés ainsi qu'un vis-à-vis entre les versions pré-correctif et post-correctif (voir figure 2.7 page 18)

Reproduire le problème Une fois que le bug est 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).

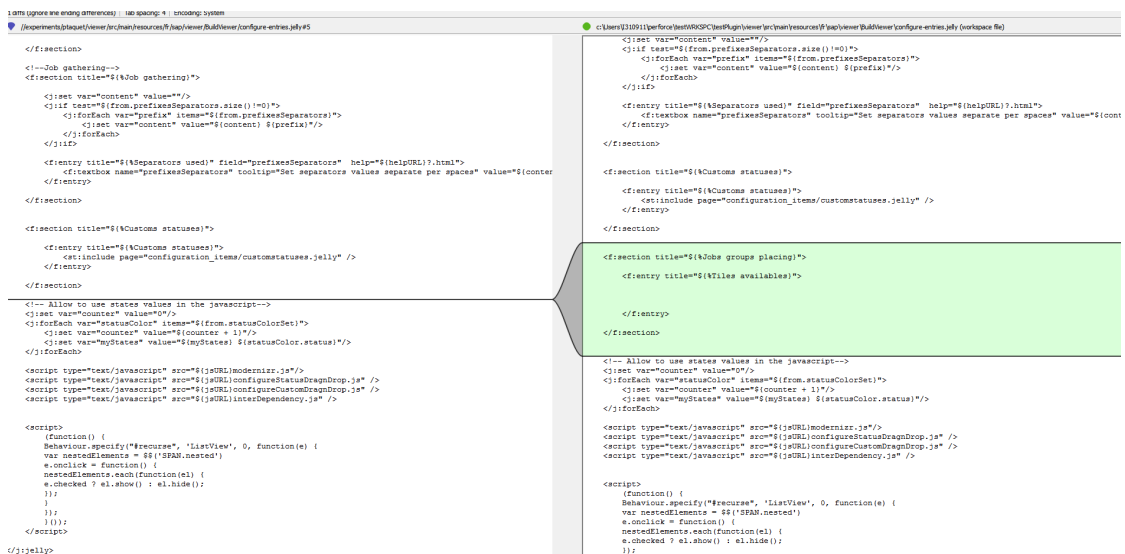


FIGURE 2.7 – Écran de comparaison des 2 versions d'un même fichier (avant et après correctif)

Si l'implémentation du test est possible, il faut choisir la meilleure manière de 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 a bien été nommé, test case et script.xml. Et, en fonction du besoin, s'assurer de la génération sur le serveur des .wid. Comme schématisé figure 2.8 page 18, chaque test automatique est intégré à un test plan, lui-même étant intégrés à une test suite.



FIGURE 2.8 – 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 `testplans.rebean_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. Voir l'implémentation d'exemple partie 2.5.1 page 13).
2. **Test case** Uniquement dans le cas du test dynamique. Créer le test case dans le package correspondant au client (dans `testcases.aurora_customers.{clientID}`). Attention à respecter le pattern de nommage « `CM_{CM_id}_shortText` », voir l'implémentation d'exemple partie 2.5.1 page 13.
 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 (`querspec`, `.wid`, etc.) et les ajouter au dossier portant le nom de la CM (respect de la convention de nommage), garantissant l'unicité de ce dossier.
 6. **parameters.xml** Renseigner l'url du CMS ciblé et mettre à jour les extracted JARs permettant la compilation du test vide
 7. Vérifier que tous les éléments nécessaires sont dans la change list de perforce, puis « push » les modification.

D'une manière générale, la structure à connaître pour implémenter correctement la coquille vide est illustrée figure 2.9 page 20. L'utilisation des ces fichiers ou dossiers est systématique pour certains et quasi-systématique pour d'autres, les connaître est essentiel. Et, comme illustré figure 2.10 page 21, il est important de savoir à partir de quoi l'on peut construire un test et ce qu'il est possible de générer par celui-ci.

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.
Cette ressource est générée à l'exécution du test à partir du document en cours, il faut donc d'abord le copier/coller depuis le répertoire où est généré cette ressource vers celui où la référence sera récupérée.
Une question se pose alors : cette référence que j'ai ainsi généré est-elle garantie sans erreur ?

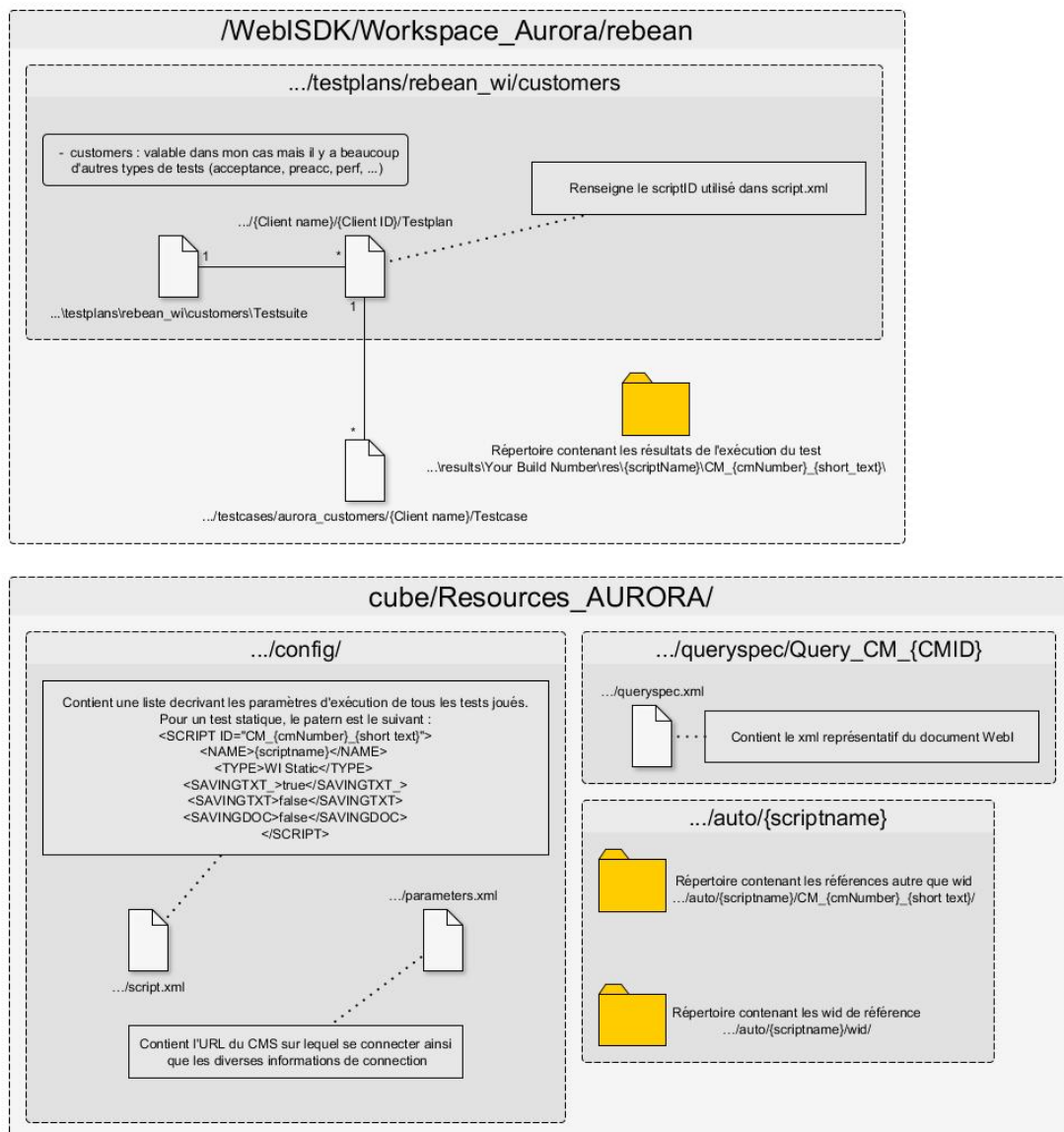


FIGURE 2.9 – Diagramme représentant les différents éléments qui compose la coquille vide d'un test dynamique

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 2.11 page 21). Ensuite, dans le système de fichiers du serveur (par exemple :

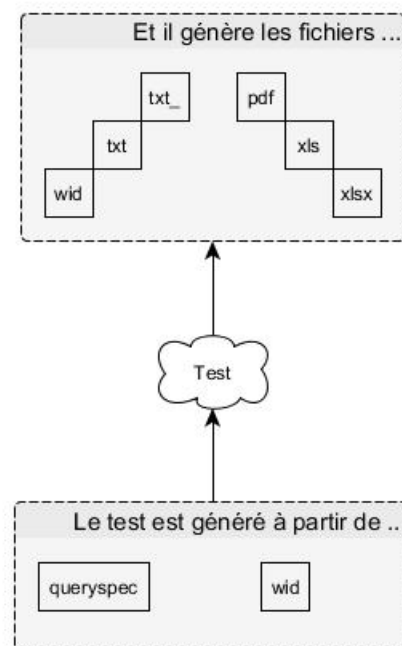


FIGURE 2.10 – Les fichiers utilisés ou générés par le test

« \\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.

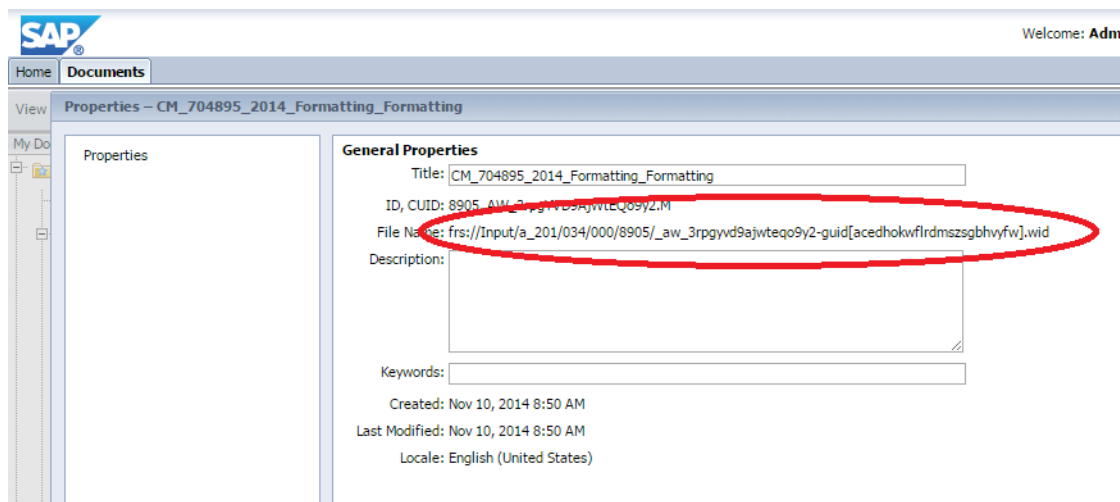


FIGURE 2.11 – Capture de l'écran de propriété d'un document WebI

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 la 1^{ère} période

Tout au long des mois de juillet, août et septembre j'ai implémenté de nombreux tests⁴ pour des bugs divers, que ce soit une faute de frappe ou une fonctionnalité ne fonctionnant plus du tout. 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.

J'ai aussi pu parfaire mes connaissances en Java ainsi que ma méthodologie lorsque je dois me former à une nouvelle technologie.

Au début de cette mission je n'avais jamais vraiment implémenter de tests à l'exception de tests unitaires. Aujourd'hui, je suis ravi de comprendre la problématique du test et quels types de tests il existe. Pourquoi et comment les implémenter et être capable de faire des tests qui sont soient facilement maintenables.

2.6.1 Connaissance du framework

La principale perte de temps était dû à une méconnaissance du Framework de test. Celui-ci possédant un grand nombre de méthodes aux intérêts divers et variés, d'emblée il est très difficile de pouvoir déterminer laquelle utiliser. D'autant qu'au moment où le besoin de telle ou telle méthode se faisait sentir je n'avais pas connaissance de son existence. En conséquence, je me suis vu trop souvent implémenter des méthodes déjà existantes, me faisant perdre un temps précieux.

L'expérience accumulée durant cette période m'a permis de ne plus perdre de temps à ré-inventer la roue et de me concentrer sur le code ad hoc nécessaire à un test précis et fiable.

2.6.2 Méthodologie

L'autre grande perte de temps venait surtout de ma méconnaissance du produit testé. Certaines fonctionnalités étant très complexes il m'arrivait de ne pas identifier exactement le problème pour finalement perdre du temps à implémenter du code inefficent.

Le meilleur exemple que je puisse citer s'est produit quand j'ai commencé à tester des bugs intégrés à des workflows complexes. Un certain nombre de manipulations et/ou de configurations était alors nécessaires pour que le bug survienne. Et c'est dans ces conditions que j'ai implémenté la quasi totalité du workflow au niveau SDK pour finalement arriver sur la zone à tester. Le test fonctionnait bien, il échouait sur les versions buggées et passait sur les versions fixées. Le problème ne vient pas du test que j'ai implémenté, car celui-ci faisait ce qu'il fallait, mais de la manière ! J'ai implémenté ce code en approximativement 3 jours. Alors que si j'avais utilisé le CMS pour générer un document à une étape seulement du bug, et que si j'avais implémenté uniquement l'appel à tester, alors j'aurais pu implémenter ce test en quelques heures seulement !

4. cf. annexe C page 51 pour la liste complète des tests implémentés

2.6.3 Travail en équipe

Le framework de test est entretenu quotidiennement par des dizaines de testeurs implémentant des tests et modifiant le framework lui-même. Le résultat de l'exécution des suites de tests est visible par testeurs, développeurs et par d'autres. J'étais donc intégré à une grande équipe, leurs travaux ayant des répercussion sur les miens, et les miens sur les leurs. Il faut donc faire attention au code que l'on « push » sur Perforce ! L'anecdote me venant à l'esprit s'est passée un vendredi soir, alors que je venai de terminer un test et que celui-ci se comportait bien. J'ai livré mon test sur perforce. L'ennui était que j'utilisais un package dans lequel je mettais des méthodes de test pour ne pas polluer mon test « final ». Je n'ai malheureusement pas pensé à revoir l'import de mes sources, mon code ne compilait pas et a donc fait crashé toute la suite de test. Le mail automatique envoyé à informer tout le monde de mon erreur de manipulation (illustrée annexe F page F) !

J'ai eu aussi la chance de travailler dans une équipe maîtrisant très bien le framework et le produit. J'ai appris énormément pendant les entretiens que j'ai eu avec ces différentes personnes.

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 projets 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

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 est constitué de deux choses distinctes, d'une part son code source et d'autre part les tests joués pour garantir¹ 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 pour être représentatif du problème. Dès lors que quelqu'un s'en aperçoit, il inscrit un defect dans Jira² ou dans JCWB.

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 jobs³. Un groupe de jobs est un carré coloré dont la couleur représente l'état des jobs qui le compose, voir la capture d'écran figure 3.1 page 26.

Les statuts pris en compte sont les statuts Jenkins⁴ ainsi qu'un statut supplémentaire issue du plugin Claim. Ce plugin permet à un développeur de « claimer » 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 laissé.

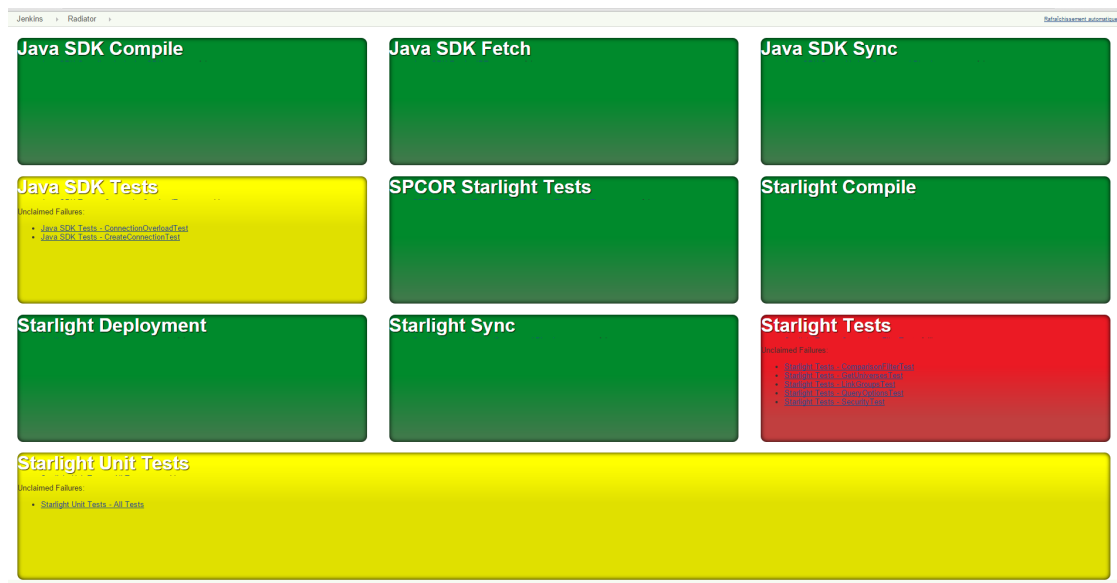


FIGURE 3.1 – Plugin Radiator utilisé actuellement

En résumé Radiator :

- permet de rassembler les jobs en un seul groupe de jobs
- où chacun des jobs a un statuts⁵
- que le groupe de projet arbore la couleur représentative du statut jugé le plus important
- permet 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).

3. Typiquement, un job correspond à un projet logiciel

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

5. statuts Jenkins

La figure 3.2 page 27 illustre bien les sources d'informations du plugin. Il puise ses résultats depuis Jenkins et le plugin claim, les résultats de Jenkins puisant ses résultats en partie des résultats de tests JUnit.

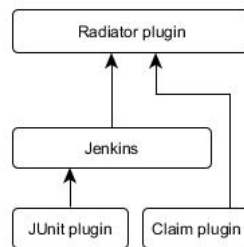


FIGURE 3.2 – Plugin Radiator utilisé actuellement

Situation actuelle

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 sans defect (le processus de correction n'est pas initialisé) apparaissent de la même manière que les tests échoués avec defect (le processus de correction est en cours). 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) en vert.

De cette manière, seuls les nouveaux échecs apparaissent en rouge. En grande partie des régressions.

Problématique

Actuellement seuls les defects rentrés sur Jira sont pris en compte, la mécanique en place ne permet pas de récupérer ces mêmes informations de CWB.

Puisque de plus en plus de defects seront enregistrés sur CWB, et dans un souci d'homogénéisation du process actuel, il faut mettre à jour le process actuel et permettre aux statuts d'être ajustés quelque soit l'origine du defect (Jira ou CWB).

3.3 Process actuel

Afin de faire passer le status d'un test d'échoué à succès, c'est le fonctionnement de JUnit lui-même qui est surchargé. La mécanique en jeu est composée :

D'une annotation Appliquée au test plan, elle permet au ST de renseigner l'url du defect Jira.

```
1 @Defect(url="https://sapjira.wdf.sap.corp/browse/BITBIWEBISL2-1542",
    message="expected:<WARNING> but was:<OK>")
```

Cette annotation est prise en compte, ou pas, en fonction de la valeur d'un paramètre -D (voir annexe ??? sur les paramètres -D).

Une classe surchargeant JUnit JiraIssueWatcher.java, elle prend en compte l'existence de l'annotation et, en fonction du defect, fait passer un test échoué en test passé.

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

Dans l'état actuel des choses le process utilisé est illustré figure 3.3 page 28

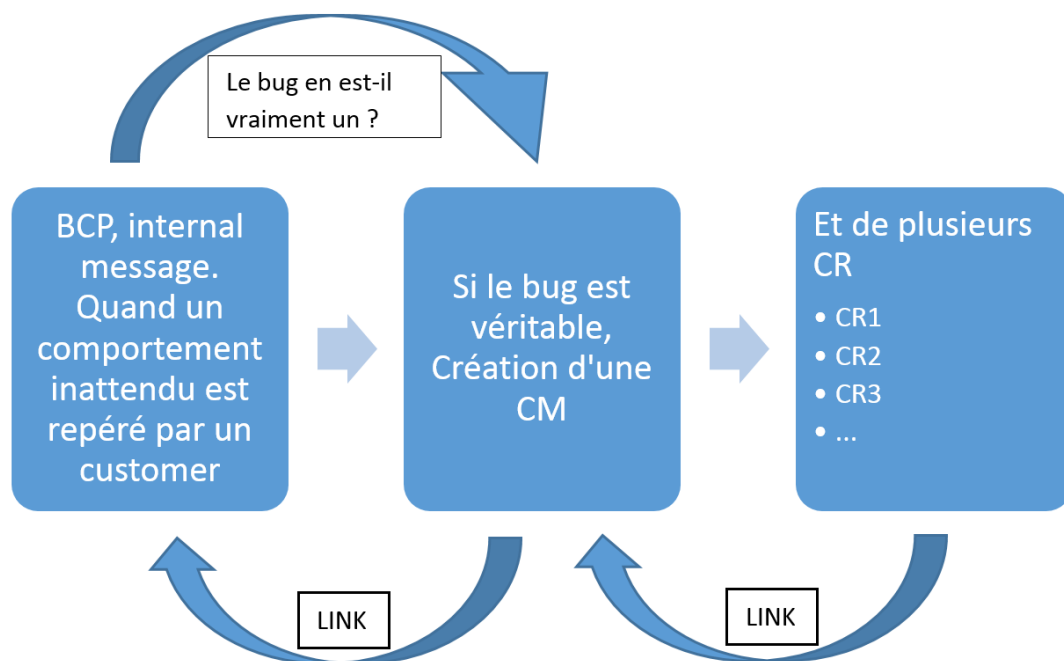


FIGURE 3.3 – Process de gestion de bug utilisé

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 échanges 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 32 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

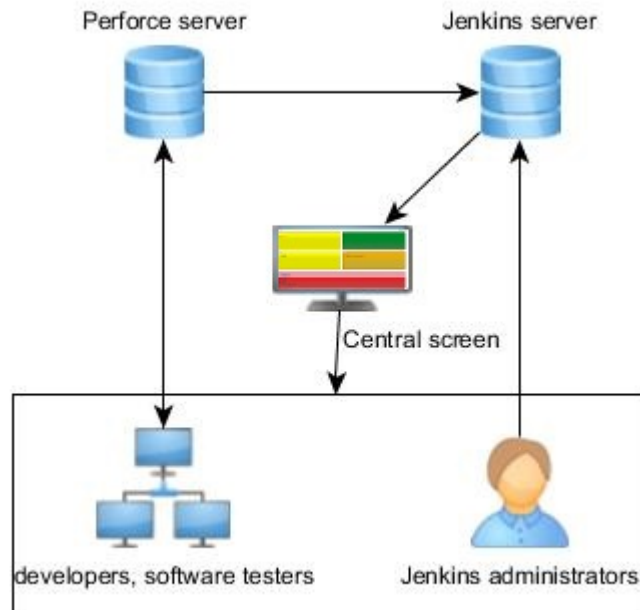


FIGURE 4.1 – L’environnement d’utilisation du plugin

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 33). Par contre l’affichage n’est pas représentatif de la réalité, une partie de l’information est perdue puisque’un test échoué avec defect apparaît en vert (succès).

La solution proposée

Dans un 1^{er}, 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 liste¹ des jobs avec defect (exemple page 33).

```

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"

```

1. Au format xml

```

    status="failed">
21   <defect expectedMessage="Failing test" link="https://css.wdf.sap.corp/sap/
    bc/bsp/spn/jcwb?crPointer=012006153200001159182015" status="In Process"
    type="cwb"/>
    </testcase>
23 </testcases>

```

4.1.1 Organisation du travail

meeting hebdomadaire avec manager pour faire le point sur l'évolution du projet et fixer les nouveau objectifs.

Code review hebdomadaire avec Fabien

4.2 Étude préalable

Les 1^{er} impératifs ont été de maitriser 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 forciori sur la manière de l'étendre.

Il existe un archétype maven (cf annexe E page 59) 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 clair. Mais la documentation officiel² 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.

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

4.3 Le 1^{er} plugin

Je me suis d'abord rendu sur la page du tutoriel officiel de Jenkins³ afin d'obtenir un plugin fonctionnel avec lequel je pouvais faire mes expériences. Ensuite j'ai suivi un autre tutoriel⁴ 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 maven⁵ pour générer le squelette. Maven a besoin de configurations supplémentaires dans son `settings.xml`⁶ 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
```

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

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`

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

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 36. Certaines choses importantes sont à noter, le pom.xml⁷ 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.

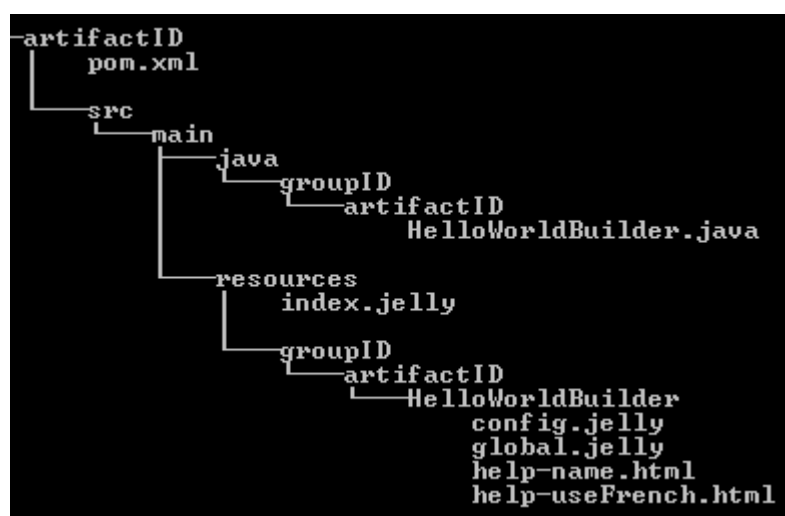


FIGURE 4.2 – Architecture d'un plugin Jenkins généré par maven

!!!!!! 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 ⁸, 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 ⁹ 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 38.

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.

8. à l'emplacement du POM.xml

9. port 8080



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

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.

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 projet associés à la vue (figure 4.4 page 39).

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

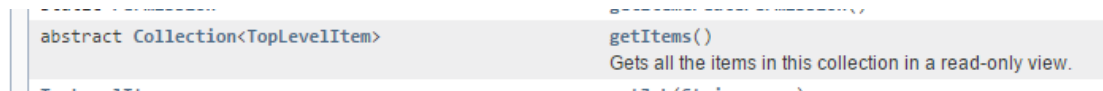


FIGURE 4.4 – Extrait de la javadoc de la classe ListView

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 fichiers. Une fois que Jenkins a localisé 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 trouvent 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 39. Il convient maintenant de copier ce fichier et de le coller dans le répertoire *ressources* de notre projet.

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 40) et deuxièmement on supprime l'affichage du side-panel, après chargement de la page (cf figure 4.7 page 40).

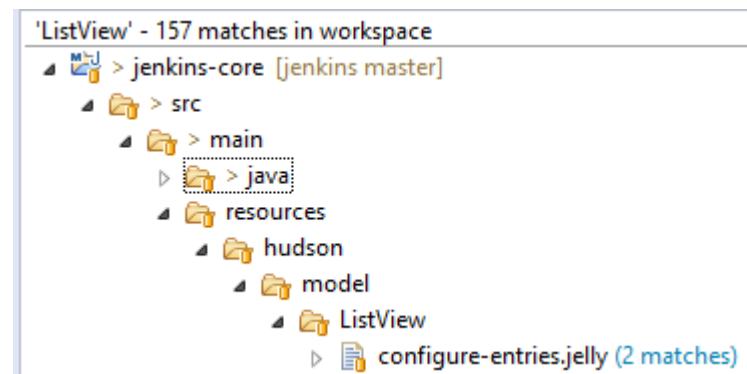


FIGURE 4.5 – Chemin d'accès au fichier de configuration par défaut

```
<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

4.5.1 Caractéristiques du logiciel

changer couleur background et police regroupe par prefixe priorise les status va chercher info dans archive

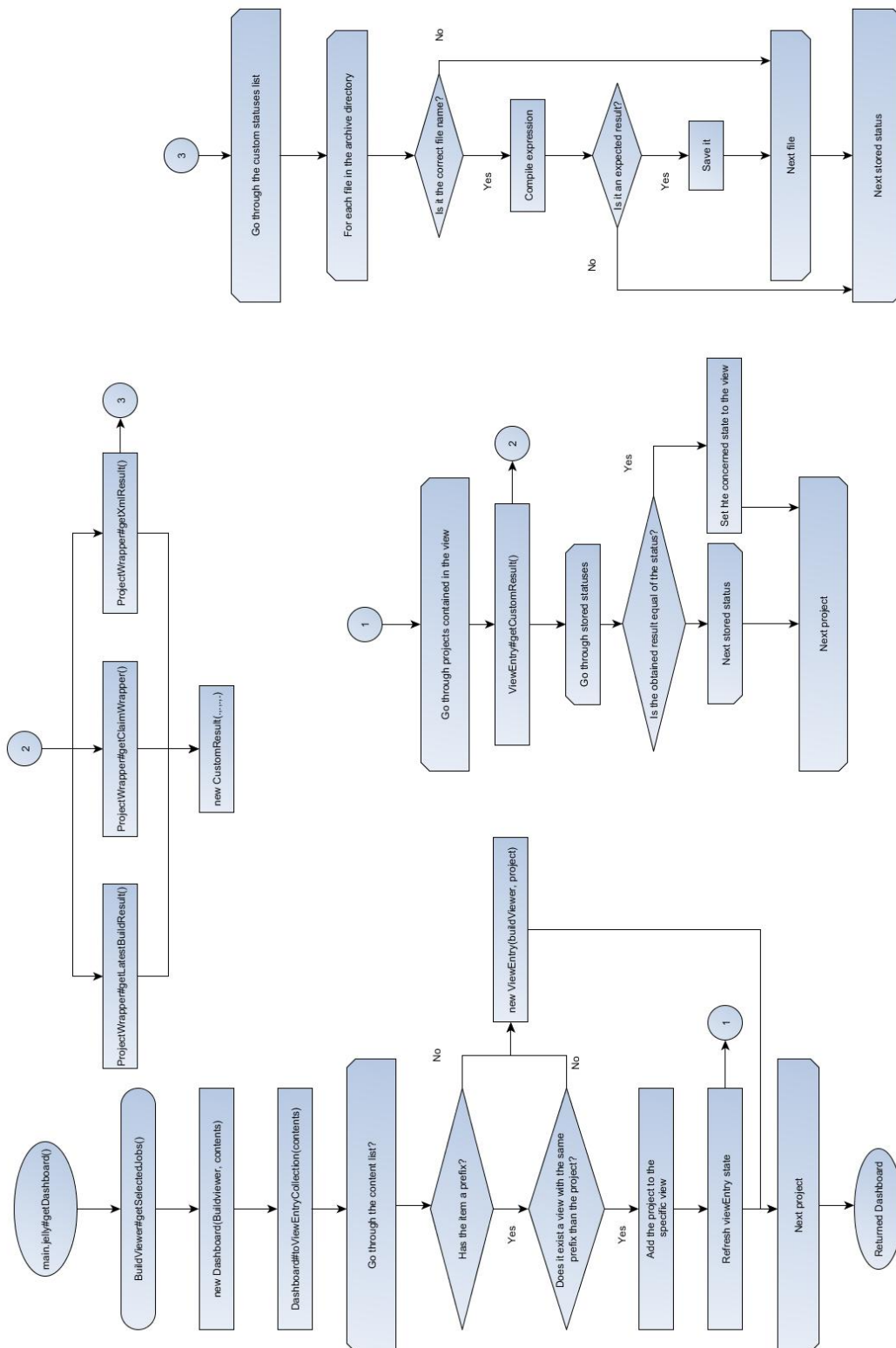


FIGURE 4.8 – Algorithme de création du Dashboard

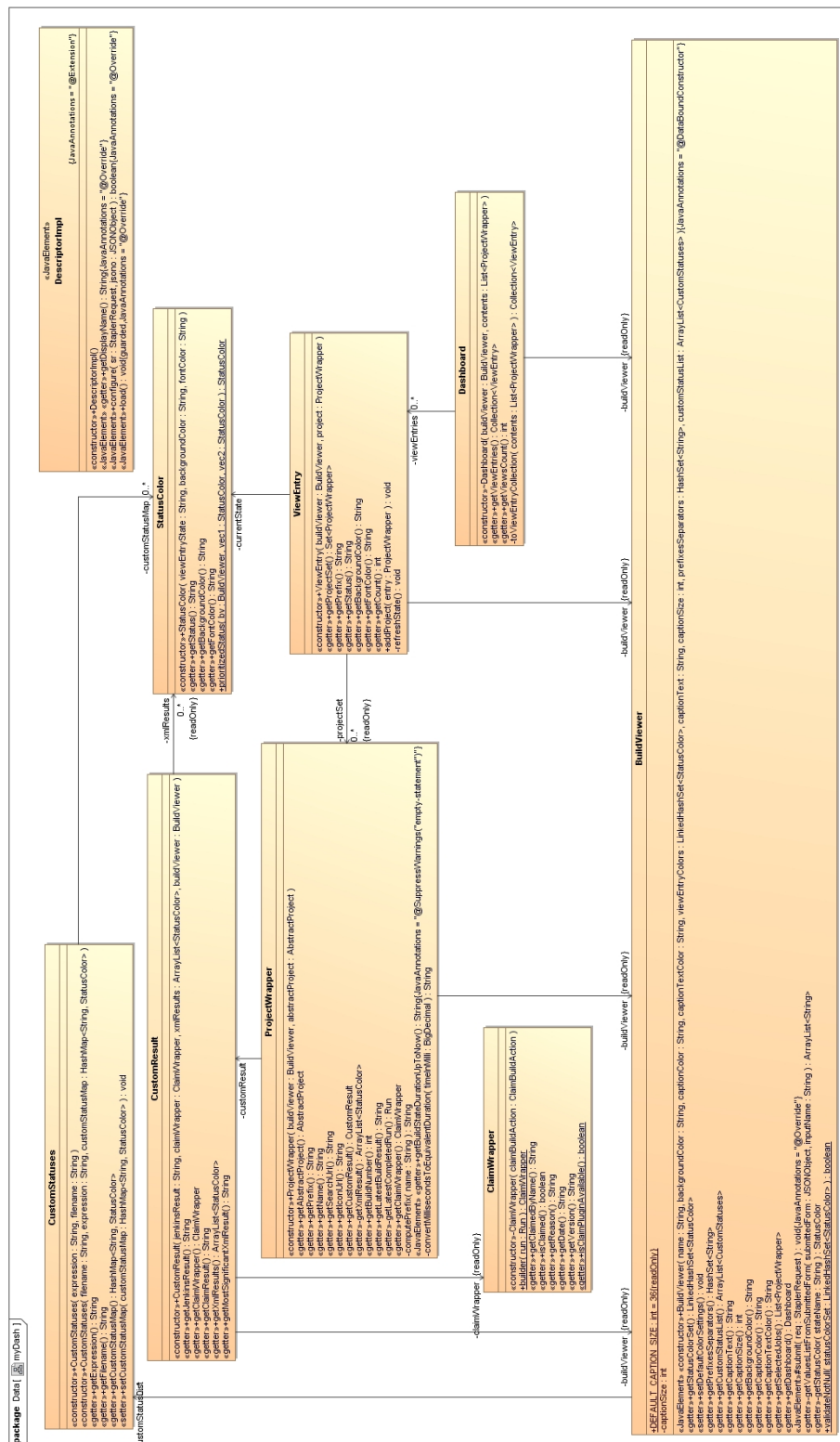


FIGURE 4.9 – Diagramme de classe du plugin réalisé

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

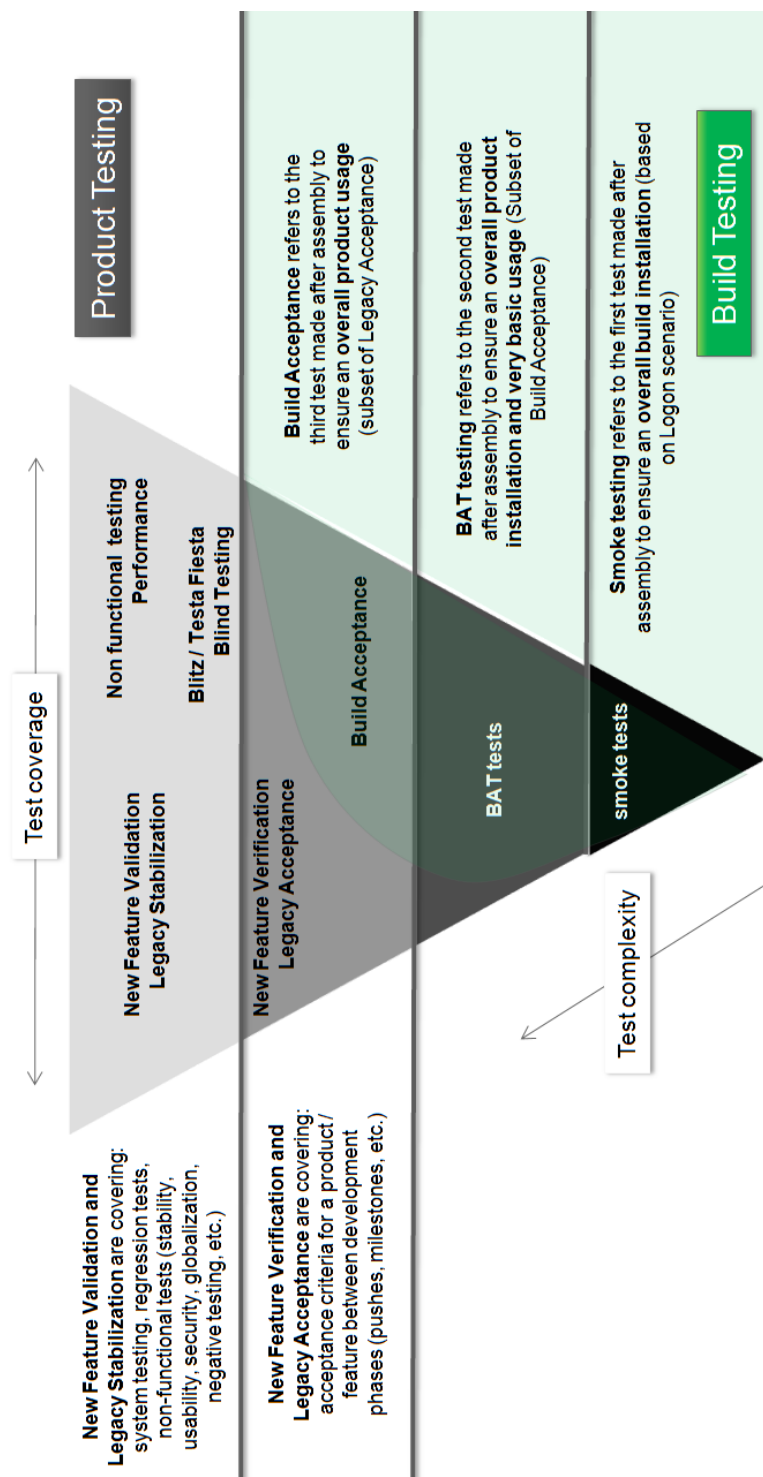


FIGURE A.1 – Les différents niveaux de test

Résultats quotidiens 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
rebean_wi.customers	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
TestSuite_Customers	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_0000035772_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.Aquitais_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	666569 5	add a testplan		08\04082014 071708.xps
04/08/2014	666579 6	Add the testplan in the testsuite. Opening and outline tests.	- BIT BIWEBISL2_14 64_OpenADoc - BIT BIW EBIS L3_1 605_ outli ne	08\04082014 084249.xps
05/08/2014	666950 8	Add wid files for webi_perf_wrkf lw_amandine		08\05082014 031026.xps
11/08/2014	669672 3	Add testcase for break header	CM_1276108_2014_M ergeBreakHeaderTC	08\11082014 084911.xps
13/08/2014	670608 2	Add testcase for input controls upgrade	CM_1386568_2014_In putControlsUpgrade	08\13082014 093545.xps
13/08/2014	670616 5	Modify the test file		08\13082014 104552.xps
14/08/2014	670969 2	Modify test case about upgrade		08\14082014 020039.xps
14/08/2014	671013 0	Add test case architecture for ADAPT0170901 4_spacing	ADAPT01709014	08\14082014 040011.xps

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

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

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

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

Annexe F

Résultats quotidiens des tests

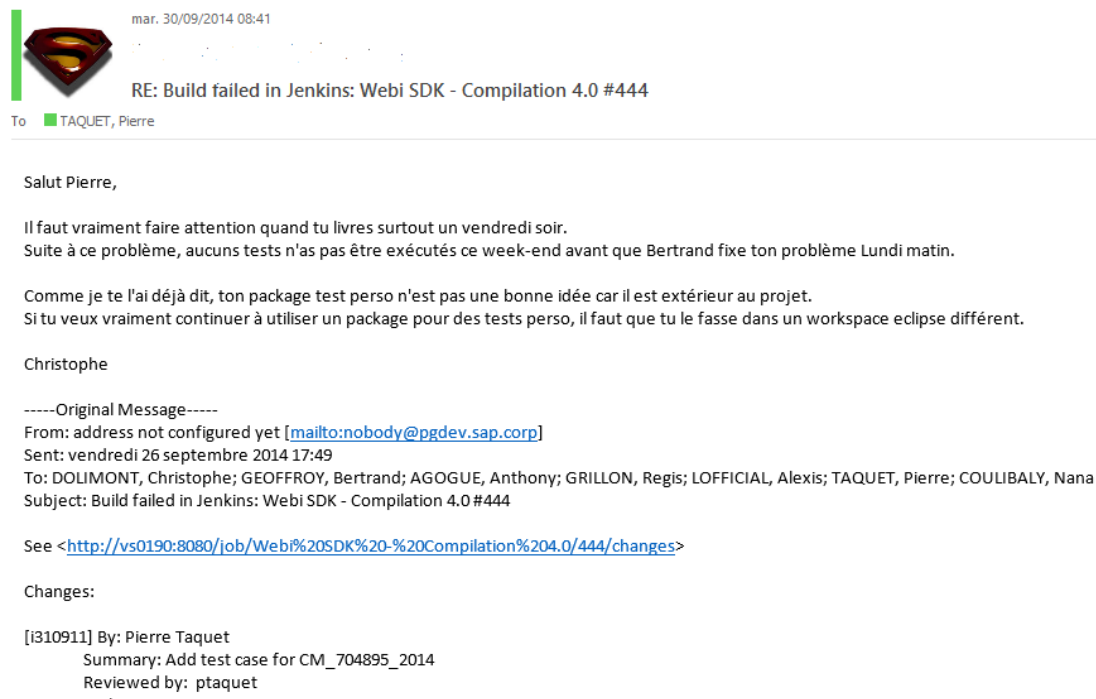


FIGURE F.1 – Capture d'écran du résultat de ma mauvaise manipulation

Annexe G

Maven

explication des paramttreds -d

Fiche d'information de SAP



SAP Global Corporate Affairs

July 22, 2015

SAP: Run Simple - The World's Largest Provider of Enterprise Application Software

As market leader in enterprise application software, SAP (NYSE: SAP) helps companies of all sizes and industries innovate through simplification. From back office to boardroom, warehouse to storefront, on premise to cloud, desktop to mobile device – SAP empowers people and organizations to work together more efficiently and use business insight more effectively to stay ahead of the competition. SAP applications and services enable customers to operate profitably, adapt continuously, and grow sustainably.

CUSTOMERS

- SAP serves > 293,000 customers in 190 countries
- > 80% of SAP customers are SMEs
- SAP customers include:
 - 87% of the Forbes Global 2000 companies
 - 98% of the 100 most valued brands
 - 100% of the Dow Jones top scoring sustainability companies
- Our customers produce
 - 78% of the world's food
 - 82% of the world's medical devices
- 74% of the world's transaction revenue touches an SAP system¹

HOW WE RUN SIMPLE

- Our vision is to help the world run better and improve people's lives
- Our mission is to help our customers run at their best
- To fulfill our mission, we help our customers master complexity with simple and easy to use solutions focused on innovation in the Cloud and on SAP HANA, e.g. S/4 HANA, Simple Finance
- Our strategy is to become THE cloud company powered by SAP HANA with focus on
 - Public Cloud: Standard and Suite solutions
 - Business Network: Ariba, Concur, Fieldglass
 - Private Cloud on HANA Enterprise Cloud

FINANCIALS

Revenue (IFRS) per industries
in € millions, as of December 31, 2014



- Revenue – FY 2014 (non-IFRS@const. curr.)
 - Cloud subscr. & support € 1.1 bn (+45%)
 - SW&Support² € 13.8 bn (+5%)
 - Total € 17.6 bn (+5%)
- Revenue – Q2 2015 (non-IFRS@const. curr.)
 - Cloud subscr. & support € 555 mn (+92%)
 - SW&Support² € 3.51 bn (+3%)
 - Business Network € 333 mn (+145%)
 - Total € 4.4 bn (+8%)
- Outlook 2015 (non-IFRS @ const. curr.)
 - Cloud subscr.&support rev.: €1.95 - €2.05 bn
 - Cloud &SW³ rev up 8%-10% (2014: €14.33 bn)
 - Operating profit in a range of €5.6 - €5.9 bn
- Ambition 2017 (non-IFRS@cc)
 - Cloud subscr.&support rev.: €3.5 - €3.6 bn
 - Total revenue: €21 - €22 bn
 - Operating profit in a range of €6.3 - €7.0 bn
- Ambition 2020 (non-IFRS@cc)
 - Cloud subscr.&support rev.: €7.5 - €8.0 bn
 - Total revenue: €26 - €28 bn
 - Operating profit in a range of €8 - €9 bn

1 Source: McKinsey/SAP analysis update 4/2013

2 SW&Support: Software and Support

3 Cloud&SW: Cloud Subscriptions and Software

MARKET POSITION

ENTERPRISE APPLICATION SOFTWARE

- SAP is market leader in
 - applications
 - analytics
 - mobility solutions
- Fastest growing database vendor
- Broadest portfolio of modular and suite solutions available on premise, in the cloud and hybrid: customers have full choice of consumption model

TOP CLOUD VENDOR

- Fastest growing company at scale in the cloud
- Cloud user base: ~82 mn subscribers
- Largest cloud portfolio: >30 solutions for all lines-of-business (LoB) as well as Business Suite
- Market leader in Human Capital Mgmt. solutions

LEADING MOBILITY VENDOR

- Market leader for mobile business applications: >130 mn mobile users, >500 mobile apps
- SAP mobile solutions reach 99.9% of mobile subscribers via text messaging
- 1.8 bn text messages per day processed and delivered by SAP Mobile Platform

INNOVATION

- 14 Development centers (SAP Labs) worldwide
- 100 Development locations worldwide
- 13 Co-Innovation and Living Labs worldwide
- 21 Research locations worldwide
- Innovation Center in Potsdam, Germany
- Partner network with >13,000 SAP partner companies around the world
- Sapphire Ventures: Invested in >150 IT startups globally since 1996
 - US\$1.4 bn capital under management
 - Operates independently from SAP
 - Gives SAP early visibility and access to markets, trends & innovation

EMPLOYEES AND BASIC FACTS

Employees per functional area
as of December 31, 2014



- Headquarters: Walldorf, Germany
- Founded: April 1, 1972
- Listing: Frankfurt, New York
- 74,497 employees worldwide (06/30/2015)
 - EMEA: 33,467
 - Americas: 21,574
 - APJ: 19,456
- >120 nationalities worldwide
- nearly 80 nationalities at headquarters

USEFUL LINKS

[Executives](#) – [Supervisory Board](#) – [Products](#) – [Industries and Solutions](#) – [Events](#) – [Financials](#) – [Photos and Films](#) – [SAP Profile](#)

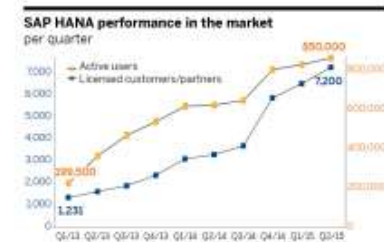
SAP'S END-TO-END SOLUTIONS

Simple user experience designed with a mobile first mindset

1 – APPLICATIONS

- Packaged solutions for 25 industries and 11 lines-of-business. On premise, cloud, hybrid
- SAP Business Suite optimizes all business-critical processes
- Market leader in products for business analysis and a technology leader for real-time analysis: Business intelligence, Predictive Analytics etc.
- S/4HANA: the most significant business software innovation since SAP R/3
 - 10x smaller data footprint than conv. systems
 - 7x higher throughput
 - 1800x faster analytics and reporting
 - All data: text, social data, geo data, graph processing
 - Instantaneous simulations, predictions, recommendations
 - Delivery on premise, cloud and hybrid
 - Connected to Internet of Things

2 – SAP HANA PLATFORM



- SAP HANA is the market-leading platform for real-time computing:
 - Open platform
 - Basis for all major SAP solutions, will become underlying technology for all SAP applications
 - SAP HANA Cloud Platform enables customers to extend existing Cloud applications or quickly develop entirely new ones
 - HANA Enterprise Cloud: access to the full potential of HANA via managed Cloud
- Customers:
 - >7,200 HANA customers
 - >850,000 active users
 - Suite on HANA: 1,850 customers
 - S/4HANA (launch Feb. 2015): 900 customers
 - >2,100 startups developing on HANA platform

3 – BUSINESS NETWORK

- SAP's Business Network companies provide the leading solutions in the areas of
 - goods and services (Ariba)
 - travel and expense (Concur) and
 - contingent labor (Fieldglass)
- > 35 million users in the Networks worldwide
- The SAP Business Networks connect industry ecosystems of more than 1.9 million businesses, their partners, 3rd party developers and system integrators
- Trade volume > US\$0.8 trillion p.a.



Index

Business Intelligence, 10

CMS, 15, 17, 19, 20, 22

Correction Measure, 17, 19

Defect, 25, 27

Framework, 10

Graphical User Interface, 14

JAR, 10

Java Correction WorkBench, 9, 17, 25, 27

Jenkins, 25

Jira, 9, 25, 27

Perforce, 8, 25

querspec, 10, 14, 18–20

Rich Client, 10

SDK, 14

Software Tester, 8, 9, 19

Source Code Management, 10, 17

Test coverage, 8

wid, 10, 11, 14, 15, 18–22

Table des figures

2.1	Le process de test	9
2.2	Contenu du script.xml pour générer une référence	12
2.3	Capture d'écran de WebI de l'arborescence du document de référence . . .	15
2.4	Capture d'écran de WebI de l'arborescence du document généré	16
2.5	Capture d'écran de WebI de l'arborescence des références locales	16
2.6	Écran de JCWB propre à une CM	17
2.7	Écran de comparaison des 2 versions d'un même fichier (avant et après correctif)	18
2.8	Diagramme UML des suites de tests	18
2.9	Diagramme représentant les différents éléments qui compose la coquille vide d'un test dynamique	20
2.10	Les fichiers utilisés ou générés par le test	21
2.11	Capture de l'écran de propriété d'un document WebI	21
3.1	Plugin Radiator utilisé actuellement	26
3.2	Plugin Radiator utilisé actuellement	27
3.3	Process de gestion de bug utilisé	28
4.1	L'environnement d'utilisation du plugin	32
4.2	Architecture d'un plugin Jenkins généré par maven	36
4.3	Jenkins à l'exécution de la commande maven qui l'encapsule	38
4.4	Extrait de la javadoc de la classe ListView	39
4.5	Chemin d'accès au fichier de configuration par défaut	40
4.6	Initialisation des css avant chargement	40
4.7	Initialisation des css après chargement	40
4.8	Algorithme de création du Dashboard	41
4.9	Diagramme de classe du plugin réalisé	42
A.1	Les différents niveaux de test	45
F.1	Capture d'écran du résultat de ma mauvaise manipulation	61

Table des matières

Remerciements	ii
Sommaire	vi
Introduction	1
1 Présentation de l'entreprise	3
1.1 SAP dans le monde	3
1.1.1 Les produits SAP	4
1.1.2 SAP en France	4
1.2 Contexte	4
1.2.1 Les 8 initiatives qualité	5
1.3 Présentation de l'équipe	5
1.3.1 Son histoire	5
1.3.2 Sa mission	5
1.3.3 Les membres	5
1.4 Ingénieur logiciel à SAP	5
2 Software tester	7
2.1 Généralités sur le test	7
2.1.1 Description des différents types de test	7
2.2 Le test à SAP	8
2.3 Présentation du produit testé : WebI	10
2.4 Préparation aux tests	10
2.5 Synthèse des 1 ^{ères} mission	10
2.5.1 Tests statiques ou dynamiques	11
2.5.2 De l'étude à l'intégration	16
2.6 Bilan de la 1 ^{ère} période	23
2.6.1 Connaissance du framework	23
2.6.2 Méthodologie	23

2.6.3	Travail en équipe	24
3	Migration Jira/Java Correction WorkBench	25
3.1	Qu'est-ce que Jira et Java Correction WorkBench	25
3.2	Présentation du contexte	25
3.2.1	Plugin de reporting	26
3.3	Process actuel	27
3.4	Travail effectué	29
3.4.1	Fonctionnement et utilisation de prod-pass-access	29
3.5	Résultats obtenus	29
4	Plugin Jenkins	31
4.1	Contexte initial	31
4.1.1	Organisation du travail	34
4.2	Étude préalable	34
4.3	Le 1 ^{er} plugin	35
4.3.1	Le squelette du plugin	35
4.4	Implémentation de la base du nouveau plugin	38
4.4.1	Import du code dans l'IDE	38
4.4.2	Implémentation de la base	39
4.5	Travail réalisé	40
4.5.1	Caractéristiques du logiciel	40
	Conclusion	43
	A Les niveaux de test	45
	B Résultats quotidiens des tests	47
	C Tous les tests implémentés	51
	D Les bases pour comprendre Jenkins	57
	E Maven	59
	F Résultats quotidiens des tests	61
	G Maven	63
	H Fiche d'information de SAP	65
	Table des figures	70
	Table des matières	72

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