



Mise en place d'un système d'information

Élodie Bardaji

► To cite this version:

| Élodie Bardaji. Mise en place d'un système d'information. Autre [cs.OH]. 2014. dumas-01221539

HAL Id: dumas-01221539

<https://dumas.ccsd.cnrs.fr/dumas-01221539>

Submitted on 28 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL ASSOCIE DE TOULOUSE

MEMOIRE

présenté en vue d'obtenir

le DIPLOME D'INGENIEUR CNAM

SPECIALITE : **Informatique**

OPTION : **Systèmes d'information (SI)**

par **Elodie Bardaji**

MISE EN PLACE D'UN SYSTEME D'INFORMATION

Lieu de la mission : Solutions Isonéo – Montréal, Québec, Canada

Responsable : Erwan DESCHAMPS

Tuteur : Xavier CREGUT

Soutenu le 14 janvier 2014

JURY

PRESIDENT : Madame Anne Wei – Professeur des Universités - CNAM Paris

MEMBRES : Monsieur Thierry Millan – Maître de conférence – CNAM Midi-Pyrénées

Monsieur Xavier Crégut – Maître de conférence – INPT-ENSEEIH

Monsieur Erwan Deschamps – Chef d'entreprise – Solutions Isonéo

Monsieur Julien Siega – Responsable technique – Solutions Isonéo

Remerciements

Tout d'abord je souhaite remercier le président et les membres du jury pour l'intérêt qu'ils ont porté à mon travail ainsi que Xavier Crégut, mon tuteur, pour l'accompagnement tout au long de celui-ci.

Je témoigne également toute ma reconnaissance à tous mes collègues de travail, amis et famille qui m'ont apporté leur soutien, leur aide, leur conseil mais aussi tout simplement leur présence durant l'année d'organisation qui a précédé mon stage au Canada. Un mot particulier cependant pour leurs réponses et retours de documents rapides à Frédérique Monferran, responsable administrative au CNAM, Sarah Pécantet en charge de mon dossier au Fongecif et Krystèle Blocard, assistance de gestion chez Astek Sud-Ouest ; et pour leur compréhension et la liberté d'action qu'ils m'ont fourni professionnellement parlant : Bastien Vialade, chef de l'agence Astek Sud-Ouest, Jean-Yves Dicembre, responsable du pôle de gestion de configuration et Karine Le Roux, consultant manager du projet pour lequel je travaillais en France.

La réalisation de ce stage a également été possible grâce à la société d'accueil Solutions Isonéo. Ma reconnaissance va à Pierre Duverneuil, PDG de la société, Raphael Giraud, son Directeur Général, Florence Serié, assistance de gestion et Erwan Deschamps, responsable local de la société à Montréal qui ont mis tout en œuvre pour obtenir mon entrée sur le territoire canadien au plus vite. De bonnes conditions de travail ont ensuite été possibles auprès d'Evelyne Fatela-Nabais, Erwan Deschamps, Julien Siega et Jonathan Lasalle que je remercie pour leurs disponibilités, leurs idées et leur sympathie.

Je souhaite également exprimer mes plus grands remerciements à Johan qui a été très disponible durant ce stage pour m'apporter son expertise d'administrateur système Linux. Ses qualités pédagogiques m'ont permis d'acquérir une base de connaissances solide dans ce domaine inconnu.

J'ai également été touchée par la gentillesse des personnes qui ont accepté de relire, d'apporter des corrections ou de me conseiller sur mon travail : Michel, Nicolas et Jonathan.

Un grand merci à ma famille pour les encouragements et la reconnaissance de mes efforts ; tout particulièrement à mon copain également pour sa patience, sa compréhension, son expertise en architecture matérielle et sa capacité à me faire prendre du recul sur mon travail durant ces 5 années de cours de soir.

Abréviations

CNAM	Conservatoire National des Arts et Métiers
SI	Système d'information
VIE	Volontariat International en Entreprise
GED	Gestion Electronique des Documents
SA ² GE	Système Aéronautique d'Avant-Garde pour l'Environnement
IDE	Integrated Development Environment
SQL	Structured Query Language
HSQldb	Hypersonic Structured Query Language DataBase
POM	Project Object Model
LDAP	Lightweight Directory Access Protocol

Glossaire

Artefacts Maven : librairies téléchargées de façon automatique par l'outil de compilation Maven. Elles sont vérifiées à chaque compilation du projet et téléchargées sur chacun des postes des programmeurs excepté lorsqu'elles sont centralisées par la société dans un outil de gestion des artefacts comme Nexus.

Cas de test : chemin fonctionnel représentant un des comportements possibles d'une méthode Java évaluée par la mise en œuvre d'un jeu d'essai.

Couverture de tests : terme utilisé pour évaluer la proportion d'une application validée par la mise en œuvre de tests informatiques. L'objectif est généralement d'obtenir une couverture complète dans l'optique de vérifier le bon fonctionnement de la totalité du code. Des outils comme Sonar gèrent la couverture de tests et fournit des statistiques en pourcentage sur le nombre de tests par rapport au nombre de méthodes Java présentes sur le projet.

Daily Scrum : réunion journalière matinale durant laquelle tous les membres de l'équipe présentent les tâches qu'ils ont réalisées la veille et expliquent leurs objectifs pour la journée à venir.

Gestion de configuration : métier informatique associé aux personnes en charge de la mise en place du système d'information et de sa configuration. Généralement ce métier consiste également à mener la politique de changement et à former les utilisateurs.

Gestionnaire de demandes : outil permettant de gérer des demandes sous forme d'enregistrements dans une application. Une demande possède un titre, une description et est assignée à une personne qui est en charge de sa résolution. Les demandes évoluent ensuite suivant un cycle de vie (workflow).

Gestionnaire de sources ou de versions : outil permettant de gérer du code sources Java, .Net, PHP, etc. dans une équipe de plusieurs personnes sans risquer de supprimer le travail réalisé par un collègue. Ce logiciel enregistre toutes les modifications déposées sur le projet et permet ainsi de retourner sur une version antérieure du logiciel développé si besoin.

IDE Eclipse : logiciel de programmation englobant un environnement de développement fournissant ainsi un environnement de production de logiciels libres qui peut réutiliser d'autres projets Eclipse.

Intégration continue : procédé permettant de compiler les sources de façon automatisée afin de vérifier que les sources déposées sur le projet n'appliquent pas de régression sur l'application programmée.

Méthode Java : sous-programme permettant de mettre en œuvre une partie d'une fonctionnalité de l'application.

Plug-in ou plugin : ensemble de sources additionnelles qui a pour objectif d'ajouter de nouvelles fonctionnalités au logiciel auquel il est associé.

Procédure de test : ensemble des cas de tests.

Règles de codage : règles définissant les bonnes pratiques à respecter quant à la mise en forme et au contenu du code source. Elles sont aussi très souvent désignées par « règles checkstyle » provenant de leur dénomination anglaise « checkstyle rules ».

Scrum : méthode Agile utilisée par Solutions Isonéo qui consiste à utiliser des cycles de développement itératif afin d'éviter l'effet tunnel qui peut apparaître dans un cycle en V. Scrum s'organise autour de réunions journalières et mensuelles afin de suivre l'évolution et les charges de chacun des développeurs. Ce terme provient historiquement du mot anglais « scrum » signifiant « mêlée » car son objectif est de souder une équipe autour d'un projet.

Sprint : période délimitée par les réunions ayant lieu toutes les 2 à 4 semaines durant laquelle un certain nombre de fonctionnalités doivent être programmées.

Sprint planning : réunion ayant lieu au début du « Sprint » pour planifier la programmation des fonctionnalités en leur attribuant des priorités et des charges. Il est ainsi possible d'établir en fonction du nombre de programmeurs et de jours travaillés par chacun d'eux une liste de fonctionnalités à livrer lors du « Sprint review ».

Sprint review : réunion ayant lieu à la fin d'un « Sprint » avec les partenaires et clients pour vérifier la livraison des fonctionnalités prévues lors du « Sprint planning ».

Test unitaire : programme informatique qui a pour but de vérifier le bon fonctionnement d'une méthode Java.

Workflow Jira : Dont la traduction est « cycle de vie », désigne un ensemble d'états et de transitions pouvant être parcourus par une demande comme « pris en charge », « en cours » ou « clôturée ».

Sommaire

ABREVIATIONS	1
GLOSSAIRE.....	3
INTRODUCTION	9
1. PROBLEME POSE ET TRAVAUX ANTERIEURS.....	11
1.1. CONTEXTE	11
1.1.1. <i>Un stage à l'étranger</i>	<i>11</i>
1.1.2. <i>Solutions Isonéo, une entreprise née d'une holding française</i>	<i>11</i>
1.1.3. <i>Citrus, un produit Java en cours de maturité</i>	<i>12</i>
1.1.4. <i>Une « start-up » en cours d'organisation.....</i>	<i>12</i>
1.2. CONTRAINTES	13
1.2.1. <i>Contraintes administratives</i>	<i>13</i>
1.2.2. <i>Contraintes de technologie</i>	<i>14</i>
1.2.3. <i>Contraintes de gestion de projet.....</i>	<i>16</i>
1.3. EXISTANT	16
1.3.1. <i>Scrum, méthodologie choisie par la société</i>	<i>16</i>
1.3.2. <i>Jira OnDemand.....</i>	<i>18</i>
1.3.3. <i>Git couplé à Github.....</i>	<i>20</i>
1.4. BESOIN : CONSTRUIRE LE SYSTEME D'INFORMATION D'UNE « START-UP » QUEBECOISE	23
1.4.1. <i>Respecter les contraintes imposées par le partenaire</i>	<i>24</i>
1.4.2. <i>Optimiser la gestion de projet.....</i>	<i>24</i>
1.4.3. <i>Automatiser la génération documentaire.....</i>	<i>25</i>
1.4.4. <i>Automatiser la création du produit.....</i>	<i>26</i>
1.4.5. <i>Evaluer la qualité du code au travers de métriques</i>	<i>28</i>
1.4.6. <i>Gérer la documentation</i>	<i>28</i>
1.4.7. <i>Centraliser le système d'information</i>	<i>30</i>
2. MISE EN ŒUVRE.....	33
2.1. CONCEPTION GLOBALE.....	33
2.1.1. <i>Organiser les besoins</i>	<i>33</i>
2.2. GESTIONNAIRE DE DEMANDES	35
2.2.1. <i>Optimiser la gestion du travail.....</i>	<i>35</i>
2.2.2. <i>Appliquer les contraintes imposées : migrer Jira sur le serveur de Solution Isonéo.....</i>	<i>47</i>
2.2.3. <i>Générer automatiquement la documentation</i>	<i>50</i>
2.3. ENVIRONNEMENT DE DEVELOPPEMENT	55
2.3.1. <i>Appliquer les contraintes du partenaire.....</i>	<i>55</i>
2.3.2. <i>Générer le produit.....</i>	<i>61</i>

2.4.	AIDE AU DEVELOPPEMENT	73
2.4.1.	Générer automatiquement le produit.....	73
2.4.2.	Appliquer le contrôle de la qualité du code.....	83
2.4.3.	Centraliser les artefacts « Maven ».....	91
2.5.	ACCOMPAGNEMENT AUX UTILISATEURS.....	95
2.5.1.	Centraliser la documentation.....	95
2.5.2.	Diffuser l'information	98
2.5.3.	Sécuriser l'information	100
2.6.	GESTIONNAIRE D'AUTHENTIFICATION.....	100
2.6.1.	Mettre en place un annuaire.....	100
2.6.2.	Coupler l'annuaire avec les applications.....	101
2.7.	SERVEUR.....	105
2.7.1.	Centraliser le système d'information	106
2.7.2.	Construire un serveur fiable et adapté aux besoins	116
3.	RESULTATS, DISCUSSIONS ET PERSPECTIVES	121
3.1.	SYSTEME D'INFORMATION COMPLET MIS EN PLACE	121
3.2.	RESULTATS POSITIFS	122
3.2.1.	Evaluer la documentation créée	122
3.2.2.	Respecter des règles définies sous Jira	123
3.2.3.	Accroître la réactivité de l'équipe.....	123
3.2.4.	Respecter des règles de codage	124
3.2.5.	Etablir une architecture projet	124
3.2.6.	Automatiser les lancements applicatifs du serveur.....	125
3.3.	GENERATION DOCUMENTAIRE PARTIELLEMENT AUTOMATIQUE	125
3.4.	AXES D'AMELIORATION DU SYSTEME MIS EN PLACE	128
3.4.1.	Migrer les applications pour respecter totalement les contraintes	128
3.4.2.	Installer un outil de GED.....	128
3.4.3.	Améliorer le processus de notification par Skype.....	128
3.4.4.	Modifier des paramètres de notifications sur Jira.....	129
3.4.5.	Ouvrir les applications sur Internet	129
3.4.6.	Sauvegarder les éléments générés à la compilation	129
3.4.7.	Exploiter au maximum des éléments du S.I.....	129
	CONCLUSION	131
	ANNEXES	133
	REFERENCES	136
	LISTE DES FIGURES	140

LISTE DES TABLEAUX.....	143
--------------------------------	------------

Introduction

Après 4 années de cours du soir au Conservatoire National des Arts et Métiers (CNAM) à Toulouse, je présente ce mémoire en vue d'obtenir mon diplôme d'ingénieur CNAM en informatique dans les systèmes d'information. Ce document décrit le contenu du stage que j'ai réalisé dans la société Solutions Isonéo au Canada dans la ville de Montréal.

Dans un premier temps, il présente l'analyse globale effectuée à mon arrivée. Ayant travaillé auparavant dans deux sociétés de services informatiques toulousaines, les caractéristiques de la société d'accueil ont une grande importance dans le déroulement de ce stage et dans mon apprentissage, elles sont décrites en suivant. Ce contexte implique des contraintes qui sont présentées dans une seconde sous-partie. Un système d'information ayant déjà été mis partiellement en place à mon arrivée, les composantes en sont détaillées dans une troisième sous-partie. L'analyse globale se termine par un éclaircissement des besoins de la société Solutions Isonéo concernant le système d'information à installer. En effet mon principal travail fut de construire le système d'information de la « start-up » québécoise avec des attentes bien précises.

Dans un second temps, la mise en œuvre de cette construction est exposée. La structure présentée témoigne du raisonnement suivi : une conception globale organise les besoins en précision et les mets en adéquation avec les connaissances des systèmes d'information acquises dans mes précédents emplois pour définir des « familles » d'outils pouvant répondre aux besoins. Chaque « famille » d'outils est ensuite associée à la liste des besoins auxquels elles peuvent répondre, permettant de vérifier que tous les besoins sont étudiés. Les sous-parties suivantes présentent pour chaque « famille » d'outils, la conception détaillée, la modélisation et l'implémentation réalisées pour chaque besoin associé. Certains points techniques bloquants et sur lesquels a dû être menée une réflexion plus profonde sont également décrits.

Pour finir, les résultats de cette mise en place sont présentés. L'objectif étant de détailler l'interconnexion entre les différents outils composant le système d'information de Solutions Isonéo et de vérifier ainsi la résolution de l'ensemble des besoins qui ont été listés dans la première partie de ce mémoire. Les points en suspens,

les évolutions qui pourraient encore être apportées à court terme et à moindre coût sont ensuite exposés ; mais également les évolutions que nous pouvons imaginer si les promesses des systèmes d'informations actuellement sur le marché étaient exploitées.

Ce mémoire conclut par une analyse et une réflexion sur l'organisation à établir autour d'un système d'information.

1. Problème posé et travaux antérieurs

1.1. Contexte

1.1.1. Un stage à l'étranger

Après 4 années de cours du soir, le stage pour obtenir mon diplôme d'ingénieur CNAM s'est présenté comme une opportunité pour partir à l'étranger. J'ai choisi le Québec au Canada. Ma priorité étant l'obtention de mon diplôme d'ingénieur, je me suis orientée vers un pays principalement francophone afin que la barrière de la langue ne soit pas un frein. Le travail et la vie quotidienne avec des anglophones ont cependant été également une motivation afin de m'améliorer dans ce domaine.

Pour ce stage j'ai été accueillie par la société Solutions Isonéo située à Montréal.

1.1.2. Solutions Isonéo, une entreprise née d'une holding française

Solutions Isonéo a été créée par Pierre Duverneuil et Raphaël Giraud, tous deux membres d'une holding dont le siège est à Toulouse, ayant pour assistante de gestion Florence Serié résidant à Paris. Cette holding est issue d'Artal Technologies, société de service informatique, présente à Labège près de Toulouse et proposant entre autres de la sous-traitance en programmation informatique pour Airbus.

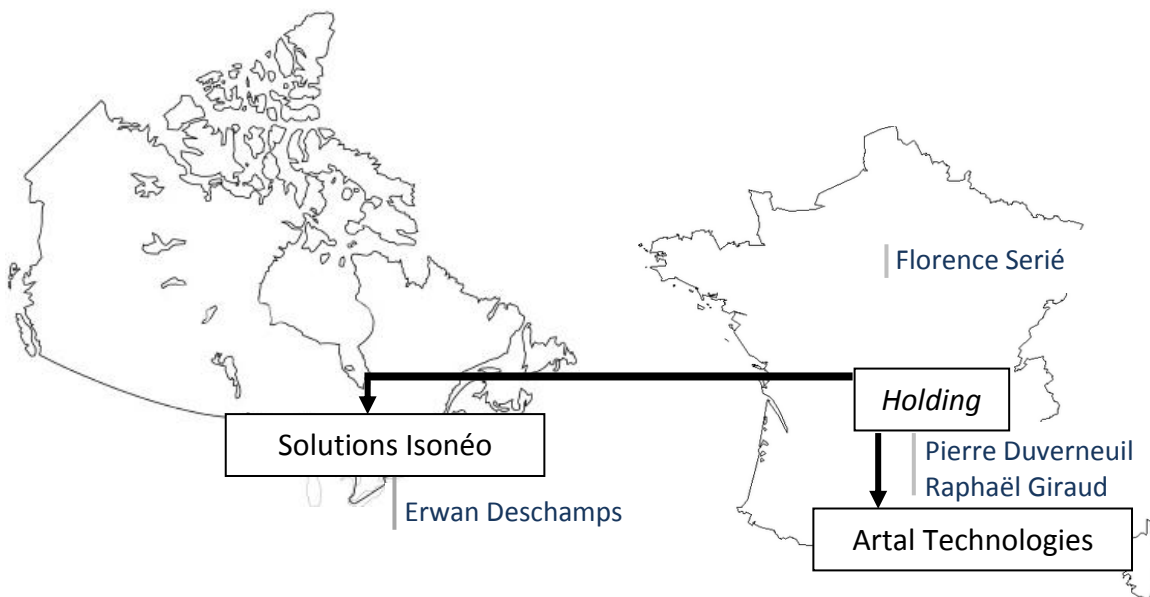


Figure 1 : Organigramme haut niveau

Ayant une grande expertise en aéronautique, les deux dirigeants ont décidé de l'exporter au Canada pour répondre aux besoins d'entreprises telles que Bombardier ou Pratt & Whitney.

Solutions Isonéo est restée au stade d'étude pendant plusieurs mois avec l'arrivée d'Erwan Deschamps, salarié d'Artal Technologies dans le cadre d'un Volontariat International en Entreprise (V.I.E.) à Montréal pour la prospection commerciale. Après l'obtention d'un partenariat avec la société montréalaise CMC Electronique intéressée par les activités d'Artal Technologies pour Airbus, la société Solutions Isonéo a pu intégrer le programme gouvernemental SA²GE (Systèmes Aéronautiques d'Avant-Garde pour l'Environnement) pour participer au projet d'avion « vert » c'est-à-dire à la création d'un avion le plus écologique possible. Ce projet réunit les grands donneurs d'ordre de la région montréalaise (Sitographie [1]).

Solutions Isonéo participe à ce programme en développant un outil, nommé Citrus, qui est sa propriété et pourra donc être vendu. CMC Electronique est donc le premier utilisateur de cet outil.

1.1.3. Citrus, un produit Java en cours de maturité

Solutions Isonéo n'est pas une société de services mais une société souhaitant vendre un produit que les programmeurs sont en charge de porter à maturité pour la fin de l'année 2015. Le partenaire CMC Electronique avait besoin de personnes compétentes sur la programmation dans le domaine aéronautique pour pouvoir développer un outil permettant la gestion de modèles. Cet outil est nommé Citrus.

Ce projet étant un projet gouvernemental de recherche et développement, je ne m'attarderai donc pas à la description de l'outil développé par l'équipe de programmation.

1.1.4. Une « start-up » en cours d'organisation

Une fois le partenariat mis en place, une seconde personne, Julien Siega, a été transférée de la société française vers la filiale canadienne pour gérer la partie programmation du projet. J'ai ensuite intégré cette équipe pour construire le système d'information quasi inexistant de la nouvelle société. Deux autres personnes ont été embauchées peu de temps après mon arrivée pour épauler Julien en programmation.

Les différents pôles de Solutions Isonéo se dessinent de la façon suivante :

- ❖ **Commercial et management** en charge d'Erwan Deschamps pour obtenir de nouveaux contrats mais aussi pour faire l'interface avec la holding française, gérer les budgets, les achats et assurer la visibilité de la société afin que le produit développé puisse obtenir ses clients rapidement lorsqu'il aura atteint sa maturité.
- ❖ **Programmation** principalement portée par le « Scrum master » : Julien Siega, qui assure les échanges avec le partenaire CMC Electronique. Il assure la cohésion de l'équipe et la tenue du planning.
- ❖ **Systèmes d'information** dont j'ai pris la charge en arrivant dans la société (Bibliographie [1]). Celui-ci a été créé prioritairement pour les programmeurs. De nombreux échanges ont donc eu lieu tout au long de ce stage pour exprimer les besoins mais aussi évaluer le rapport charges / bénéfices.

Le pôle réseau est pour l'instant inexistant car la société est encore trop petite. Pour le moment le besoin n'est que ponctuel pour affecter une ressource à cette tâche.

1.2. Contraintes

1.2.1. Contraintes administratives

a) Synergie

Même si la création d'un système d'information est très formateur, le lien fort avec la société Artal Technologies et le parallèle avec leurs processus a rapidement impliqué une réutilisation de l'existant pour une adaptation dans la société naissante.

Les deux entreprises étant de taille différente - Artal Technologies ayant un effectif de 150 personnes environ - il était important de se questionner à chaque étape sur la rigidité engendrée par la réutilisation des processus existants chez Artal Technologies afin que le potentiel de croissance de la petite structure soit conservé.

b) Temps

Mon stage s'est déroulé de fin janvier 2013 à mi-décembre 2013 et a impliqué ma première visite au Canada. Ne sachant pas si je souhaiterai rester dans la société après la

fin de mon stage, il fallait que j'accorde une grande importance à la documentation de mon travail afin qu'il puisse être repris rapidement et que les outils puissent être maintenus facilement.

Il fut également primordial d'accorder de l'importance aux fonctionnalités à développer. En effet de nouveaux besoins ont émergé au fur et mesure de l'année avec des priorités changeantes. Le rapport charges / bénéfices fut souvent évalué dès que la complexité s'élevait.

c) Confidentialité

Le projet SA²GE étant un projet gouvernemental, le partenaire CMC Electronique a imposé que les sources et la documentation soient à court terme stockées dans les locaux de Solutions Isonéo et protégées autant que possible afin de conserver la confidentialité du projet.

1.2.2. Contraintes de technologie

a) Normes

L'aéronautique, secteur pour lequel Solutions Isonéo travaille principalement à travers le programme SA²GE, possède plusieurs normes évaluant la qualité d'un projet. La norme utilisée dans notre cas fut la « DO-330 ».

Cette norme détaille l'intégralité des documents à fournir si la société souhaite obtenir la certification et explique les différents niveaux de qualification possibles en fonction de l'activité et du besoin de la société.

b) Java et Eclipse

Eclipse est un IDE (Integrated Development Environment) fournissant aux programmeurs un environnement de travail sous forme d'une application qui embarque plusieurs fonctionnalités comme entre autres : la compilation, l'aide à la programmation, l'aide à la rédaction de tests. Aujourd'hui une communauté active existe autour de cet outil et des projets sont développés par celle-ci dans le monde entier. Les projets sont développés en Java, langage de programmation avec lequel est développé Eclipse lui-même.

1. Problème posé et travaux antérieurs

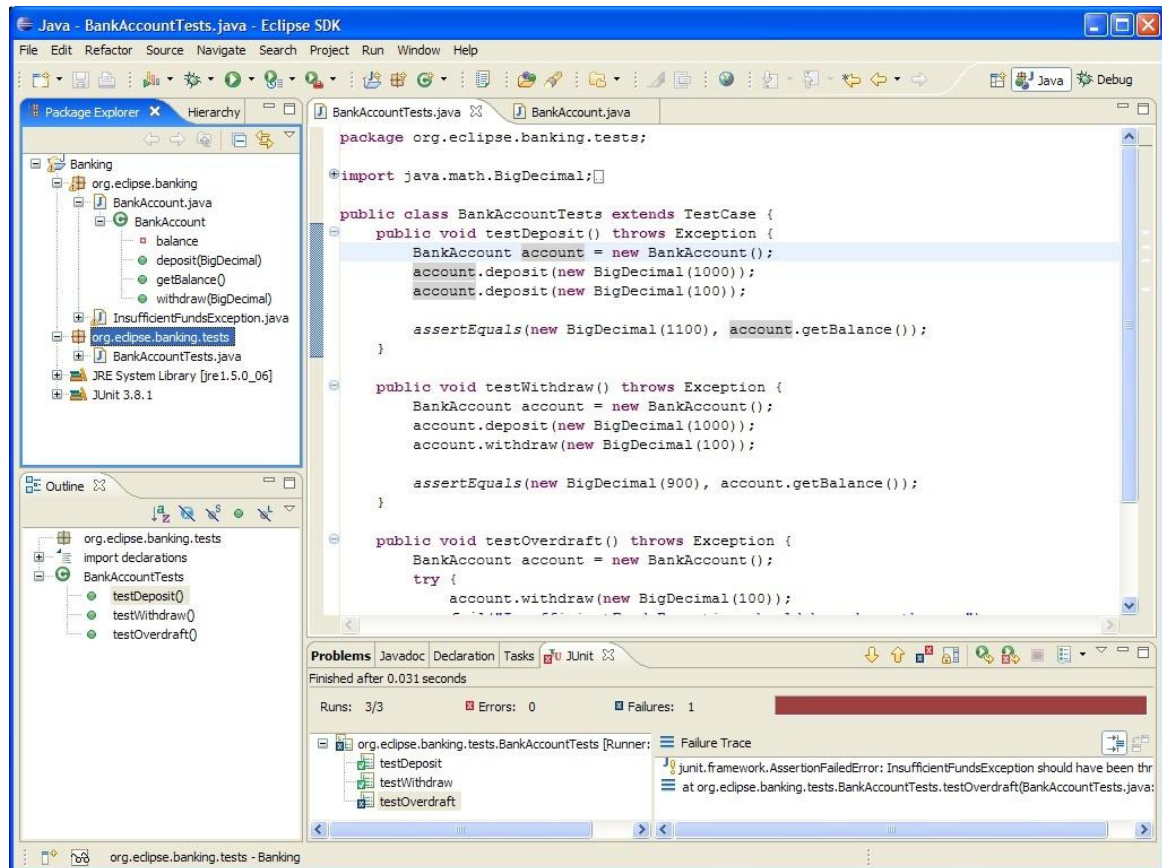


Figure 2 : Différents panneaux composant l'IDE Eclipse

L'outil devant être livré par Solutions Isonéo dans le cadre du programme SA²GE est développé en Java avec Eclipse Juno car ce projet réutilise des composants développés par la communauté (Obeo, SysML, AADL).

Les outils composant le système d'information doivent donc respecter cette contrainte afin de compiler, tester, et s'interfacer correctement avec du code Java et afin de réutiliser les compétences des programmeurs si il arrivait que la société ait besoin de développer des plug-ins pour ces outils.

1.2.3. Contraintes de gestion de projet

a) Budget

Mon travail a été d'intégrer une petite société dont les moyens sont beaucoup plus modérés que ceux d'une grande entreprise. De plus, étant naissante, Solutions Isonéo est épaulée financièrement par la société mère lorsque cela est nécessaire.

Due à cette contrainte, la base du système d'information avait été orientée vers des produits OpenSource performants et à moindre coût sur lesquels beaucoup de support en ligne est fourni. Nous avons fait le choix de poursuivre dans cette voie.

1.3. Existant

1.3.1. Scrum, méthodologie choisie par la société

La société naissante a dû choisir une méthodologie pour organiser son travail afin de fournir à tous les futurs projets une ligne directrice pour leur organisation et leur évolution.

Tous les projets « internes » liés à de l'activité informatique (réseau, programmation et système d'information) devront suivre la méthodologie Scrum adoptée par Solutions Isonéo (Bibliographie [2] et [3]). En effet Scrum pourrait ne pas être retenu pour des projets liés directement à un client imposant sa méthodologie.

Scrum est une des méthodes agiles les plus connues (Sitographie [2]). Les méthodes agiles aident à la gestion de projets principalement de conception logicielle, en promouvant une forte implication du client augmentant ainsi la réactivité de l'équipe de développement face aux réels besoins du demandeur. Plusieurs méthodes agiles existent telles que Scrum, RAD, Extreme programming.

Les méthodes agiles sont opposées à la gestion du projet suivant le cycle en V. En effet cette dernière méthode implique un long cycle avec peu de réunions. Si le projet dure plusieurs longs mois, le produit livré peut ne plus correspondre aux évolutions des technologies ayant eu lieu durant le développement du produit. Les méthodes agiles promeuvent des itérations rapprochées afin de maîtriser l'évolution de ce produit.

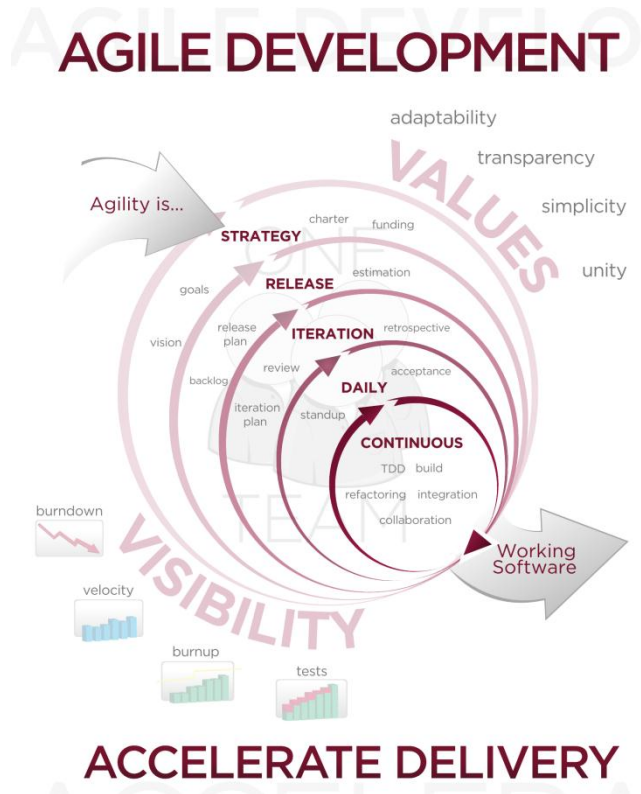


Figure 3 : Modèle des méthodes agile

Solutions Isonéo a porté son choix sur la méthode Scrum. Travaillant avec un partenaire sur le programme SA²GE, la société CMC Electronique est considérée comme cliente. Cette dernière définit une liste de souhaits pour le futur produit que l'on appelle dans la méthode un « product backlog ». Le travail est alors organisé autour de « Sprint ». Un « Sprint » dure, dans cette société, environ un mois – un « Sprint » courant normalement entre 2 et 4 semaines – période durant laquelle l'équipe décide d'implémenter les souhaits prioritaires durant le temps imparti et choisi de reporter les autres. Tous les jours, tous les membres de Solutions Isonéo se réunissent dans la matinée lors du « Daily Scrum » afin de discuter du travail effectué la veille et des travaux prévus pour la journée. Au bout du « Sprint », CMC Electronique et l'équipe de développement se réunissent lors du « Sprint review » pour vérifier, discuter et acter du travail effectué durant ce dernier. Cela permet de ne pas travailler trop longtemps dans une direction sur laquelle les deux protagonistes se seraient mal compris. Une autre réunion sera planifiée quelques jours après, le « Sprint planning », afin de discuter des objectifs pour le prochain « Sprint ».

Cette méthodologie a imposé le choix de deux outils du système d'information présent à mon arrivée :

- ❖ Jira OnDemand pour la gestion de demandes
- ❖ Git pour la gestion des versions

1.3.2. Jira OnDemand

Jira est un outil de gestion de projet et de suivi de bugs appelé « gestionnaire de demandes ». Il en existe plusieurs sur le marché comme ClearQuest d'IBM ou encore Remedy pour les plus connus. Jira est développé par la société australienne Atlassian proposant un logiciel et un service de support gratuit avec des fonctionnalités payantes mais à des prix très raisonnables. Il est porté par une large communauté OpenSource qui nourrit considérablement les tutoriels et discussions sur le web. Ce logiciel est développé en Java présentant ainsi un avantage notable : si nécessaire, les programmeurs de Citrus peuvent plus facilement réaliser des plugins pour cet outil. Enfin, c'est un logiciel éprouvé car utilisé aujourd'hui par de nombreuses sociétés dans le monde.

Jira permet de gérer des demandes. Ce type d'outil est aussi identifié par « outil de ticketing ». En effet Jira permet de créer des enregistrements afin de tracer toutes demandes concernant un outil afin d'obtenir du support, de l'administration ou d'organiser les fonctionnalités de l'outil à développer. L'objectif est d'abandonner l'utilisation des mails afin de ne plus perdre la trace de certains échanges et d'éviter les demandes sans réponse.

Jira intègre également des tableaux de bords composés de rapports et graphiques afin de suivre l'avancement du travail sur ces demandes. Il est ainsi possible de savoir combien de temps une personne a passé sur une demande donnée.

Ce gestionnaire de demandes est donc un outil complet pour le chef de projet qui souhaite suivre l'activité mais aussi pour les « clients » ou « demandeurs » qui ont souhaité une action de la part d'une équipe et qui attendent un retour de celle-ci.

Jira propose une interface web pour réaliser cet échange.

1. Problème posé et travaux antérieurs

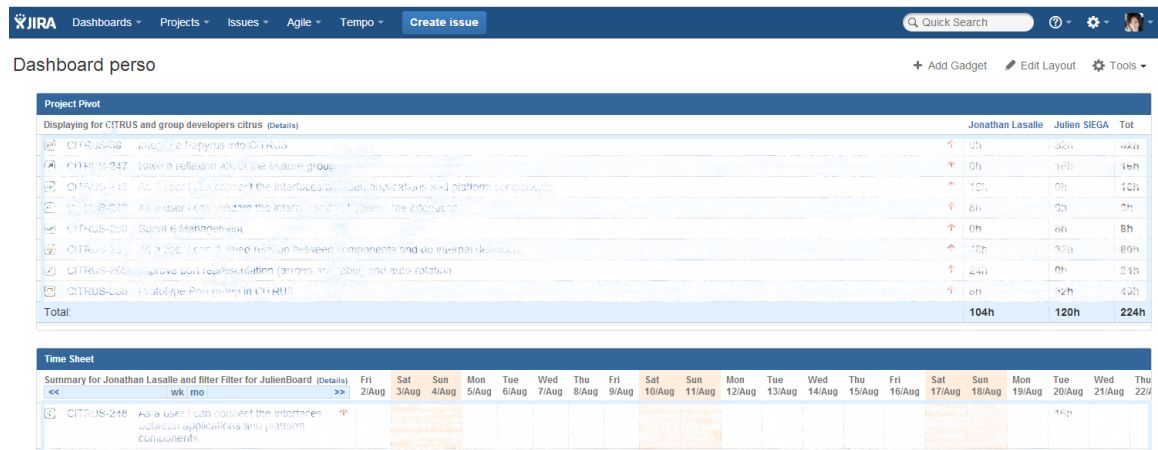


Figure 4 : Interface web de Jira

Par exemple, cet outil pourrait très bien être utilisé par l'équipe en charge du support du client de messagerie Google : Gmail. Un utilisateur Gmail pourrait alors créer une demande pour obtenir du support sur l'envoi d'un mail en tant que message urgent. Jira possède un système de notification paramétrable qui en ce cas enverrait un mail à l'équipe en charge des fonctionnalités de Gmail. Une personne compétente pourrait alors prendre en charge la demande, la faire évoluer suivant un cycle de vie (workflow) pour la faire passer au statut « Résolu » par exemple et en déposant des commentaires pour expliquer que cela n'est pour l'instant pas réalisable sur l'application en question. Le système de notification permettrait également d'informer le demandeur de chaque évolution ou dépôt de commentaire sur la demande enregistrée dans Jira.

Jira correspond totalement à l'activité de Solutions Isonéo car l'outil englobe des fonctionnalités pour l'utilisation des méthodes agiles. Il permet de tracer les actions de chaque « Sprint », de visualiser ensuite rapidement les tâches qui ont été effectivement terminées à temps et celle qui sont reportées au prochain « Sprint ». La solution a donc été conservée pour la construction du système d'information.

Pour des raisons de facilité et de rapidité de mise en place, Solutions Isonéo utilisait à mon arrivée la version appelée Jira OnDemand hébergée par la société créatrice du logiciel, Atlassian.

Ci-dessous, une vue d'ensemble provisoire des éléments qui composaient le système d'information de Solution Isonéo à mon arrivée.

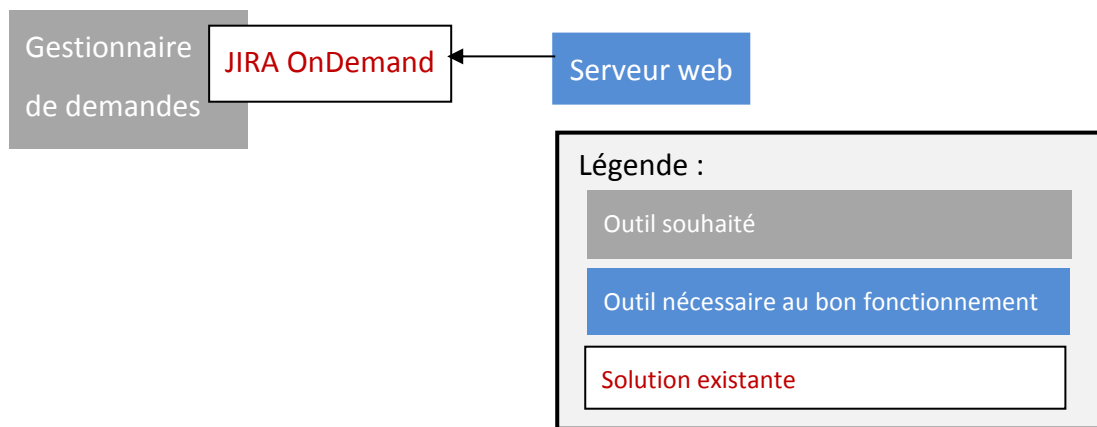


Figure 5 : Composant du système d'information existant : « Jira OnDemand »

1.3.3. Git couplé à Github

Git est un gestionnaire de versions, aussi appelé gestionnaire de sources. Il est totalement gratuit, porté par la communauté OpenSource et sa parution récente lui a permis de proposer une gestion réellement différente des sources. Il devance aujourd'hui largement son concurrent payant ClearCase proposé par IBM ou son concurrent gratuit SVN, plus ancien. Comme Jira, Git est éprouvé car il est utilisé dans de nombreuses entreprises dans le monde.

Git permet d'enregistrer tout le code développé par les programmeurs sur un serveur distant et ainsi le sauvegarder. Il y ajoute des fonctionnalités communautaires afin de pouvoir travailler sur les mêmes sources sans écraser le travail de son collègue et de pouvoir revenir à une version antérieure. Git est également un système sécurisé car un nom d'utilisateur et un mot de passe sont demandés à chaque connexion vers le dépôt central.

Du point de vue utilisateur, Git permet une gestion du code source. L'utilisateur récupère ce qui est présent dans le dépôt via une action de clonage dans un dossier sur son ordinateur. Ensuite il travaille sur sa propre version de sources. Git gère les nouveaux fichiers, les modifications des fichiers existants, leurs suppressions, déplacements, etc. Quand le programmeur commence à modifier ces sources, il travaille

« en local », c'est-à-dire avec du code qui n'est pas encore mis en commun avec les autres développeurs et qui n'est pas pris en compte par Git. Ce qu'il est important de comprendre, c'est que le dépôt récupéré par l'action de clonage ou par la suite par les actions « pull » est stocké physiquement exactement à l'endroit où le programmeur va appliquer ses modifications. Sur le schéma suivant cette particularité est matérialisée par un filtre parallèle au dépôt cloné. A tout moment, le programmeur peut appliquer l'action « commit » sur ses modifications, c'est-à-dire les faire prendre en compte par Git. La puissance de Git réside dans cette action qui ne dépose pas les sources sur le serveur distant avec les sources des autres programmeurs mais qui les sauvegarde malgré tout. De ce fait, si le lien entre le serveur possédant le dépôt et la machine de l'utilisateur est coupé, le programmeur peut poursuivre son travail et se reconnecter plus tard au dépôt distant. Une fois que le développeur a pu tester son code, il peut ensuite le mettre en commun et faire un « push ». Les autres développeurs pourront récupérer les modifications via un « pull ».

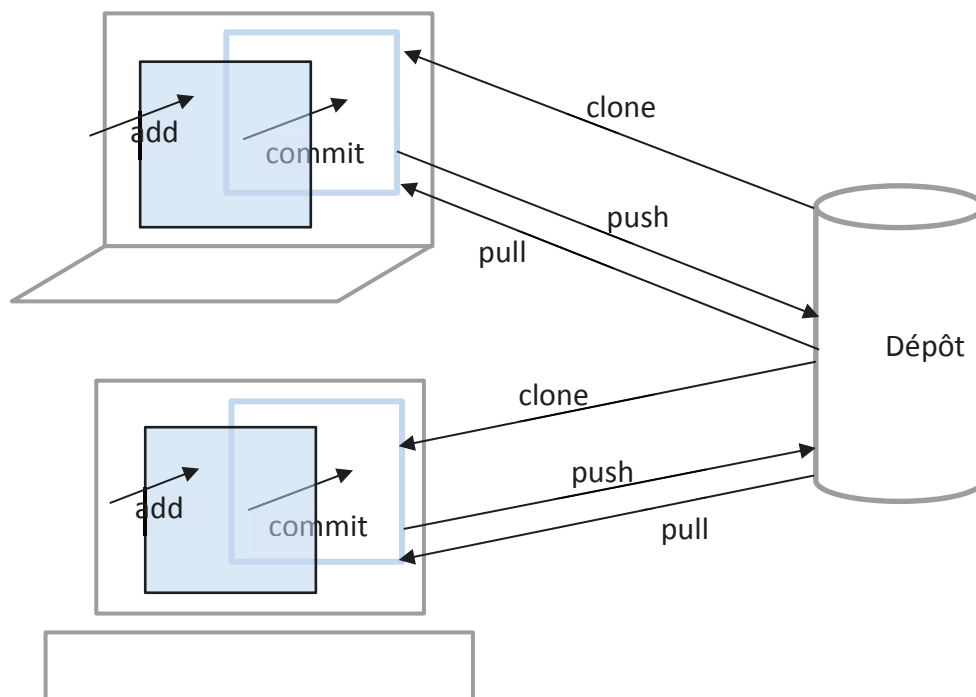


Figure 6 : Schéma de fonctionnement de Git

Ce type d'outil s'impose automatiquement lors d'un développement logiciel avec des équipes composées de plusieurs personnes.

Solutions Isonéo a également choisi cette technologie car elle était déjà connue des premiers programmeurs de la société.

Pour des raisons de praticité, Solutions Isonéo a opté dans un premier temps pour un dépôt de source Git hébergé sur un serveur appartenant à la société créatrice de la technologie, c'est-à-dire sur un serveur Github.

Github propose une interface web pour gérer votre dépôt, consulter les derniers « commit » et l'architecture du projet entre autres.

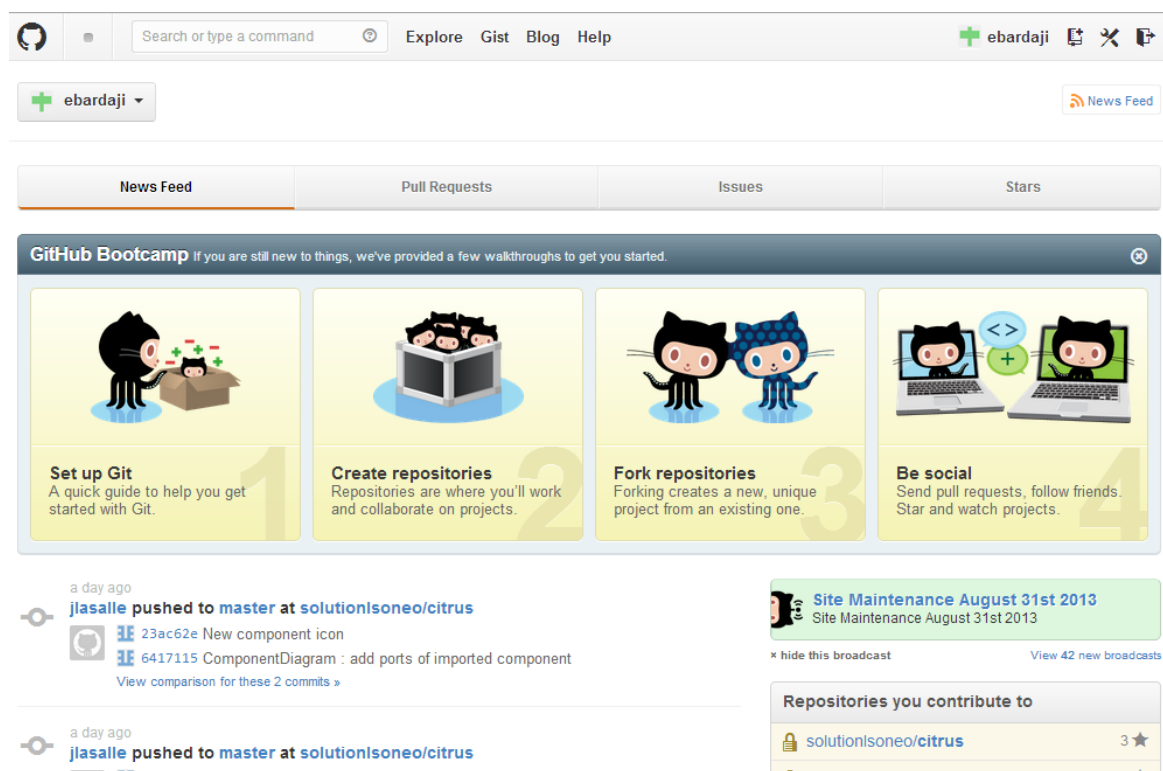


Figure 7 : Interface web de Github

Le système d'information de Solutions Isonéo est donc déjà composé des éléments suivants : Jira OnDemand qui possède sa propre interface web et Github qui permet d'avoir une interface web pour gérer le serveur Git hébergé à distance.

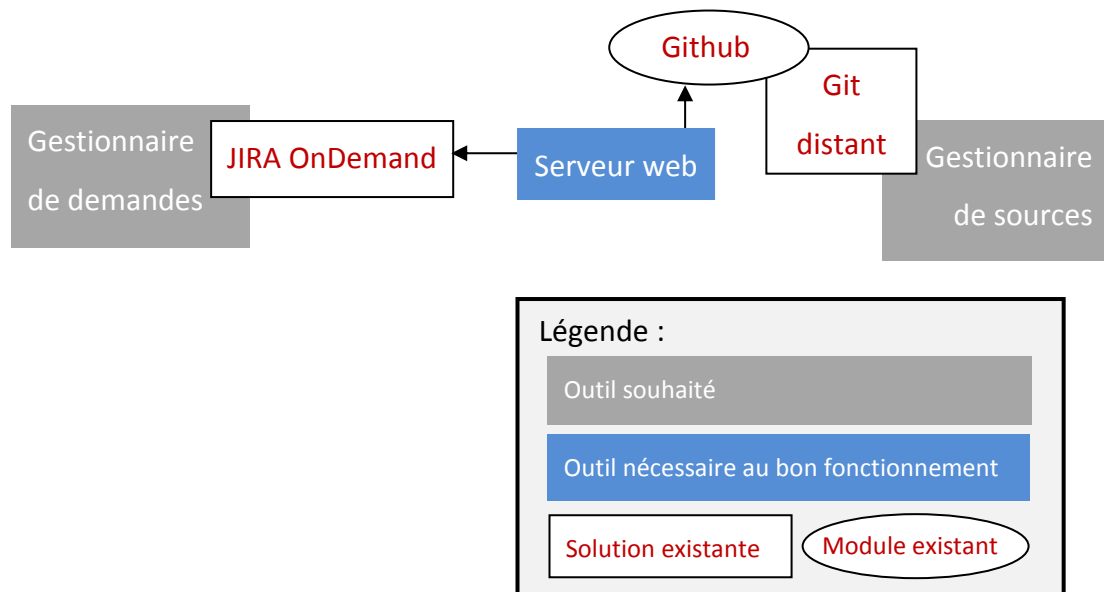


Figure 8 : Composants du S.I. existant : prise en compte de « Git distant »

1.4. Besoin : Construire le système d'information d'une « start-up » québécoise

Erwan Deschamps a décrit une partie de mon stage par : « *Solutions Isonéo est la filiale d'Artal Technologies basée à Montréal nouvellement créée. Pour assurer sa croissance, la structure canadienne doit mettre en place toute une infrastructure de systèmes d'information.* »

Associé aux contraintes décrites précédemment, il s'agit donc de mettre en place un système d'information fonctionnel performant à moindre coût basé sur le langage de programmation Java dans une « start-up » québécoise qui entretient un lien étroit avec sa jumelle française en un temps limité puis de l'optimiser durant le temps restant pour répondre au mieux à la norme aéronautique DO-330 entre autres.

Un « système d'information » est un ensemble de ressources interconnectées afin de traiter, stocker, et échanger des éléments de connaissances (code, documentation, etc.). Ce terme désigne donc un large périmètre d'outils et de possibilités dans leurs interconnexions. Nous avons décidé de préciser les besoins et les priorités au fur et à mesure de mon stage.

Ce sujet requiert des connaissances techniques très variées mais aussi de multiples études pour chaque outil.

Le système d'information doit être adapté au contexte dans lequel évolue Solutions Isonéo : réduite mais qui a l'intention de s'agrandir rapidement. De même, devant partir à la fin de mon stage et l'embauche d'un ingénieur en système d'information n'étant pas prévu, le système doit être facilement récupérable et maintenable par un programmeur. Le système d'information à mettre en place doit donc être simple à utiliser, sans trop de contraintes fonctionnelles et techniques, le plus automatisé possible et documenté.

Une fois ce sujet défini, j'ai pu recenser les besoins des membres de Solutions Isonéo pour eux-mêmes, leurs clients ou partenaires.

1.4.1. Respecter les contraintes imposées par le partenaire

Le système d'information de Solutions Isonéo avait déjà été initialisé avec deux outils précédemment présentés : Jira et Git. Cependant aucune de ces solutions ne respectaient la contrainte imposée par le partenaire CMC Electronique : la contrainte de confidentialité. En effet les deux solutions étaient hébergées sur des serveurs distants. Afin de respecter cette contrainte, Solutions Isonéo eu besoin d'une migration de ces deux solutions vers notre serveur.

1.4.2. Optimiser la gestion de projet

a) Uniformisation des processus entre les deux projets

A mon arrivée dans la société, un projet nommé « SI » pour « système d'information » avait été créé dans Jira. Des tâches y avaient déjà été enregistrées et m'étaient affectées. Les deux personnes présentes dans la société à ce moment n'ayant aucune connaissance des possibilités offertes par le gestionnaire de demandes, ma première tâche fût d'uniformiser les processus entre les deux projets existants : « Citrus » hébergeant toutes les tâches des programmeurs pour le produit « Citrus » et « SI » traçant toutes les actions pour créer le système d'information souhaité.

b) Amélioration de la gestion de projet

Un gestionnaire de demandes a pour principal objectif de fournir facilement un suivi de l'activité de chacun des membres de l'équipe sur chacun des projets. Erwan Deschamps

devait maîtriser l'outil pour obtenir rapidement des rapports d'activité en fonction des besoins du partenaire et de la holding française et ainsi améliorer la gestion de projet. Mon objectif a été de fournir des rapports pré-paramétrés pour répondre aux besoins et fournir à Erwan Deschamps de la documentation et une formation minimale pour les manipuler.

c) Optimisation des processus

L'équipe Solutions Isonéo enregistre toutes leurs actions dans le gestionnaire de demandes Jira OnDemand qui propose nativement plusieurs familles d'actions (bug, demande, évolution, etc.). Tous n'étant pas utiles aux besoins de la société, l'objectif dans un premier temps fut de limiter les erreurs et de supprimer les types d'enregistrements inutiles.

Pour compléter l'étape de suppression, il était également nécessaire de recenser les besoins de l'équipe et les documents externes au gestionnaire de demandes afin de pouvoir centraliser les informations à l'intérieur de l'application et ainsi optimiser les processus.

1.4.3. Automatiser la génération documentaire

Solutions Isonéo a décidé de respecter la norme DO-330 qui est une norme aéronautique régissant la production documentaire autour d'un projet de développement.

La rédaction de documents étant rarement une tâche agréable pour un programmeur, Solutions Isonéo souhaite automatiser la production de ces documents.

Les documents devant être produits pour la DO-330 comme pour la plupart des normes informatiques sont toujours composés des mêmes informations. Solutions Isonéo souhaite que ces informations soient récupérées dans les différents outils composant le système d'information afin d'être insérées automatiquement dans les fichiers nécessaires à l'obtention de la certification DO-330.

Solutions Isonéo étant une société naissante, l'intégralité de la norme ne s'y applique pas. Le besoin s'organisait donc déjà autour de plusieurs axes :

- ❖ Etudier et comprendre la DO-330,
- ❖ Vérifier la pertinence des éléments de la DO-330 pour les processus de Solutions Isonéo. Cette tâche a été réalisée par des membres d'Artal Technologies,
- ❖ Détailler le contenu de chaque document pouvant répondre à la DO-330 dans le cas de la société Solutions Isonéo,
- ❖ Trouver l'endroit puis la façon de récupérer ce contenu.

Pour tous les documents à créer manuellement, j'ai échangé avec Artal Technologies pour réutiliser au maximum leurs concepts qualités.

L'objectif de cette tâche est d'optimiser le système d'information de Solutions Isonéo en y ajoutant un « module » de génération automatique de documents pour la DO-330.

1.4.4. Automatiser la création du produit

La première grande tâche dans la construction du système d'information fut d'automatiser la création du produit de l'application Citrus. Il est construit grâce à l'IDE Eclipse présenté précédemment.

a) Définition d'un produit

Eclipse donne la possibilité de générer un fichier contenant toute la configuration, composantes et règles d'exécution du projet afin de pouvoir générer un nouvel exécutable sur n'importe quel autre Eclipse Juno. Ce fichier est généré à la demande des programmeurs lors de la compilation via l'IDE. Il se nomme « produit » et possède l'extension .product. De nombreuses informations ont été trouvées dans la Sitographie [3].

Ce .product permet à d'autres développeurs de retrouver toutes les règles permettant de lancer l'application « Citrus » grâce au mécanisme interne de génération d'Eclipse s'ils ont récupéré les sources Java à partir du Git.

A partir de ce produit il est également possible de générer un exécutable avec une extension .exe permettant de lancer « Citrus » sans avoir accès au code.

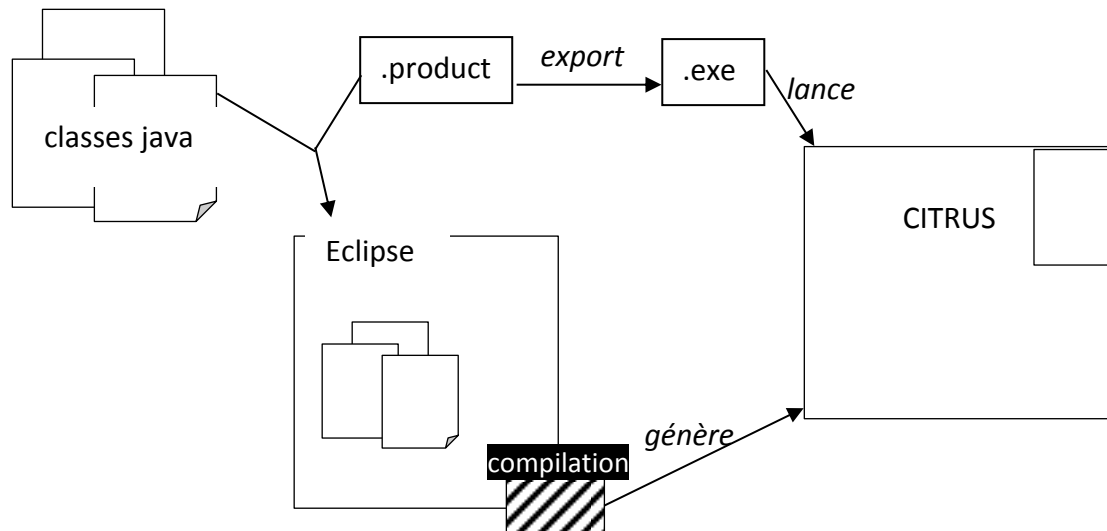


Figure 9 : Environnement de génération de Citrus

L'objectif est de générer automatiquement ce produit afin d'avoir toujours une version vérifiable à jour.

Dès que le terme « automatisation » est employé à propos du système d'information, l'intégration continue doit être considérée.

b) Définition de l'intégration continue

L'intégration continue permet de compiler le projet automatiquement à période fixe. Il permet de vérifier que le code déposé sur le gestionnaire de versions – dans notre cas Git - durant cette période n'applique pas de « retour en arrière » sur le bon fonctionnement de l'application, ce qui est appelé : tests de non-régression.

Ce type d'outil répond parfaitement au besoin de génération automatique du produit. Cependant Solutions Isonéo génère le produit Citrus via un compilateur intégré à Eclipse qui ne peut être utilisé par l'outil d'intégration continue.

L'intégration continue requiert donc la mise en place d'un outil de compilation qui interprètera automatiquement le code lorsque l'outil d'intégration continue le lui demandera.

1.4.5. Evaluer la qualité du code au travers de métriques

Le suivi de la qualité peut être appliqué sur de nombreux aspects dans un projet de programmation. Deux aspects primordiaux sont cependant importants à mettre en place.

a) Evaluation de la couverture de tests

Pour assurer aux clients et partenaires que le code mis en place par les programmeurs chaque jour est bien fiable, il faut leur assurer que des tests sont mis en place au fur et à mesure du code de l'application et que ces tests sont concluants.

L'objectif est de produire des rapports assurant que la totalité du code de l'application est testée et fournissant l'état de ces tests (succès ou erreur).

b) Application de règles de codage

L'autre aspect primordial dans la programmation est le respect de règles de codage. De nombreuses règles sont applicables. Solutions Isonéo a décidé de ne respecter que certaines d'entre elles comme la mise en place de commentaires Javadoc, la mise en place d'un en-tête particulier contenant le copyright ou encore un format de nommage pour les variables de classes.

Pour le contrôle de ces règles, des outils pouvant être intégrés directement dans Eclipse et/ou des outils émettant des rapports existent et devaient être intégrés au système d'information en construction pour Solutions Isonéo.

1.4.6. Gérer la documentation

Ce besoin n'était pas clairement défini au début de mon stage. Cependant, dès mon arrivée, le manque de structure pour l'archivage de la documentation fut ressenti.

Etant en charge de la mise en place d'un système d'information utilisé par les programmeurs et manipulé par les ressources humaines, j'ai rapidement eu besoin de me documenter sur les activités des deux pôles déjà en place.

Etant nouvelle employée de la société, j'ai également dû prendre connaissance des processus pour enregistrer des congés, tracer une absence maladie ou saisir l'activité.

Toute cette documentation fut difficile à trouver car aucun système ne la centralisait.

a) Assurance d'une capitalisation sur le long terme

La seule référence existante était la gestion documentaire mise en place dans la société Artal Technologies. Solutions Isonéo allant grandir, il fallait également considérer que le nombre de projets pouvait se multiplier et ainsi faire naître des documents confidentiels et propres à un seul projet. Comme dans toute structure, des documents seraient liés à la gestion interne de la société (congés, maladie, etc.).

L'objectif était de créer un référentiel documentaire simple, complet et pouvant évoluer facilement. Comme toute structure naissante, il n'y a pas de personnes pouvant être chargées de la gestion documentaire et de la qualité de celle-ci. Il est donc de la charge de chacun des développeurs de maintenir ces documents même si ce n'est pas leur fonction première et qu'ils ont peu de temps à allouer à ces tâches.

Tout en considérant ce support existant et ces contraintes, la mise en place d'un système de gestion documentaire s'est composée de deux étapes :

- ❖ Définir les types de documents existants ou nécessaires pour leur attribuer un identifiant directement inséré dans le titre du document
- ❖ Créer des modèles pour chacun d'eux afin de définir les informations utiles mais également le style du document (sommaire, en-têtes, etc.)

b) Stockage de cette base de connaissance

Dans un second temps, le système d'information se mettant en place, une de ses composantes logiques fut le Gestionnaire Electronique de Documents appelé plus communément GED.

Un outil de GED présente des avantages incontestables par rapport à une gestion de la documentation qui consiste à déposer des fichiers sur un serveur dans une architecture de dossiers.

L'application permet de sauvegarder automatiquement les différentes versions du document et ainsi revenir facilement à une version précédente ou la consulter. Il permet aussi le travail collaboratif qui assure que le travail d'une personne ne sera pas supprimé

par l'enregistrement d'une autre personne sur le même document. Ces outils proposent un système d'authentification délimitant ainsi les différents périmètres en fonction des projets et de la gestion interne de la société ; de cette façon les documents confidentiels sont accessibles uniquement par les gens qui en ont la permission.

Aucun choix n'a été fait antérieurement sur ce type d'outil. La priorité, suivant les contraintes, était de trouver un outil gratuit.

1.4.7. Centraliser le système d'information

Afin d'obtenir un système d'information stable pour l'avenir, la création d'une architecture réseau était indispensable. En effet des données confidentielles concernant le projet SA²GE transitent depuis le début par Solutions Isonéo : le code source de l'application Citrus est stocké sur des architectures externes afin d'utiliser Git, des informations sur l'évolution du produit sont enregistrées à distance également dans Jira OnDemand. Le système d'information devait donc être supporté par une architecture centralisée.

a) Architecture à modifier

L'architecture initiale dépendait de la SICAM qui gère l'internet pour la pépinière d'entreprises. Les postes des membres de Solutions Isonéo sont donc directement reliés au réseau Internet comme dans un réseau domestique via un wifi.

Les deux serveurs externes utilisés pour Git et Jira OnDemand sont accessibles via Internet.

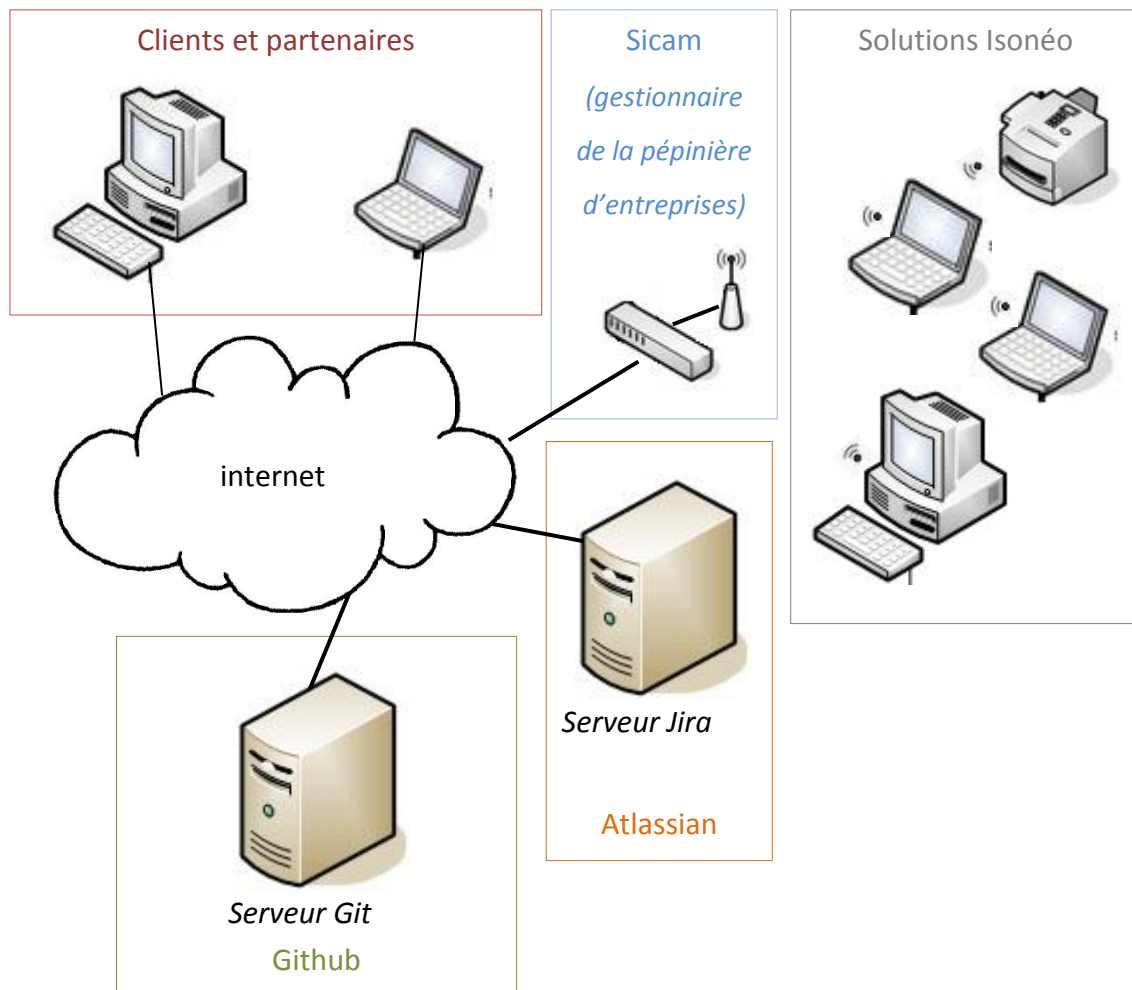


Figure 10 : Représentation du réseau fourni par la Sicam et utilisé par Solutions Isonéo

La mise en place d'une telle architecture passe par l'achat d'un serveur avec des capacités en rapport avec l'évolution de Solutions Isonéo mais un accord doit être obtenu de la holding française qui peut engendrer des délais plus importants pour obtenir le matériel.

b) Modifications nécessaires

Ce besoin engendre également d'autres besoins secondaires comme le choix d'un système d'exploitation, d'une infrastructure réseau, d'outils pour la sécurité réseau et d'un système d'authentification.

Pour respecter les contraintes, les migrations des sources hébergées sur le serveur externes pour l'utilisation de Git ainsi que toutes les données enregistrées sous Jira OnDemand devaient être migrées respectivement vers un nouveau dépôt Git hébergé sur le serveur de Solutions Isonéo et sur un Jira « hosted ».

L'objectif final de cette centralisation était également de donner accès aux outils composant le système d'information depuis l'extérieur par le partenaire, les salariés d'Artal Technologies, mais aussi les potentiels futurs clients.

2. Mise en œuvre

2.1. Conception globale

Les besoins présentent un large périmètre d'action que j'ai dû organiser afin d'avoir une vue d'ensemble sur les solutions à mettre en place pour ce système d'information.

2.1.1. Organiser les besoins

Comme présenté précédemment, mon travail s'organisait autour des axes suivants :

- ❖ Respecter les contraintes imposées par le partenaire
- ❖ Optimiser la gestion de projet
- ❖ Automatiser la génération de la documentation
- ❖ Automatiser la génération du produit
- ❖ Evaluer la qualité du code
- ❖ Gérer la documentation
- ❖ Centraliser le système d'information

J'ai alors mené une réflexion autour de ces besoins afin de les regrouper autour d'aspects techniques communs qui me permettraient ensuite de trouver plus facilement des solutions techniques adaptées :

- ❖ Le **respect des contraintes** imposées par CMC Electronique doit passer par la préservation maximum de la confidentialité de ce projet. Les migrations des deux outils déjà présents à mon arrivée, [Jira](#) et [Git](#), devaient être planifiées.
- ❖ La solution sur laquelle se base la **gestion de projet** de la société est [Jira](#). Pour l'améliorer, un travail doit être fait sur cet outil.
- ❖ La **génération automatique de la documentation** pour répondre à la norme DO-330 fut un sujet moins précis. Dans un premier temps, je l'ai placé à l'extérieur du système d'information à mettre en place. Au fur et à mesure de l'étude, nous avons décidé que cette tâche s'organiserait principalement autour de [Jira](#), raison pour laquelle j'ai finalement décidé de la présenter dans la mise en œuvre du gestionnaire de demandes.

- ❖ L'automatisation, qui eut lieu dans un second temps, fut précédé par la prise en main de l'**environnement de développement** composé en partie de **Git** et de l'**IDE Eclipse**, à partir duquel les développeurs de Solutions Isonéo généraient manuellement le produit. Cet apprentissage permit d'extraire d'Eclipse le procédé de **génération du produit**.
- ❖ **L'automatisation de cette génération** n'est pas essentielle à l'évolution du produit et ne fait ainsi pas partie de l'environnement de développement. Elle appartient aux **solutions qui aident ce développement** tout comme le **contrôle de la qualité du code**.
- ❖ La **documentation** est obligatoire à la bonne capitalisation des compétences acquises sur tous les composants du système d'information et à l'**accompagnement des utilisateurs**. La **diffusion de ces informations** possède une place importante dans le contexte particulier de Solutions Isonéo qui est une « start-up » ayant un lien fort avec sa jumelle française.
- ❖ Pour finir, la **centralisation du système d'information** englobe tous les autres besoins : en effet tous les outils répondant aux attentes seront installés sur un **serveur** physiquement présent dans les locaux de Solutions Isonéo.

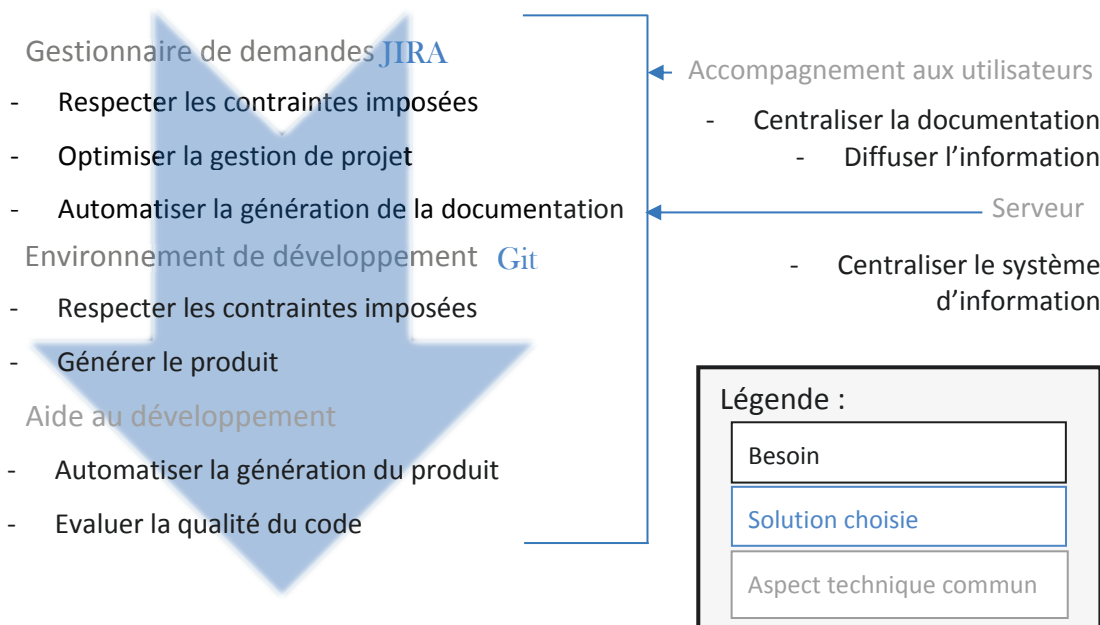


Figure 11 : Organisation des besoins selon des aspects techniques

La mise en œuvre s'est organisée autour de ces aspects techniques qui constituent les parties suivantes de ce mémoire.

2.2. Gestionnaire de demandes

2.2.1. Optimiser la gestion du travail

Avant de concevoir et de modéliser des solutions à mettre en place sur le gestionnaire de demandes pour répondre aux besoins, je me suis positionnée comme utilisatrice de l'outil afin de monter correctement en compétence sur l'existant.

Jira n'était pas un outil inconnu car je l'avais déjà administré dans un précédent emploi. Cependant les plug-ins utilisés pour le suivi du projet selon la méthodologie agile Scrum étaient nouveaux pour moi.

a) Montée en compétences sur les plug-ins Jira

Pour aider à l'utilisation de méthodes Agiles, Jira prévoit des plug-ins comme « **Greenhopper** » qui permet d'organiser chaque tâche du « Sprint » selon la hiérarchie suivante.

- ❖ A faire
- ❖ En cours
- ❖ Fait

Cela permet de pouvoir associer une charge à chaque tâche et de voir rapidement l'ensemble du planning, les avances et les retards.

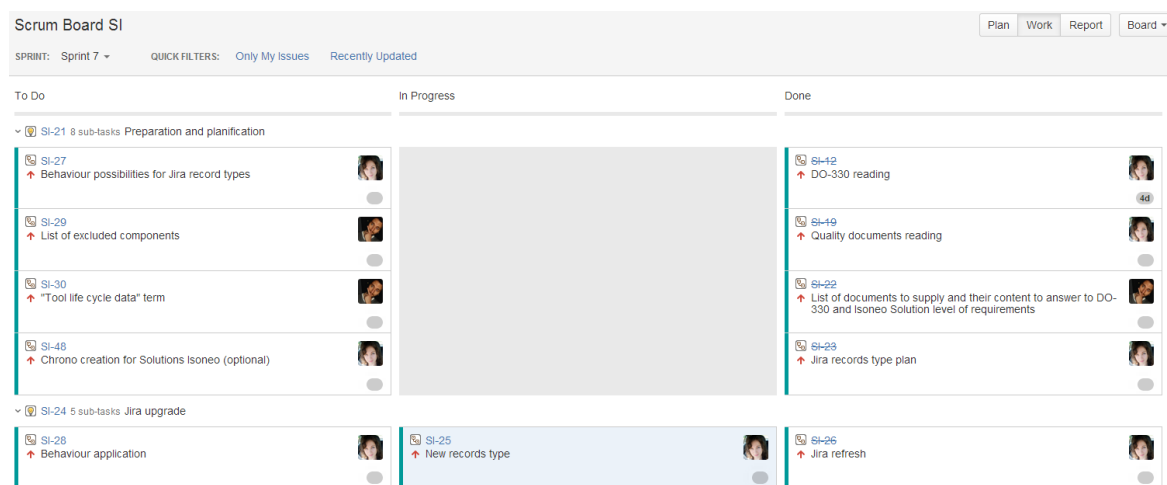


Figure 12 : Plug-in « Greenhopper » de Jira OnDemand

Ce plug-in met à disposition un « tableau de bord » qui fonctionne avec certains types d'enregistrement que l'utilisateur doit respecter pour exploiter convenablement le plug-in.

A la visualisation de ce « tableau de bord » j'ai pu en extraire certaines règles de fonctionnement de l'existant.

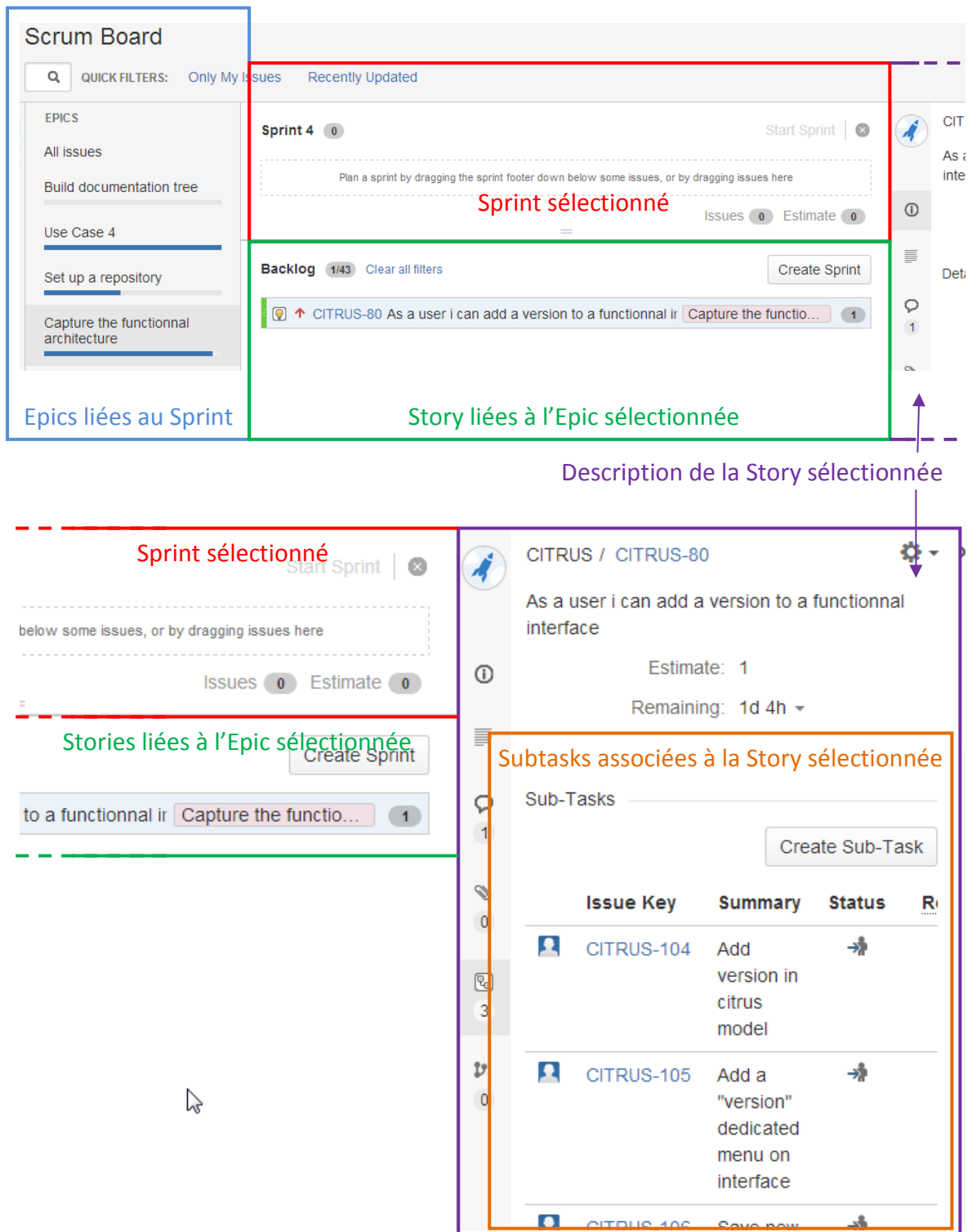
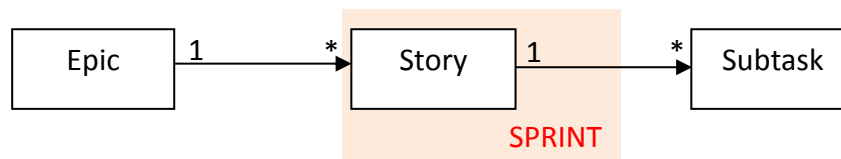


Figure 13 : Schéma explicatif du tableau de bord du Sprint 4 du projet Citrus



L'imbrication des types d'enregistrement se fait donc de la façon suivante :

Figure 14 : Liens entre les types d'enregistrement existants et nécessaires au plugin « Greenhopper »

Une « Epic » correspond aux besoins opérationnels de l'outil à développer. Ils décrivent ainsi les raisons d'être du système à créer.

Une « Story » aussi appelée « User Story » décrit les besoins fonctionnels, c'est-à-dire les différentes fonctionnalités à mettre en place pour répondre à un des besoins opérationnels. Ceux-ci sont généralement décrits dans une documentation, mise en place lors de la phase de conception, qui regroupe les spécifications fonctionnelles attendues par le client.

Les « Subtasks » correspondent aux besoins techniques et décrivent ainsi tous les éléments à mettre en place pour développer une fonctionnalité de l'outil.

Le « Sprint », élément central de la méthodologie Scrum, est associé à la « Story » car une « Epic » est dans la majorité des cas livrée après plusieurs Sprints. Lors du « Sprint planning » il est habituel de définir une liste de « Story » à développer et à livrer lors du « Sprint review ».

Un autre plugin Jira adapté au suivi de l'activité sur le projet était disponible : le plugin « [JIRA TimeSheet Plugin](#) » mais n'était pas utilisé par Erwan Deschamps en charge des rapports de fin de « Sprint », justifiant du travail de l'équipe par rapport à l'avancement planifié. J'ai donc documenté ce plugin et formé rapidement l'équipe à l'utilisation de celui-ci qui est aujourd'hui toujours utilisé pour la gestion de projet et la gestion personnelle du suivi.

Le travail chez Solutions Isonéo est organisé suivant la méthode Scrum associant à chaque tâche un « temps passé » qui peut s'accumuler si la personne en charge met en attente la demande pour reprendre ensuite le travail.

L'équipe de développement travaillant avec un partenaire il est important de lui communiquer l'efficacité durant le « Sprint » passé. Ainsi Solutions Isonéo a cherché à

fournir une vue d'ensemble sur la répartition de la charge de travail par rapport à la totalité du « Sprint » en fonction des membres et des fonctionnalités à développer.

Pour cela j'ai activé la fonctionnalité « Pivot » du plugin « Jira TimeSheet Plugin » sous Jira qui permet en temps réel de suivre l'activité sur le « tableau de bord » disponible sur la première page lors de la connexion à Jira et fournit une extraction au format excel facilement transportable (transmission par mail par exemple).

Sur l'impression écran suivante, nous pouvons voir en ordonnée les tâches actives du projet « Citrus », en abscisse les personnes travaillant sur chacune d'entre elles, avec à la jointure le nombre d'heures passées. De cette façon, il est facilement repérable si plusieurs personnes ont travaillé sur la même tâche et combien d'heures totales la tâche comptabilise.

Project Pivot					
Displaying for CITRUS and group developers citrus (Details)			evelyne fatela-nabais	Jonathan Lasalle	Julien SIEGA Tot
📌	CITRUS-15	As a user i can create a partition in IMA project	↑ 0h	16h	0h 16h
📌	CITRUS-160	As a user i can edit a CMC ICD using CITRUS	↑ 0h	0h	4h 4h
📌	CITRUS-220	Asset Manager : Read RAS documentation	↑ 16h	0h	0h 16h
📌	CITRUS-225	As a user I can define a logical architecture for platform	↑ 0h	0h	20h 20h
Total:			16h	16h	24h 56h

Figure 15 : Vue du plug-in « Pivot » via le tableau de bord Jira

L'extraction au format excel permet en plus de voir en détail le temps passé sur chaque tâche jour après jour.

Project	Type	Key	Title	Date	Username	Time Spent (h)	Comment
CITRUS	Story	CITRUS-225	As a user I can define a logical architecture for platform	17-juil-13	Julien SIEGA	8	
CITRUS	Story	CITRUS-225	As a user I can define a logical architecture for platform	16-juil-13	Julien SIEGA	8	
CITRUS	Story	CITRUS-225	As a user I can define a logical architecture for platform	15-juil-13	Julien SIEGA	4	
CITRUS	Task	CITRUS-220	Asset Manager : Read RAS documentation	15-juil-13	evelyne fatela-nabais	8	
CITRUS	Task	CITRUS-220	Asset Manager : Read RAS documentation	16-juil-13	evelyne fatela-nabais	8	
CITRUS	Story	CITRUS-160	As a user i can edit a CMC ICD using CITRUS	15-juil-13	Julien SIEGA	4	
CITRUS	Story	CITRUS-15	As a user i can create a partition in IMA project	15-juil-13	Jonathan Lasalle	16	

Figure 16 : Extraction au format excel fournit par le plug-in « Pivot »

b) Uniformisation des processus et organisation du travail

Deux projets existent sous Jira : « Citrus » pour le logiciel à développer en partenariat avec CMC Electronique et « SI », diminutif de Système d'Information, pour enregistrer toutes les tâches à réaliser pour la mise en place du système d'information de la « start-up » québécoise sur lequel j'ai enregistré mes tâches.

Ces deux projets sont informatiques mais ne font pas partie du même « métier » : l'un est créé pour de la programmation et l'autre pour de la gestion de configuration.

Cependant les deux projets possèdent des utilisateurs, clients et/ou partenaires. CMC Electronique est le partenaire qui utilise Citrus comme une application pour en déceler les problèmes, les développeurs de Solutions Isonéo sont les utilisateurs du système d'information. Le projet « SI » est donc un projet interne qui possède les mêmes composantes que le projet « Citrus ». De ce fait nous avons décidé de gérer les deux projets de la même façon, suivant la méthodologie Scrum.

Mon travail a donc été organisé selon des « Sprint » durant environ un mois et terminant 2 jours après le « Sprint review » du projet Citrus. Des « Epic », « Story » et « Subtask » ont été utilisées.

Tous les membres de la société ont participé au « Daily Sprint », tous les matins vers 10h.

Cette tâche d'uniformisation fût importante afin que les développeurs comprennent mon travail et qu'un esprit de cohésion se crée dans cette nouvelle société. Ainsi les problématiques rencontrées par certains pouvaient être comprises par les autres et trouver des solutions grâce à un point de vue extérieur.

Le suivi de mes tâches sous Jira m'ont permis d'avoir une planification claire et rendait possible le suivi de l'avancement de mon stage par mon responsable direct Erwan Deschamps mais aussi par mon responsable / client, Julien Siega. Les priorités changeant souvent, l'organisation du travail fût une étape obligatoire dans ce stage.

c) Optimisation des processus et capitalisation

Jira propose nativement des types d'enregistrements correspondant à des besoins différents comme les « Epic », « Story » et « Subtask ».

Le but étant, selon la méthodologie Scrum, de pouvoir expliquer les écarts entre les fonctionnalités à développer durant le « Sprint » et les fonctionnalités effectivement livrées, il est essentiel d'enregistrer dans Jira tous les types d'actions menées durant le « Sprint ». Solutions Isonéo a donc eu besoin de **nouveaux types d'enregistrement** sur les projets Citrus et SI :

- ❖ Meeting pour tracer les réunions

- ❖ Management pour tracer les actions propre à la gestion du projet comme la mise en place sous Jira du démarrage du Sprint, de l'association des « Story » à ce « Sprint » et de l'évaluation de la charge de chacune d'elles.

La mise en place de ces deux nouveaux types d'enregistrement a été accélérée par mes compétences existantes sur l'administration Jira. Cependant la démarche a été capitalisée dans la documentation de Solutions Isonéo.

Dans un premier temps, j'ai travaillé sur les champs nécessaires à ces deux nouveaux types. Jira possède nativement des champs utiles à tous les types :

- ❖ Project : projet concerné par l'enregistrement ; pour nous « Citrus » ou « SI »
- ❖ Reporter : personne qui a créé l'enregistrement
- ❖ Assignee : personne en charge de sa résolution
- ❖ Summary : titre de l'enregistrement
- ❖ Description : contenu de l'enregistrement
- ❖ Epic Link : lien vers l'Epic correspondante

Aucun champ supplémentaire est nécessaire au type « Management ». Cependant le type « Meeting » devait se composer d'autres champs, pertinents et nécessaires que j'ai listés pour les proposer à Julien Siega, principal utilisateur de ces deux nouveaux types d'enregistrement :

- ❖ Titre de la réunion
- ❖ Liste des participants
- ❖ Lieu
- ❖ Heure
- ❖ Organisateur
- ❖ Points abordés prévus
- ❖ Points abordés effectifs

Dans un second temps j'ai exploré les éléments dans l'administration de Jira afin de mettre en place les deux nouveaux types et leurs champs. Pour cela j'ai établi un

vocabulaire commun entre les membres de l'équipe. Toute la documentation au sein de Solutions Isonéo est rédigée en anglais, de même que les termes concernant les outils:

- ❖ **Issue Type** comme son nom l'indique désigne le type des demandes de Jira comme « Epic », « Subtask », etc.
- ❖ **Workflow** est le cycle de vie de la demande. Il est composé d'états dans lesquels se trouve la demande (pris en compte, en cours, résolu, clos) et de transitions par lesquelles elle transite.
- ❖ **Project** est l'entité de Jira qui va associer certains types d'enregistrements (issue type), certains "écrans" (screens) et certains cycles de vie (workflows). Il définit donc un projet et une méthode de travail lui est associée. De cette façon nous aurions pu avoir un projet « Citrus » travaillant avec des « Epic », « Story » et « Subtask » et un projet « SI » ne suivant pas la méthode Scrum utilisant simplement des « Tasks ».
- ❖ **Screen** décrit la composition des pages permettant ainsi aux utilisateurs de saisir des informations supplémentaires. Ils sont composés de champs.
- ❖ **Custom field** désigne tous les champs qui ne sont pas natifs dans Jira et que j'aurais pu créer pour les besoins de Solutions Isonéo.

A ces éléments facilement identifiables par les utilisateurs s'ajoutent des éléments accessibles dans l'administration de Jira.

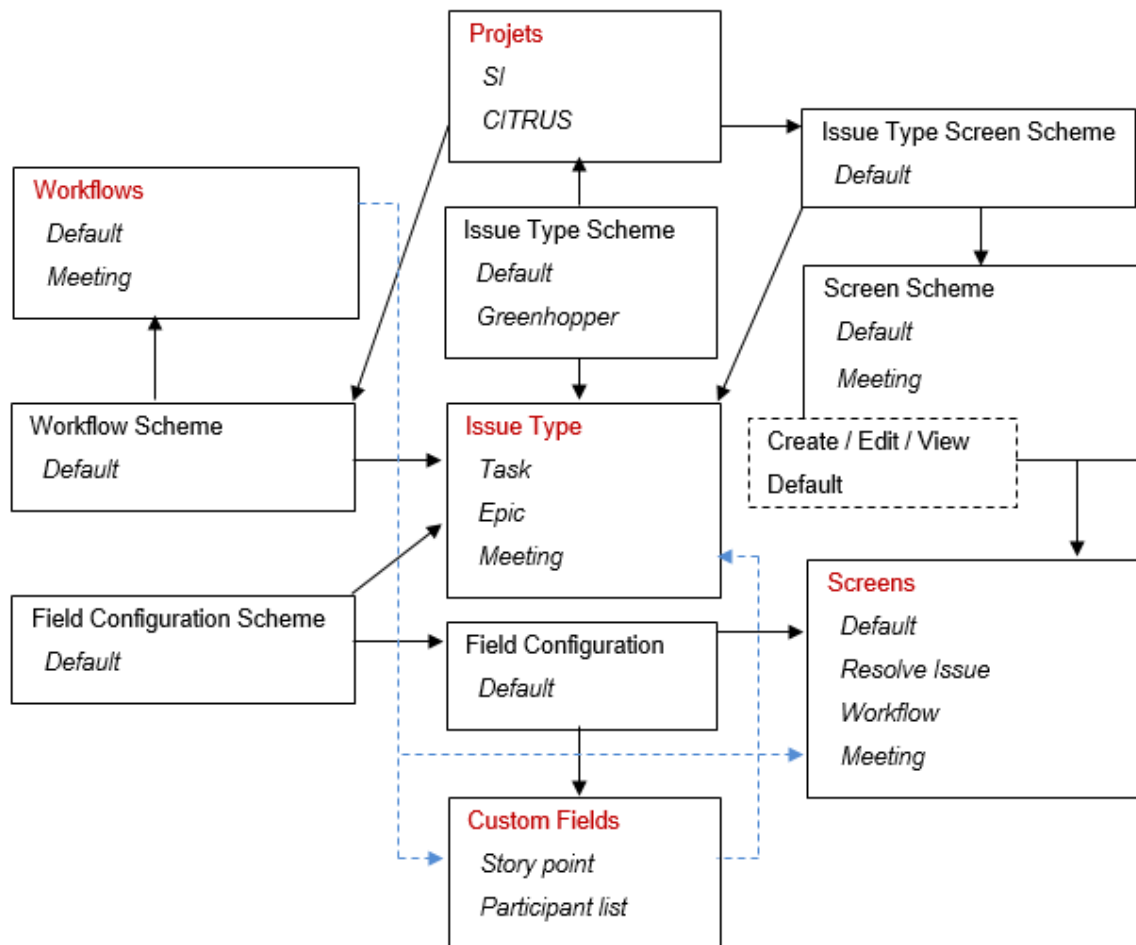


Figure 17 : *Organisations des éléments dans Jira*

Pour maîtriser l'ajout de certains champs ou l'association entre de nouveaux projets et des types d'enregistrements il est nécessaire de maîtriser l'imbrication de ces éléments.

- ❖ **Issue Type Scheme** lie un projet avec un ou plusieurs types d'enregistrement (issue type). Nativement dans Jira, deux « Issue Type Scheme » sont présents: Default and Greenhopper. Cependant, il est possible que Solutions Isonéo ai besoin d'en créer pour associer à un projet un nouveau groupe de type d'enregistrement.
- ❖ **Workflow Scheme** permet d'associer un cycle de vie (workflow) à un ou plusieurs types d'enregistrement (issue type). Le projet pointe ensuite vers le « Workflow Scheme » adapté. Le « Workflow Scheme » permet d'avoir un cycle de vie différent pour un même type d'enregistrement (issue type). Des projets peuvent avoir le même besoin (association définie par l'Issue Type Scheme) mais que celui-ci ai une place plus ou moins importante dans le projet (association définie

par le Workflow Scheme). De ce fait le type d'enregistrement « Epic » peut avoir un cycle de vie très simple pour un projet et un très complet pour un autre. Le projet « Citrus » peut être associé via « l'Issue Type Scheme » au type « Epic » et utiliser son cycle de vie simplifié via le « Workflow Scheme » alors qu'un autre projet « SI » peut être associé au même « Issue Type Scheme » mais utiliser le workflow complet via un « Workflow Scheme » différent.

- ❖ **Screen Scheme** associe une « page web », composée de champs sélectionnés en fonction des besoins, à une opération sur l'enregistrement telle que la création, l'édition ou la consultation.
- ❖ **Issue Type Screen Scheme** permet de lier un « Screen Scheme » avec les types d'enregistrement et indirectement les « pages web » aux types d'enregistrement.
- ❖ **Field configuration** est requis pour spécifier qu'un champ est obligatoire en tout temps. Il est possible d'ajouter des champs obligatoires sur certaines transitions uniquement, mais cette configuration est possible grâce à une « extension » de Jira appelé «Jira Suite Utilities».
- ❖ **Field Configuration Scheme** définit la configuration d'un champ par rapport à un type d'enregistrement (issue type), raison pour laquelle la définition d'un champ obligatoire sera appliquée tout au long du cycle de vie.

Pour finir, j'ai implémenté les nouveaux champs dans le paramétrage « Custom fields » puis les nouveaux types dans « Issue type ». Afin que la modification ne soit pas difficile à faire dans le futur, j'ai créé un cycle de vie (Workflow) propre à chaque type d'enregistrement. Ainsi il est facile d'ajouter un niveau d'attente ou d'ajouter un écran avec des champs supplémentaires. Le « Workflow » est cependant lié à un seul type d'enregistrement via le « Workflow scheme ». Si nous avions voulu optimiser la gestion des cycles de vie, nous aurions pu prévoir un cycle de vie ayant seulement les deux transitions « Start » (démarrer) et « Stop » (terminer) pour les deux nouveaux types et garder le cycle de vie avec une option d'arrêt du travail sans clôture pour les « Epic », « Story » et « Subtask » qui sont des enregistrements pour des actions à plus long terme : la programmation.

Le « Field configuration » a permis de composer l'écran « Meeting » avec les champs supplémentaires créés précédemment. Le type « Management » n'ayant pas besoin de champs autre que ceux utilisés dans les « Epic », « Story » et « Subtask » les mêmes écrans ont été réutilisés afin que les modifications impactent tous les types d'enregistrement. La même réflexion a été appliquée au « Screen scheme ».

Le « Field configuration scheme » a ensuite permis d'associer l'écran précédemment créé au type « Meeting » et de lier le « Default » au type « Management ».

L'« Issue type scheme » par défaut qui est utilisé sur tous les projets a été enrichi des deux nouveaux types d'enregistrement.

Les projets « Citrus » et « SI » possèdent maintenant les types d'enregistrement « Meeting » et « Management ».

Ces actions ont été facilement réalisables grâce à l'interface graphique de Jira accessible via un navigateur web :

Name	Options	Projects
Default Issue Type Scheme Default issue type scheme is the list of global issue types. All newly created issue types will automatically be added to this scheme.	<input checked="" type="checkbox"/> Bug (Default) <input checked="" type="checkbox"/> New Feature <input checked="" type="checkbox"/> Task <input checked="" type="checkbox"/> Improvement <input checked="" type="checkbox"/> Sub-task <input checked="" type="checkbox"/> Epic <input checked="" type="checkbox"/> Story <input checked="" type="checkbox"/> Technical task <input checked="" type="checkbox"/> Meeting <input checked="" type="checkbox"/> Management	Global (all unconfigured projects)

Figure 18 : Interface graphique d'administration des « issue type scheme » de Jira

Solutions Isonéo utilisant un gestionnaire de sources et ayant déjà connecté SVN et un gestionnaire de demandes dans mes emplois précédents, j'ai cherché à [connecter Jira et Git](#). La connexion permet simplement de lier un enregistrement dans Jira à un dépôt de source (commit) dans le gestionnaire de sources. Il suffit aux programmeurs de saisir dans la description du « commit » la référence de la demande Jira, par exemple CITRUS-25. Jira affiche ensuite sur l'enregistrement, dans l'onglet « Commits », toutes les actions effectuées dans Git.

2. Mise en œuvre

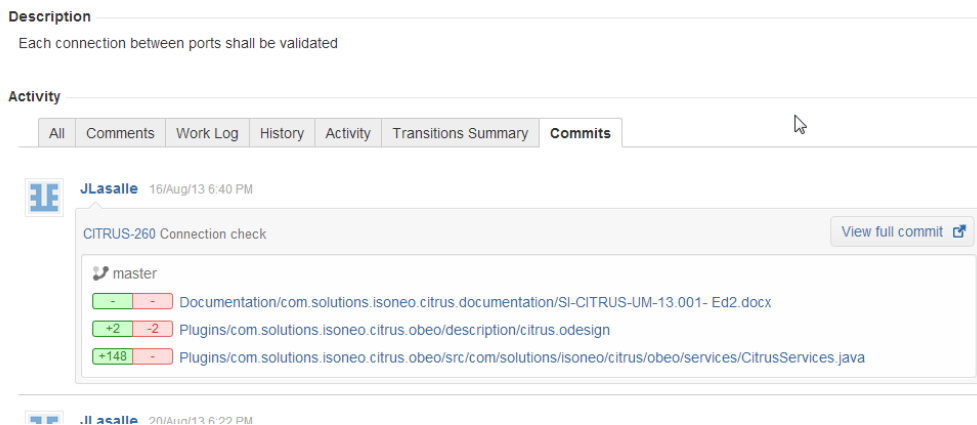


Figure 19 : Onglet « Commits » d'un enregistrement Jira

Dans la console d'administration de Jira, un plugin que j'ai ajouté à la plateforme, permet de connecter un compte Github au gestionnaire de demandes à partir d'un nouveau menu nommé « Source control ».

Pour ce besoin, j'ai demandé à l'administrateur Github, Julien Siega, de créer un compte ayant des accès uniquement en lecture sur tous les dépôts potentiellement en lien avec du travail tracé dans Jira. Le compte solutionIsoneo a été créé.

Manage DVCS Accounts

Connect your Bitbucket and GitHub accounts to JIRA OnDemand and link every commit with a bug or development task. Once configured, JIRA will query the repository searching commits for issue keys.

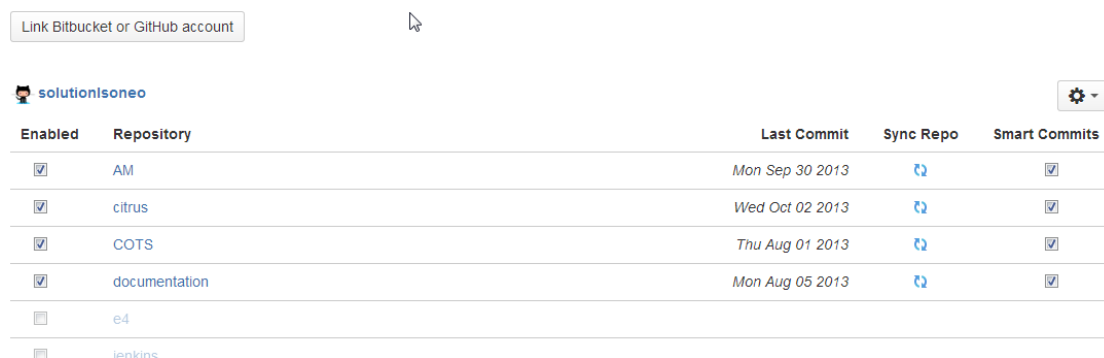


Figure 20 : Mise en place du lien entre Jira et Git exploité via Github

Jira est « à l'écoute » des mouvements sur les dépôts Git listés. Dès que le commentaire d'un « commit » sur ces dépôts contient une « clé » d'un enregistrement Jira, son onglet « Commits » sera peuplé.

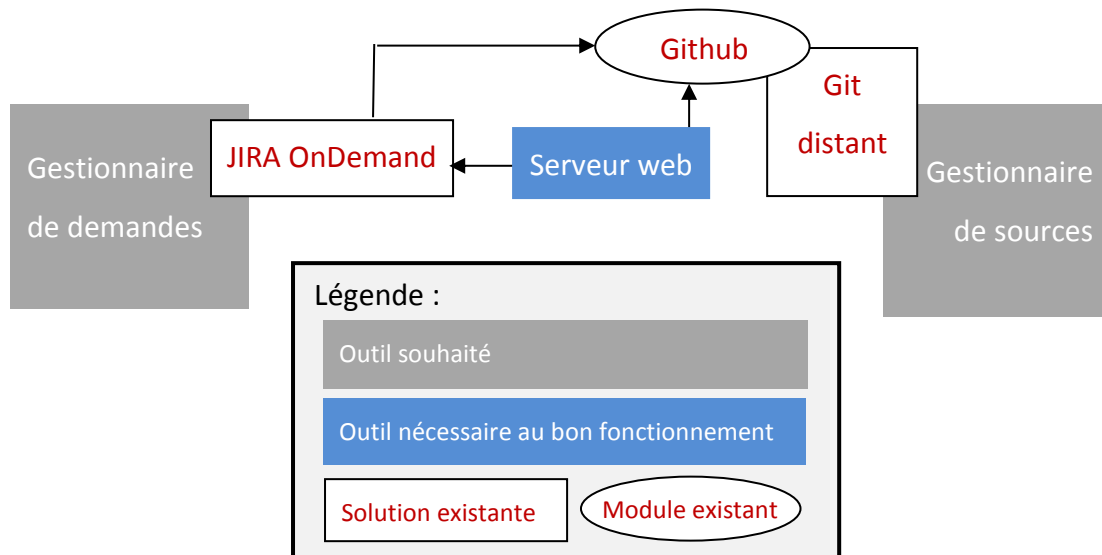


Figure 21 : Composants du S.I. en construction : mise en place du lien entre « Jira OnDemand » et « Git distant » via « Github ».

2.2.2. Appliquer les contraintes imposées : migrer Jira sur le serveur de Solution Isonéo

Le projet Citrus étant un projet gouvernemental comportant des documents confidentiels, CMC Electronique a imposé que toutes les données en lien avec ce projet soient contrôlées au maximum. De ce fait, j'ai été en charge d'étudier la migration de Jira OnDemand hébergé sur un serveur d'Atlassian vers « Jira hosted » hébergé sur le serveur de Solutions Isonéo.

La migration n'est pas difficile à réaliser car l'outil propose un export et import de la base. La complexité de cette tâche résidait dans la vérification de l'intégrité des éléments migrés et dans le recensement des éléments à ajouter pour les besoins de Solutions Isonéo. En effet la version Jira OnDemand est toujours à jour sur les dernières évolutions de l'outil alors que la version « hosted » est une version livrée qui ne contient donc pas les toutes dernières fonctionnalités. De plus la version « OnDemand » propose une utilisation « vitrine » de l'outil et certains plugins y sont juste en version d'essai.

J'ai établi une matrice pour vérifier l'intégralité des éléments afin d'être sûr de ne pas perdre du travail déjà capitalisé dans Jira :

- ❖ Fonctionnalités
- ❖ Données

❖ Configuration

a) Utilité et coûts des fonctionnalités

Deux plugins Jira sont obligatoires au travail de l'équipe de développement de Solutions Isonéo :

- ❖ « Greenhopper » qui permet de gérer les tâches suivant les « Sprint » et de les démarrer via une action de « glisser, déposer ».

Cet élément n'est pas migré car il est payant. Cependant le prix étant de 10\$ par an pour moins de 10 utilisateurs, il répond aux contraintes budgétaires de Solutions Isonéo.

- ❖ « JIRA TimeSheet Plugin » qui permet de suivre l'activité des membres de l'équipe suivant la méthodologie Scrum. Ce plugin n'est pas migré mais peut être ajouté.

Certains champs sont associés à ces plug-ins, il est donc important d'envisager leur mise en place avant la migration afin de réduire les risques de perte d'information ou d'incapacité de migrations faute de champs pour réceptionner les informations.

b) Récupération des données

Pour vérifier que toutes les données ont été migrées, pour chacun des types d'enregistrements et pour l'ensemble des projets, j'ai tout simplement vérifié que le nombre d'enregistrements trouvés dans Jira correspondait.

J'ai également vérifié que les utilisateurs avaient été migrés ainsi que leurs groupes.

c) Conservation de la configuration additionnelle

Toute la configuration Jira est dorénavant migrée : « Issue type », « Workflows », « Workflows Scheme », « Screen », « Screen Scheme », « Issue Type Screen Scheme », « Custom fields », « Field configuration », « Issue Statuses », « Issue resolutions », « Issue priorities », « Issue Security Scheme », « Notification Scheme » et « Permission Scheme ».

Tous ses éléments ont été consultés dans l'administration Jira et contrôlés grâce à mes compétences existantes sur les outils et à ma connaissance sur la totalité de ces éléments.

d) Choix d'une base de données

Jira embarque par défaut une base de données HSQLDB. Cette base de données n'est pas faite pour une utilisation à long terme mais est fournie dans l'installation à des fins de démonstration.

Nous avons décidé d'utiliser MySQL, système de gestion de base de données utilisée dans le monde entier dans un cadre professionnel et connue de Julien Siega.

Le premier composant du système d'information à construire est donc mis en place et nous savons également paramétré le lien entre ce composant et Git via l'utilisation de Github. Les solutions « Jira OnDemand » et « Git distant » possèdent des interfaces web cependant ne les ayant pas moi-même installés et afin d'alléger le schéma suivant, j'ai préféré ne pas représenter les flèches depuis le « Serveur web ». « Jira hosted » englobe dans son installation un « Apache Tomcat » installé automatiquement lors de la mise en place de la solution : ce lien est donc représenté.

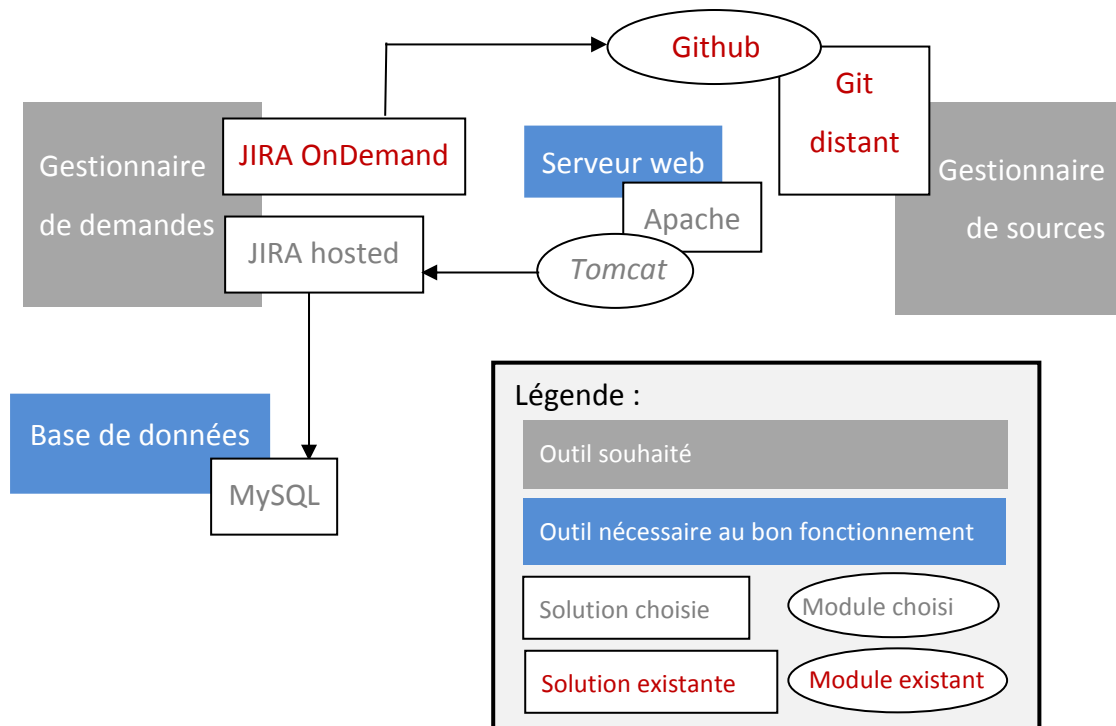


Figure 22 : Composants du S.I. en construction : mise en place de « Jira hosted »

2.2.3. Générer automatiquement la documentation

Les projets Airbus d'Artal Technologies répondent à la norme IEEE. L'objectif était que le projet « Citrus » réponde également à une norme : la DO-330. Pour cela, plusieurs documents devaient être produits. La mise en place des éléments nécessaires à l'obtention d'une norme demande énormément de temps. En effet les documents à produire sont nombreux, précis et doivent être tenus à jour.

Pour répondre à la norme aéronautique DO-330, Solution Isonéo a souhaité générer une partie de ces documents de la façon la plus automatisée possible à partir des différents éléments composant le système d'information.

Dans un premier temps un document excel m'a été fourni sur lequel tous les documents de la DO-330 pertinents pour la société Solutions Isonéo avaient été répertoriés. Ce document est disponible dans la première annexe.

Après lecture de la DO-330, j'ai éclairci chacun des besoins auxquels devait répondre Solutions Isonéo en mettant en place un fichier excel ayant un onglet par document à produire. A l'intérieur de ceux-ci j'ai pu ajouter chacun des chapitres attendus avec la

définition des termes importants à l'identification des informations. Un aperçu de ce document est accessible en annexe 2.

Le travail s'est ensuite organisé pour chaque document en 3 temps :

- ❖ Quels besoins le composent
- ❖ Où trouver les informations pour répondre aux besoins
- ❖ Comment récupérer ces informations

Par exemple pour le document « Tool Requirements », l'objectif est de fournir un état des fonctionnalités de l'outil et de leur décomposition.

How	
Création d'Epic dans Jira + liaison aux User Stories	
Voir les solutions de transformation des "How To" du Help Content (xml html) Eclipse vers le word ou inversement	
A Ajouter dans le Help Content : "Configuration" + avertissement vers la config dans le "HowTo" en relation	
User Story et Subtask d'Epic	
Nouveau type d'enregistrement : Specific Requirement	
Nouveau type d'enregistrement : Specific Requirement	
Nouveau type d'enregistrement : Soft Behaviour (Bon / Pas bon)	
Sur User Story :	
Ajout de composants : SE Studio / Asset Manager	
Keyword : "Interface" pour être obligé de la lier à User Story de l'autre collection	

Needs	Where
Description of the tool functions and technical features, including modes of operation	Excel existant (EPIC)
User manual (which can be part of Tool Requirements) with user instructions, installation instructions, list of error messages, and constraints.	Manuel mais génération du word automatique
Requirements to describe the users' ability to customize the tool	Manuel mais génération du word automatique
Functional requirements with the appropriate level of detail (see 5.2.1.2.k)	Jira
Specific requirements (if necessary, for compliance with tool operational environment)	Jira
Specific requirements : failure modes and response to inconsistent inputs	Jira
Expected responses of the tool under abnormal operating conditions	Jira
Interface requirements between the tools in case of collection of tools	Manuel / Considération de 2 collections d'outil : Asset manager et SE Studio

Figure 23 : Etude des besoins du « Tool Requirements » de la DO-330

D'un point de vue global, ce travail aurait pu occuper la totalité du temps de ce stage car la récupération de certaines informations demande la mise en place de plug-ins sur les outils pouvant composer le système d'information. Ce dernier étant quasi inexistant à mon arrivée, il était prioritaire - et de toute façon obligatoire - de mettre en place toutes les composantes du système d'information avant de pouvoir en tirer des informations via des plug-ins.

Cependant, possédant l'expertise sur l'administration de Jira, j'ai étudié la totalité des besoins auxquels doit répondre Solutions Isonéo pour obtenir la certification tout en me concentrant sur les réponses que pourrait apporter Jira.

Après plusieurs itérations, j'ai organisé plusieurs réunions de 30 minutes avec les programmeurs. Dans un premier temps nous avons éclairci ensemble les termes présents dans la DO-330 ; la norme étant très précise nous devons établir ensemble un langage commun ainsi que clarifier le degré de précision de chaque terme. Par exemple, établir la différence entre « tool functions », « functional requirements », « operational requirements » et « low-level tool requirements ». Dans un second temps, l'objectif de ces réunions fût de définir un degré de précision pertinent pour chacune des informations. En effet, certaines informations pouvaient être redondantes dans le cas du projet « Citrus » ou au contraire mettre en évidence le besoin de créer un nouvel enregistrement dans Jira.

Pour finir, j'ai fourni un état des lieux des enregistrements existants dans Jira ainsi qu'une liste des nouveaux besoins afin de s'entendre sur les éléments à supprimer et à créer pour obtenir un projet avec des enregistrements pertinents et une couverture complète des besoins de l'équipe.

Additionnellement, pour gérer le degré de précision sans multiplier inutilement le nombre d'enregistrements, j'ai proposé de mettre en place des informations supplémentaires sur les enregistrements existants, apparaissant en vert sur le schéma suivant.

Pour les besoins de la DO-330, Solutions Isonéo a besoin de pouvoir manipuler dans Jira des éléments existants à enrichir :

- ❖ Epic : besoin opérationnel de l'outil à développer
- ❖ Story : besoin fonctionnel de l'outil à développer pour une fonctionnalité destinée aux utilisateurs, associé à un des deux ensembles principaux de l'outil : SE Studio ou Asset Manager. Il peut également être qualifié :
 - d'« Interface » : besoin permettant d'interfacer les deux ensembles SE Studio et Asset Manager. Dans l'idéal, il faudrait que Jira oblige la création d'un lien entre les « Story » des deux entités. Par exemple, sur une

- « Story » associée à « SE Studio », dès que celle-ci est définie comme « Interface », Jira doit proposer de la lier à une « Story » existante liée à « Asset Manager ».
- de « Specific Requirement » : besoin spécifique comme des adaptations pour que le logiciel fonctionne dans un certain environnement opérationnel (système d'exploitation différent, version de compilateur différent, etc.). Comme montré dans l'impression écran précédente, « Specific Requirement » devait initialement donner naissance à un nouveau type d'enregistrement mais Solutions Isonéo a préféré limiter le type d'enregistrement et multiplier ses qualifications. De cette façon un besoin fonctionnel peut-être un besoin spécifique pour une fonctionnalité destinée aux utilisateurs et être associé à l'ensemble « Asset Manager » du projet « Citrus ».
 - de « Other » : besoin qui n'est ni associé à une interface, ni à un besoin spécifique.
- ❖ Subtask : besoin technique de l'outil à développer. Il est désormais possible de qualifier une « Subtask » en tant que « Design » si le besoin technique est aussi un besoin spécifique concernant l'architecture du produit.
 - ❖ Meeting : tâche pour tracer le temps passé en réunion
 - ❖ Management : tâche pour renseigner le temps passé dans des tâches de gestion de projet

Solutions Isonéo a également besoin d'enregistrements existants nativement dans Jira qui n'étaient pas utilisés :

- ❖ Improvement : évolution impactant la totalité de l'application comme la mise en place de « raccourcis clavier »
- ❖ Documentation task : tâche pour la création ou la mise à jour d'un document

Pour finir, de nouveaux types d'enregistrement ont été créés :

- ❖ Soft Behaviour : comportement spécifique du logiciel testé.
- ❖ Technical task : besoin pour des fonctionnalités techniques de bas niveau alors que les « Stories » correspondront aux fonctionnalités offertes aux utilisateurs du

logiciel. Une « Technical task » peut être qualifiée de la même façon qu'une « Story ».

- ❖ Report : problème rencontré sur le logiciel en cours de développement. Il peut être soit un « bug » s'il est détecté par un utilisateur ou un « problem » s'il est détecté par la compilation automatique via l'intégration continue présentée dans la suite du document. Ce type d'enregistrement doit être lié à la « Story » ou à la « Technical task » durant laquelle le problème a été introduit.

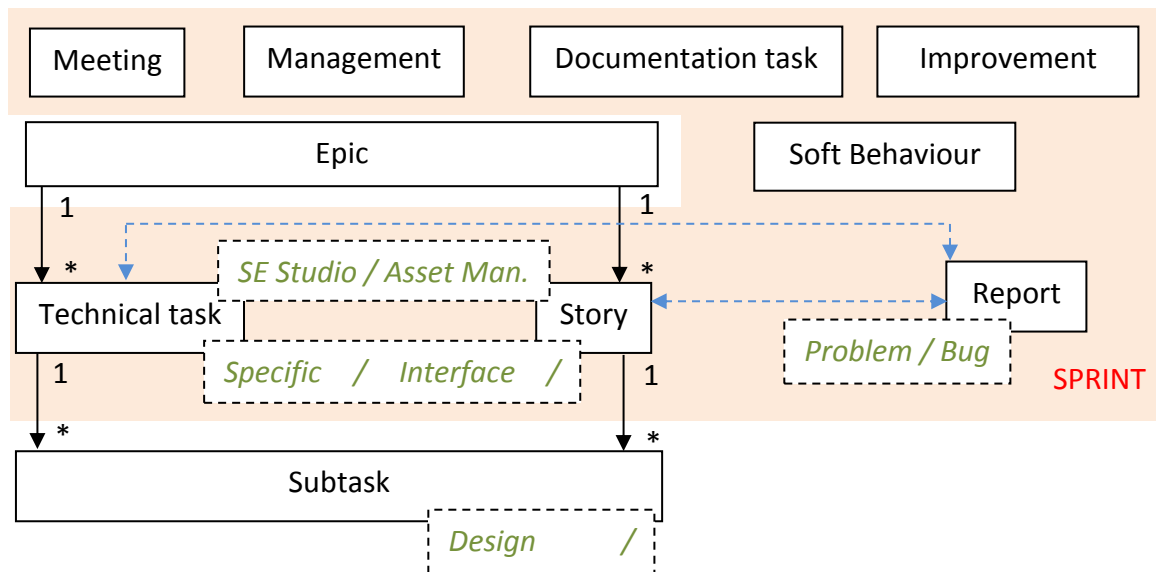


Figure 24 : Éléments à mettre en place dans Jira pour répondre à une partie des besoins pour la DO-330

Le type « Epic » n'est pas lié au « Sprint » car celui-ci correspond à une fonctionnalité globale de l'outil à développer, sa réalisation s'étale donc sur plusieurs « Sprint ». De même les « Subtasks » ne sont pas directement liés au « Sprint » mais le sont grâce à leur « enregistrement parent », dans le cas contraire, l'information serait redondante.

Il est important de noter qu'il est possible de rechercher des ensembles d'enregistrements via la recherche de Jira. Il est ainsi possible de visionner l'ensemble des besoins associés à la partie « SE Studio » du projet « Citrus » ou même de voir l'intégralité des problèmes de compilation rencontrés par l'intégration continue.

Les types d'enregistrement à supprimer sont donc des éléments natifs de Jira : « New feature » et « Task ».

Afin que le projet adopte ces évolutions j'ai diffusé l'information aux membres de Solutions Isonéo, même ceux travaillant sur d'autres projets, afin qu'ils puissent prendre en compte l'évolution du projet « Citrus » et les potentiels bénéfices que cela pourrait leur apporter. Ces informations ont également été capitalisées dans un document interne à la société Solutions Isonéo.

2.3. Environnement de développement

Le travail sur Citrus ayant déjà commencé, l'environnement de développement est clairement défini.

J'ai cependant dû me former sur Git afin de stocker le dépôt sur le serveur de Solutions Isonéo.

2.3.1. Appliquer les contraintes du partenaire

Comme Jira, Git propose soit d'héberger le projet sur un serveur distant auquel les développeurs ont accès via une interface web (nommée Github), soit d'héberger le dépôt sur n'importe quelle machine. La première option a été choisie par Solutions Isonéo par simplicité mais les contraintes imposées par CMC Electronique demandaient un passage à la seconde option dans l'année.

De ce fait, j'ai mené une étude – dont les principales connaissances ont été tirées du « livre » officiel en ligne de Git cité en sitographie [4] - pour migrer d'un hébergement du dépôt Git sur un serveur distant « inconnu » vers le serveur privé de Solution Isonéo.

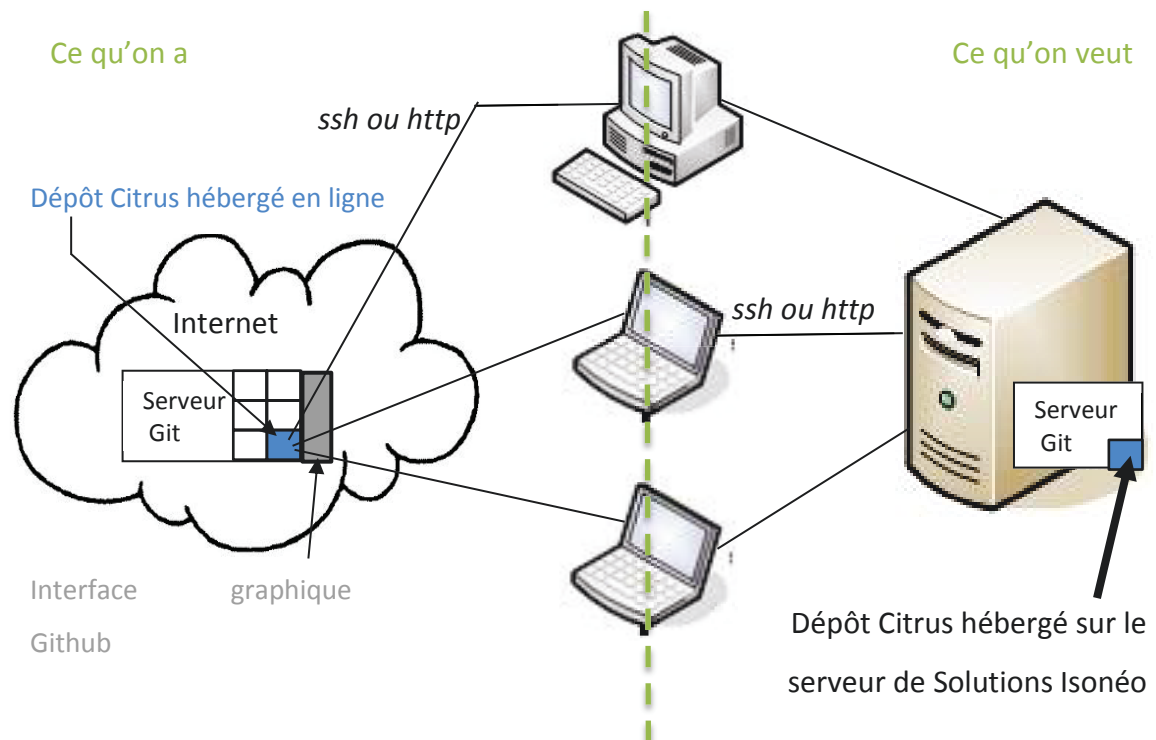


Figure 25 : Différence d'architecture entre Git en ligne et Git hébergé sur le serveur de Solutions Isonéo

Du point de vue système, Git est un outil qui fonctionne à partir d'une installation rapide disponible pour différents systèmes d'exploitation. Il possède une gestion graphique et une gestion en ligne de commande. Git est donc un composant du système d'information qui doit être installé sur tous les postes et qui permet également de transformer sa machine en serveur Git si on décide d'installer le dépôt sur son propre ordinateur. Comme dit précédemment le serveur peut également être la machine externe au réseau de la société si on souhaite héberger le dépôt en ligne et l'exploiter grâce à l'interface web graphique nommée Github.

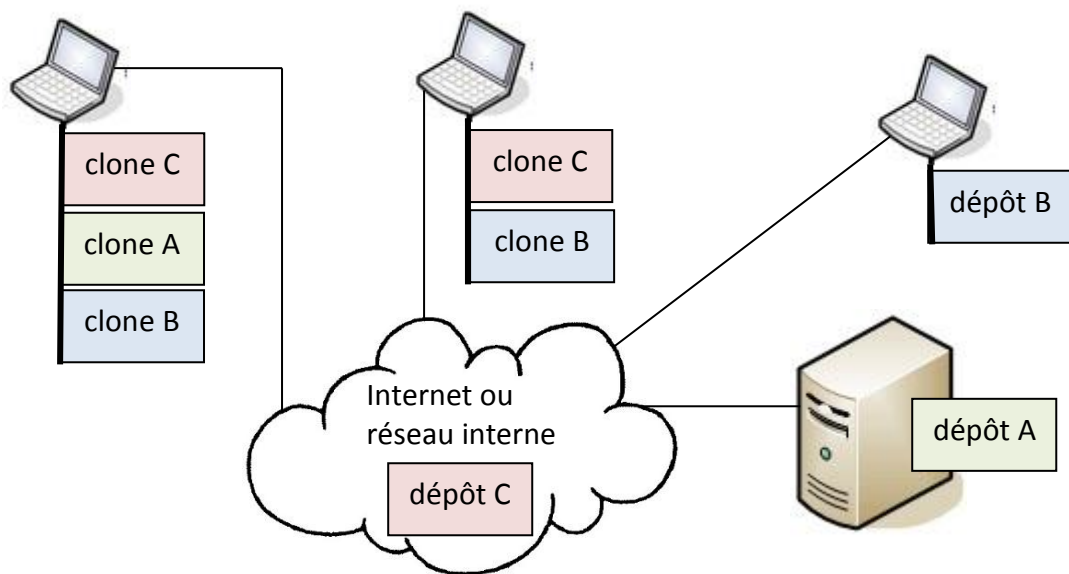


Figure 26 : Architecture et système de Git

L'objectif est d'installer Git sur le serveur de Solutions Isonéo puis de créer un dépôt : les dossiers systèmes du dépôt étaient accessibles et ce dernier pouvait désormais être totalement administré. En effet, ces éléments n'étaient pas visibles sur le dépôt distant de « Citrus » car celui-ci avait été créé par l'interface Github qui cache la vue « système » d'administration. Git propose deux vues : la vue d'administration disponible à l'endroit où le dépôt a été créé et la vue d'utilisation pour chaque clonage de ce dépôt.

L'administrateur n'aura pas accès aux sources, celles-ci peuvent donc conserver leur caractère « secret » car si un service différent gère l'administration du dépôt, il ne pourra pas avoir accès au code. Par contre il a accès à toutes les fonctionnalités d'administration du dépôt dont la partie la plus importante est « hooks ». Les « hooks » sont les procédures automatiques à appliquer avant ou après les actions de dépôt de

code. Les programmeurs peuvent ainsi décider de vérifier lors d'un « commit » que les fichiers possèdent une extension autorisée, ils peuvent ainsi exclure tout ajout de jar sur leur dépôt. Solutions Isonéo n'a cependant pas utilisé cette fonctionnalité généralement appliquée sur des projets plus importants.

L'utilisateur a accès uniquement au code source et lorsqu'il consultera un clone du dépôt il y verra son arborescence projet.

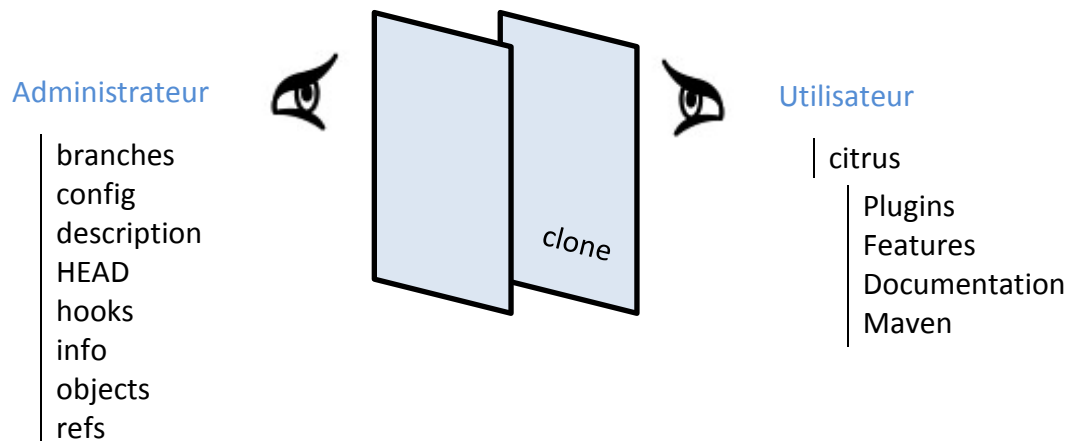


Figure 27 : Vue de l'administrateur et de l'utilisateur sur un même dépôt Git

Nous avons donc décidé qu'en tant qu'administratrice du système d'information, je devais créer sur le serveur un dossier nommé « Repository » où Solutions Isonéo stockerait tous les dépôts projets. Pour transformer un sous dossier de « Repository » en dépôt git il suffit de se positionner à l'intérieur puis de saisir la commande : `git init -bare`.

Afin de contrôler l'accès aux codes ajoutés sur ces dépôts, j'ai mis en place une authentification cryptée grâce à la commande : `htpasswd -c /home/GIT/passwd ebardaji` ; l'option « -c » étant utilisée uniquement la première fois pour l'initialisation du fichier. Cette commande crée l'entrée « ebardaji » dans le document « passwd » et demande en retour la saisie d'un mot de passe à l'écran pour ensuite le sauvegarder de manière cryptée dans le fichier « passwd ». J'ai ensuite reproduit la manipulation en demandant à chacun des membres de Solutions Isonéo de venir saisir leur mot de passe à l'écran. Cette manipulation a dû également être capitalisée dans un document afin

qu'un membre de Solutions Isonéo puissent créer les accès pour les futurs salariés de la société.

L'exploitation de ce mot de passe crypté requiert cependant la création d'un lien entre Git et le fichier « passwd ». De même l'accès entre ces dépôts et les machines utilisateurs est nécessaire. J'ai mis en place un service « Apache2 » pour créer cette connexion (Sitographie [5]). L'installation fût simple, il fallait ensuite déclarer au service l'existence de Git. Pour cela j'ai créé un fichier `GIT.conf` dans les sites disponibles d'Apache (`/etc/apache2/sites-available/`):

```
Alias /git/ « /home/GIT/repositories/ »
<Directory /home/GIT/repositories>
    DAV on
    Options Indexes FollowSymlinks MultiViews
    AllowOverride None
    Require all granted
    AuthType Basic
    AuthName "Git"
    AuthUserFile /home/GIT/passwd
    Require valid-user
</Directory>
```

Les principales caractéristiques de ce fichier sont :

- ❖ L'Alias qui permet lors de l'accès depuis un poste extérieur de saisir `http://IP_du_serveur:port/git/Nom_du_dépôt`. Le service Apache du serveur peut interpréter `/git/` en chemin physique sur son disque soit `/home/GIT/repositories` qui est l'endroit où nous souhaitons stocker l'intégralité des dépôts de la société.
- ❖ Les options « AuthType », « AuthName », « AuthUserFile » et « Require » définissent la configuration d'accès qui dans le cas présent demande un nom d'utilisateur et un mot de passe (Basic) pour « Git » dont le fichier de vérification pour les mots de passe se trouve dans `/home/GIT/passwd`. La connexion est acceptée uniquement aux personnes qui saisissent un bon nom d'utilisateur et le mot de passe associé.
- ❖ Les options « AllowOverride » et « Require » permettent de définir la politique d'accès depuis le réseau. Dans le cas présent, ils permettent à tous les utilisateurs venant de n'importe quel réseau de tenter la connexion au dépôt.

Après redémarrage du service Apache pour prendre en compte ces configurations, j'ai rencontré un problème qui m'a obligé à rentrer plus précisément dans le fonctionnement.

Le dossier d'installation d'Apache2 contenait deux dossiers : « sites-available » et « sites-enabled ». A l'intérieur de ceux-ci deux fichiers « GIT.conf » : l'un créé manuellement dans « sites-available » et le second créé par le service et étant un lien vers le premier.

A côté de ces deux dossiers s'en trouvaient deux autres « mod-available » et « mod-enabled ». De la même façon, le second dossier était peuplé uniquement de lien vers le premier. Cependant tous les éléments de « mod-available » n'étaient pas pointés par ceux de « mod-enabled ». En recherchant sur Internet, j'ai trouvé que les modules installés par l'utilisateur étaient stockés dans les bibliothèques d'Apache2 sous forme de « .so » et peuplées ensuite le dossier « mod-available ». J'ai donc recherché les modules dont j'avais besoin pour mettre en place la connexion avec Git. Un des modules posait problème : `mod_authn_core.so`. Il était effectivement dans `/lib/apache2/modules`. Celui-ci devait déployer dans « mod-available » le fichier `authn_core.load` mais aucune référence ne pointait sur celui-ci depuis le dossier « mod-enabled ». J'ai donc créé le lien avec la commande suivante écrite sur une seule ligne :

```
ln -s /etc/apache2/mod-available/authn_core.load  
/etc/apache2/mod-enabled/authn_core.load
```

Après redémarrage du service Apache, les salariés de Solutions Isonéo avait bien accès aux dépôts stockés sur le serveur de la société.

Cependant cette tâche étant relayée à une priorité moindre, l'opération de migration des sources existantes n'a pas été réalisée durant ma présence. Un document explicatif a cependant été rédigé car l'action est réalisable par n'importe quel employé de Solutions Isonéo.

Le système d'information de Solutions Isonéo s'enrichit d'un nouvel outil.

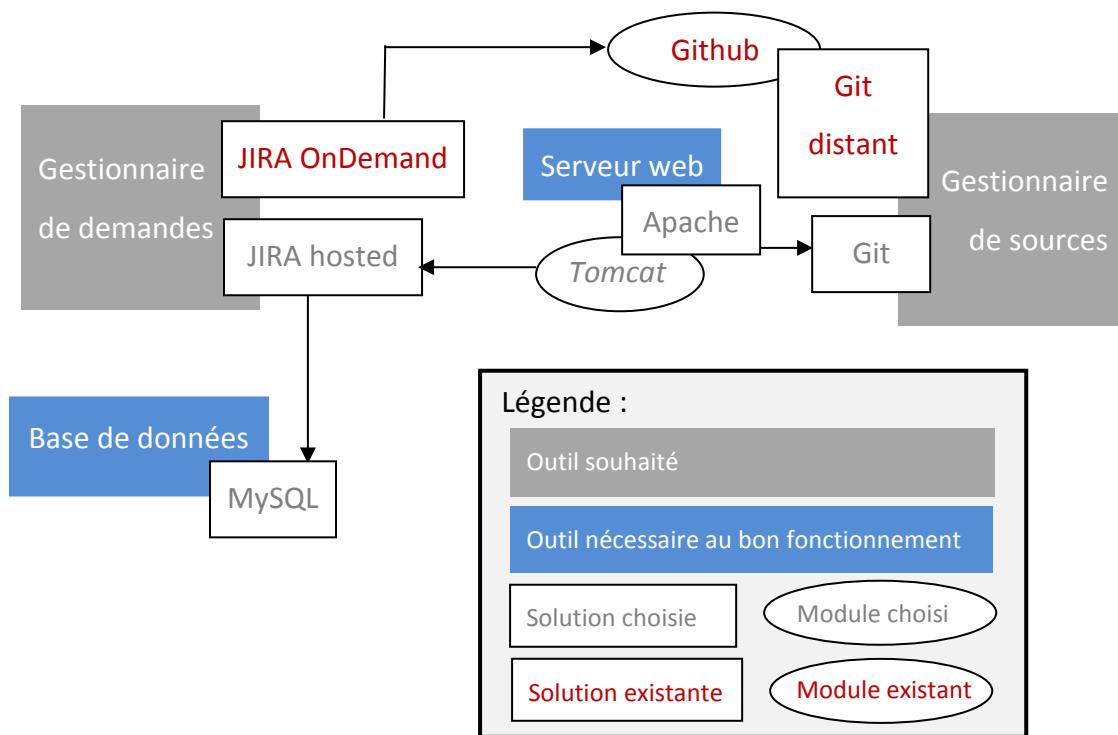


Figure 28 : Composants du S.I. en construction : prise en compte de « Git » hébergé sur le serveur de la société

2.3.2. Générer le produit

Le produit de Citrus était généré manuellement par les programmeurs grâce à Eclipse (Sitographie [3]). Solutions Isonéo devait donc intégrer un nouvel outil pour réaliser la même action sans utiliser l'interface graphique de l'IDE.

Julien Siega avait déjà cherché des solutions et m'a proposé deux axes de recherche : Buckminster (Sitographie [6]) ou Tycho (Sitographie [7]).

Ces deux solutions engendraient la mise en place d'un outil de compilation, nommé Maven (Sitographie [8]), extérieur à celui embarqué dans Eclipse qui m'était de toute façon obligatoire pour la génération automatique que j'installerai dans un second temps.

a) Maven

Cet outil est également utilisé par Artal Technologies et pour des besoins d'unification des procédés, il était recommandé de coupler le projet Citrus à cette technologie (Bibliographie [4]).

Maven est un outil de compilation qui a atteint actuellement sa troisième version et qui possède comme principal avantage le téléchargement automatique des dépendances. En effet il suffit aux programmeurs d'indiquer les librairies dont ils ont besoin et Maven se chargera d'obtenir celles-ci ainsi que celles dont elles dépendent.

Eclipse embarque un plugin nommé « m2e » permettant d'intégrer Maven en version 2 dans l'IDE ; aucun plugin n'embarquant la version 3 n'est pour l'instant disponible sur Eclipse Juno. Cependant la version 3 est rétro-compatible avec la version 2 et apporte uniquement de la rapidité dans le traitement des compilations parallèles ce qui n'impacte pas l'architecture ou les configurations à mettre en place au niveau du projet. De ce fait les programmeurs habitués à l'environnement Eclipse peuvent utiliser Maven grâce au plugin « m2e » alors que j'utilise Maven 3 sur la machine en ligne de commande pour externaliser la compilation de l'environnement Eclipse.

Une fois le plugin « m2e » intégré à Eclipse, tous les projets Citrus pouvaient être rapidement migrés en projet Maven grâce à l'interface.

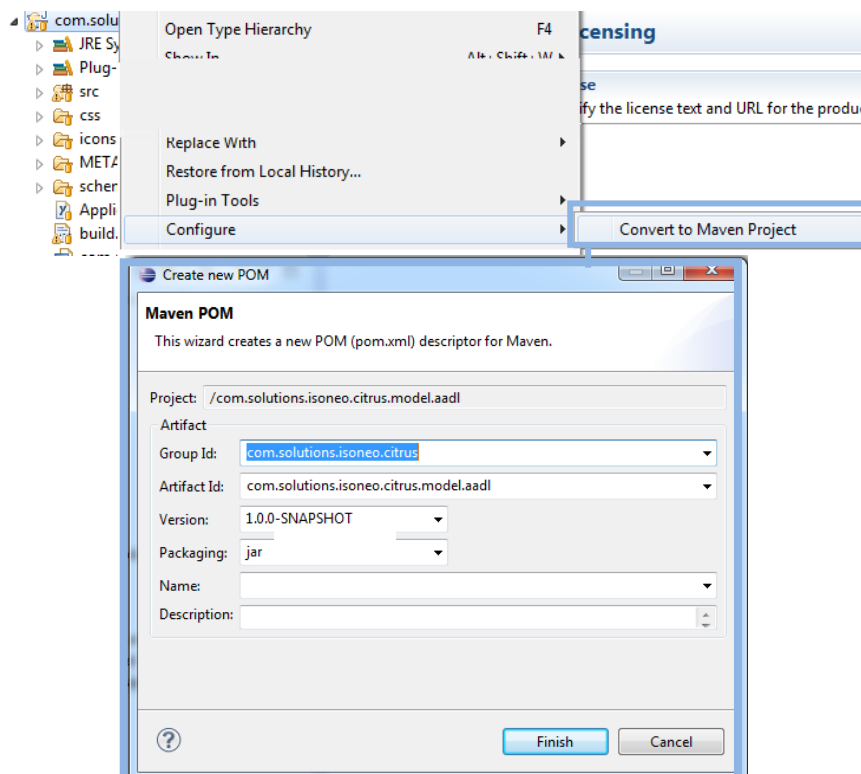


Figure 29 : Conversion d'un projet en projet Maven via l'interface d'Eclipse

Maven est basé sur la mise en place de fichiers xml : `pom.xml`, contenant toute la configuration du projet. Comme le montre l'impression écran précédente, le fichier xml va contenir 4 principales balises :

- ❖ `<groupId>` contient le nom du projet,
- ❖ `<artifactId>` contient le nom unique du module – sous ensemble du projet défini précédemment,
- ❖ `<version>` contient la version de ce projet. Pour Citrus, après une réunion avec les programmeurs, le format a été défini sous forme de *a.b.c* où *a* est la version de production livrée - actuellement toujours à 0, *b* est le numéro du « Sprint » courant, et *c* est incrémenté à chaque compilation de correction,
- ❖ `<packaging>` contient le format de sortie du projet après compilation. Il peut être en plusieurs formats dont « jar » est celui par défaut.

J'ai ensuite travaillé sur un des modules de Citrus afin de créer un échantillon qui fonctionne pour ensuite généraliser sa configuration sur l'intégralité du projet.

J'ai travaillé sur la mise en place des balises supplémentaires et nécessaires à la compilation de ce fichier xml. Le projet Citrus utilisant d'autres plugins déjà développés sous Eclipse par d'autres sociétés, le projet devait pouvoir récupérer des éléments de ces autres projets. Pour cela j'ai inséré la balise `<repositories>` précisant à Citrus sur quels autres dépôts Maven, toutes les librairies nécessaires étaient disponibles. Par exemple pour utiliser des composantes d'Obeo, j'ai inséré :

```
<repository>
  <id>obeo</id>
  <layout>p2</layout>
  <url>http://marketplace.obeonetwork.com/updates/od61/</url>
</repository>
```

Pour tester le bon fonctionnement de ce paramétrage, je suis montée en compétences sur les cycles de compilation, nommé en anglais « build lifecycle », que proposait Maven.

Plusieurs cycles de vie existent en fonction des besoins qui ont émergés dans les différents projets du monde entier utilisant Maven. Cependant n'ayant pas de spécification particulière nous avons décidé de travailler avec le cycle de vie Maven par

défaut. Celui-ci contient différentes phases en fonction de l'action que nous souhaitons faire sur le projet.

Citrus étant un projet Eclipse et devant ensuite générer le produit à chaque compilation de façon automatique, Solutions Isonéo a décidé d'utiliser la phase « install » afin de créer le package Citrus à chaque compilation Maven pouvant être à son tour utilisé dans un autre projet Eclipse. En lançant la commande `mvn install` sur le projet « échantillon », on obtient un dossier `target` qui contient le jar de ce projet ainsi que les fichiers « .class » correspondant aux sources compilées. Pour nettoyer le résultat de la génération avant de relancer une compilation, Maven prévoit la phase « clean ».

Ces deux commandes `install` et `clean` sont appelés « goals ». Ceux-ci proviennent cependant de plugins Maven qu'il faut insérer dans la configuration du projet pour pouvoir les utiliser.

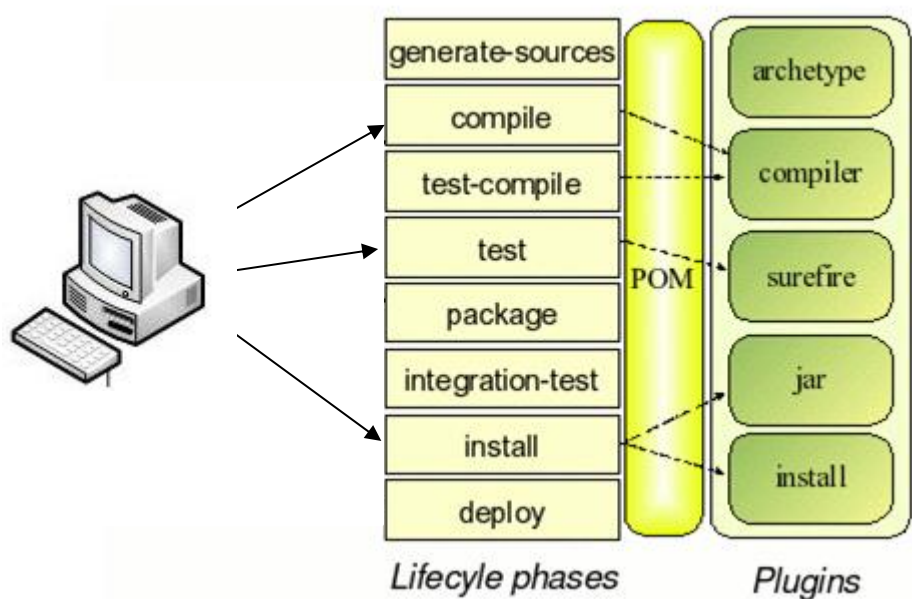


Figure 30 : Correspondance entre les « goals » Maven et ses plugins

Les plugins `install` et `clean` ont été ajoutées dans le `pom.xml`.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-install-plugin</artifactId>
      <version>2.3.1</version>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-clean-plugin</artifactId>
      <version>2.3.1</version>
    </plugin>
  </plugins>
</build>
```

Le projet « échantillon » compilait correctement. Cependant au moment d'appliquer la configuration contenue dans le `pom.xml` pour la trentaine de modules existants dans Citrus, j'ai trouvé que la maintenabilité était très réduite. En effet si plusieurs modules devaient utiliser le même plugin Eclipse de type Obeo, tous les « pom » associés à ces projets devaient être modifiés. J'ai donc organisé une réunion avec Julien Siega après avoir consulté les dépôts disponibles sur github des plugins que nous utilisons comme Obeo, Osate, etc. pour discuter de l'architecture de notre projet.

Nous avons décidé que Solutions Isonéo uniformiserait l'architecture de tous ses projets suivant l'organisation suivante :

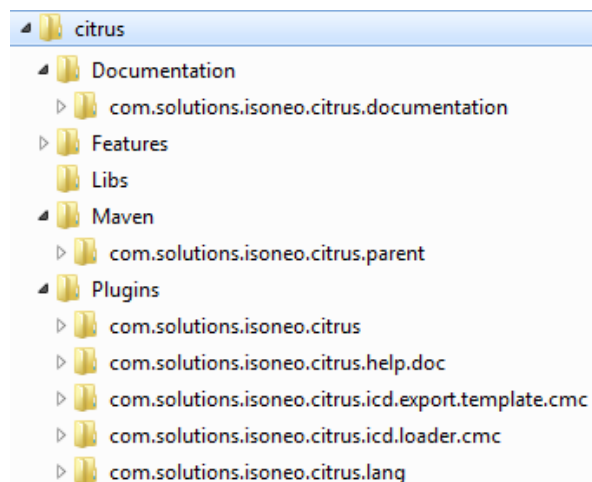


Figure 31 : Architecture du projet Citrus, référence pour tous les projets de Solutions Isonéo

Le dossier « Maven » contient le fichier pom.xml qui contient toutes les configurations vers les dépôts externes, les plugins, les dépendances, etc. communément nommé « pom parent ». Il appelle ensuite tous ses modules qui n'ont aucune configuration propre. De ce fait, si le projet Citrus a besoin d'un nouvel élément de configuration, il sera ajouté dans le « pom parent » et impactera tous les modules.

Pour déclarer ce fichier pom.xml comme « pom parent », il faut déclarer le lien « parent/enfant ». Le « pom parent » va déclarer chacun des éléments à compiler grâce à la balise `module` :

```
<modules>
  <module>../../Plugins/com.solutions.isoneo.citrus</module>
  <module>etc...</module>
  <module>../../Plugins/com.solutions.isoneo.citrus.lang</module>
</modules>
```

Le chemin est relatif, c'est-à-dire composé de « `../../` » afin que le projet puisse être reconstruit sur n'importe quelle machine. En effet si le chemin précisé était un chemin absolu de type `D:/Mes Documents/...` tous les programmeurs seraient obligés de l'adapter pour le rediriger vers l'endroit où ils ont souhaité enregistrer le projet sur leur disque. Le « pom parent » se trouvant dans `Maven/com.solutions.isoneo.citrus`, le chemin signifie qu'il faut sortir de `com.solutions.isoneo.citrus.parent` puis sortir de `Maven` pour se retrouver à la racine du projet dans « `citrus` » pour entrer dans le dossier « `Plugins` » et atteindre, dans chacun des modules, le pom.xml.

Le format de sortie change également pour le « pom parent » et devient de type « pom » :

```
<packaging>pom</packaging>
```

Les modules doivent déclarer leur projet parent avec la balise `parent` :

```
<parent>
  <groupId>com.solutions.isoneo.citrus</groupId>
  <artifactId>com.solutions.isoneo.citrus.parent</artifactId>
  <version>0.7.0</version>
  <relativePath>../../Maven/com.solutions.isoneo.citrus.parent
                                </relativePath>
</parent>
```

Les modules conservent cependant leur identité :

```
<modelVersion>4.0.0</modelVersion>
<artifactId>com.solutions.isoneo.citrus</artifactId>
<packaging>jar</packaging>
```

De cette façon, la compilation doit s'effectuer à partir de `Maven/com.solutions.isoneo.citrus.parent` et appelle lui-même ses modules pour les compiler.

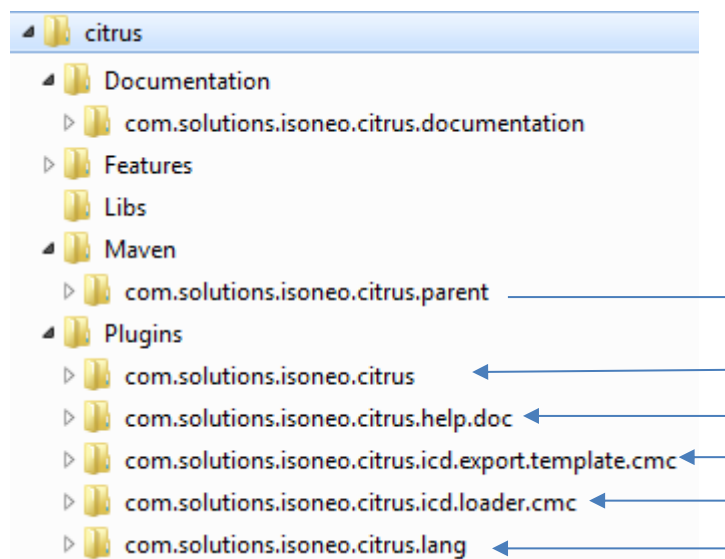


Figure 32 : Enchaînement des compilations à partir du pom parent

Dans le projet Citrus, nous avons également mis en place un dossier « Libs » pour embarquer les librairies dont les sites sont souvent inaccessibles sur internet et que nous ne pouvons donc pas mettre dans les balises `<repository>`.


```

<dependencies>
  <dependency>
    <groupId>fr.openpeople.ui</groupId>
    <artifactId>qudv</artifactId>
    <version>0.5.0.201212131714</version>
    <scope>system</scope>
    <systemPath>${project.basedir}/../../Libs/fr.openpeople.ui.qudv
    _0.5.0.201212131714.jar</systemPath>
  </dependency>
</dependencies>

```

Une fois cette nouvelle configuration généralisée, j'ai pu vérifier que l'intégralité des modules compilés :

```

[INFO] --- cyano-p2-plugin-0.16.0:update-local-index (default:update-local-index) @ com
[INFO] Reactor Summary:
[INFO] com.solutions.isoneo.citrus.parent ..... SUCCESS [6.139s]
[INFO] com.solutions.isoneo.citrus.model ..... SUCCESS [32.801s]
[INFO] com.solutions.isoneo.citrus.icd.export.template.cmc ..... SUCCESS [1.279s]
[INFO] com.solutions.isoneo.citrus.ui.utils ..... SUCCESS [1.989s]
[INFO] com.solutions.isoneo.citrus.icd.loader.cmc ..... SUCCESS [1.303s]
[INFO] com.solutions.isoneo.citrus.lang ..... SUCCESS [0.779s]
[INFO] com.solutions.isoneo.citrus.ui.wizard.project ..... SUCCESS [5.885s]
[INFO] com.solutions.isoneo.citrus.obeo ..... SUCCESS [10.676s]
[INFO] com.solutions.isoneo.repository.client.core ..... SUCCESS [1.800s]
[INFO] com.solutions.isoneo.repository.client.ui.view ..... SUCCESS [2.265s]
[INFO] com.solutions.isoneo.citrus.ui.view.project ..... SUCCESS [5.557s]
[INFO] com.solutions.isoneo.citrus ..... SUCCESS [2.577s]
[INFO] com.solutions.isoneo.citrus.help.doc ..... SUCCESS [1.969s]
[INFO] com.solutions.isoneo.citrus.model.aadl ..... SUCCESS [1.444s]
[INFO] com.solutions.isoneo.citrus.model.sysml ..... SUCCESS [0.975s]
[INFO] com.solutions.isoneo.citrus.papyrus ..... SUCCESS [0.875s]
[INFO] com.solutions.isoneo.repository.citrus.asset.provider ..... SUCCESS [0.732s]
[INFO] com.solutions.isoneo.repository.client.ui ..... SUCCESS [0.256s]
[INFO] com.solutions.isoneo.repository.provider.citrus.asset.manager ..... SUCCESS [1.004s]
[INFO] com.solutions.isoneo.repository.provider.file ..... SUCCESS [0.847s]
[INFO] com.solutions.isoneo.repository.server.generic.asset.manager ..... SUCCESS [0.679s]
[INFO] com.solutions.isoneo.repository.feature ..... SUCCESS [0.953s]
[INFO] com.solutions.isoneo.citrus.feature.obeo ..... SUCCESS [3.629s]
[INFO] com.solutions.isoneo.citrus.feature.basis ..... SUCCESS [14.363s]
[INFO] com.solutions.isoneo.citrus.feature.help ..... SUCCESS [0.470s]
[INFO] com.solutions.isoneo.citrus.feature.team ..... SUCCESS [0.662s]
[INFO] com.solutions.isoneo.citrus.feature.osate ..... SUCCESS [3.306s]
[INFO] com.solutions.isoneo.citrus.feature.papyrus ..... SUCCESS [10.730s]
[INFO] com.solutions.isoneo.citrus.repository ..... SUCCESS [3:17.193s]
[INFO] com.solutions.isoneo.citrus.tests ..... SUCCESS [43.719s]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 14:53.876s
[INFO] Finished at: Thu Oct 03 16:56:22 EDT 2013
[INFO] Final Memory: 225M/527M
[INFO]

```

Figure 33 : Rapport de compilation Maven en ligne de commande

Le projet Citrus utilise le concept de « Feature », visible sur l'impression écran précédente : `com.solutions.isoneo.citrus.feature.obeo` par exemple. Une « Feature » rassemble plusieurs modules. Les programmeurs peuvent ainsi ne générer qu'une partie de l'application Citrus en excluant toutes les fonctionnalités qui ont été développées dans le module Obeo par exemple. Plusieurs « Features » ont été mises en

place dans le projet. Une fois que le projet Citrus compilait avec Maven, cette notion m’a servi à mettre en place la génération du produit.

b) Génération du produit

Buckminster est un projet Eclipse et peut être intégré à l’IDE sous forme de plugin à télécharger. Il s’intègre également aux outils d’intégration continue permettant la génération automatique via un plugin.

Tycho est un plugin Maven, qui n’impacte le projet que par la configuration présente dans le fichier pom.xml. Il implique donc un téléchargement du plugin Tycho dans Eclipse mais s’intègre à des outils d’intégration continue comme un simple projet Maven. Les deux solutions sont gratuites cependant beaucoup plus de tutoriels existent sur le plugin Tycho. J’ai donc choisi la seconde solution.

Pour intégrer Tycho au projet Citrus, j’ai dû modifier le « pom parent » pour y insérer le plugin.

```
<plugin>
  <groupId>org.eclipse.tycho</groupId>
  <artifactId>tycho-maven-plugin</artifactId>
  <version>${tycho-version}</version>
</plugin>
```

J’ai décidé d’utiliser les « propriétés » dans le « pom parent » afin de centraliser les versions des éléments importants comme Tycho ou le format d’encodage :

```
<properties>
  <project.build.sourceEncoding>UTF-8
  </project.build.sourceEncoding>
  <tycho-version>0.18.0</tycho-version>
</properties>
```

Avec Tycho il était également nécessaire de changer le format de sortie dans lequel les modules de type « plugins » devaient être compilés :

```
<packaging>eclipse-plugin</packaging>
```

Ainsi que ceux de type “Feature” :

```
<packaging>eclipse-feature</packaging>
```

Pour intégrer la génération du produit Citrus, l'ajout d'un « Repository » est nécessaire. Le « Repository » peut être lié à une ou plusieurs « Features » qui elles-mêmes sont reliées à un ou plusieurs « plugins ». Dans le cas de la génération du produit, la « Feature » utilisée pointe vers tous les plugins internes et externes. Tous les éléments doivent avoir des fichiers pom.xml et être liés au « pom parent » créé dans le dossier « Maven » du projet. En utilisant l'interface graphique d'Eclipse, il est facile de créer un projet `com.solutions.isoneo.citrus.repository` dans un nouveau dossier « Repository », de lui attribuer la « Feature » `com.solutions.isoneo.citrus.feature.basis` pointant vers tous les plugins du projet et de lui ajouter un fichier pom.xml avec la configuration adéquate.

Le produit se génère ensuite dans le projet « Repository » :

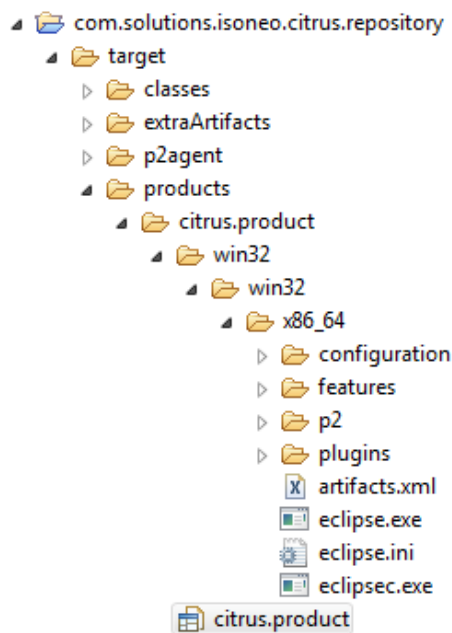


Figure 34 : Génération du produit

Pour clôturer correctement cette compilation et génération, j'ai décidé d'insérer des tests dans le projet afin de configurer le « pom parent », l'architecture du projet ainsi que le cycle de compilation en conséquence.

c) Tests

Maven préconise la mise en place de tests dans chacun des modules sous l'architecture `src/test` alors que les sources sont stockées dans `src/main`. Cependant Tycho ne

fonctionne pas suivant la même architecture, l'outil requiert que la gestion des tests se fasse dans un autre dossier. L'architecture du projet Citrus a été enrichi du dossier « Tests » dans lequel serait créé autant de projets que de modules, par exemple `com.solutions.isoneo.citrus.test` ou encore `com.solutions.isoneo.citrus.help.doc.test`. Chacun d'entre eux doit posséder un fichier `pom.xml` avec le même « pom parent » que les « plugins » ou les « features ».

Les tests à déployer seront pour la plupart dans un premier temps des tests unitaires nécessitant la librairie JUnit :

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
</dependency>
```

Au lancement de la compilation Maven avec la commande `mvn clean install`, un rapport de test est apparu en plus de la compilation :

```
-----
T E S T S
-----
Running com.solutions.isoneo.citrus.TestICDImport
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.715 sec
Running com.solutions.isoneo.citrus.TestPremier
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 sec
Results :
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO] All tests passed!
[INFO]
```

En effet la phase « install » du cycle de compilation Maven contient tous les « goals » précédents et contient donc la phase « test » :

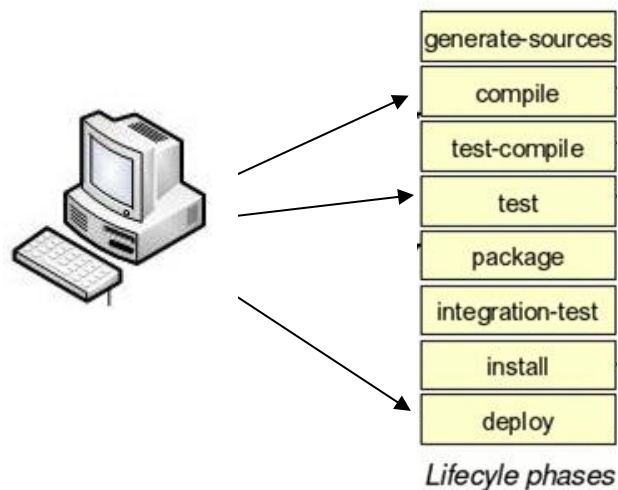


Figure 35 : Le cycle de compilation par défaut Maven

Le système d'information de Solutions Isonéo s'est enrichi d'un nouveau composant permettant la compilation du projet à l'extérieur de l'IDE Eclipse : l'outil de compilation Maven.

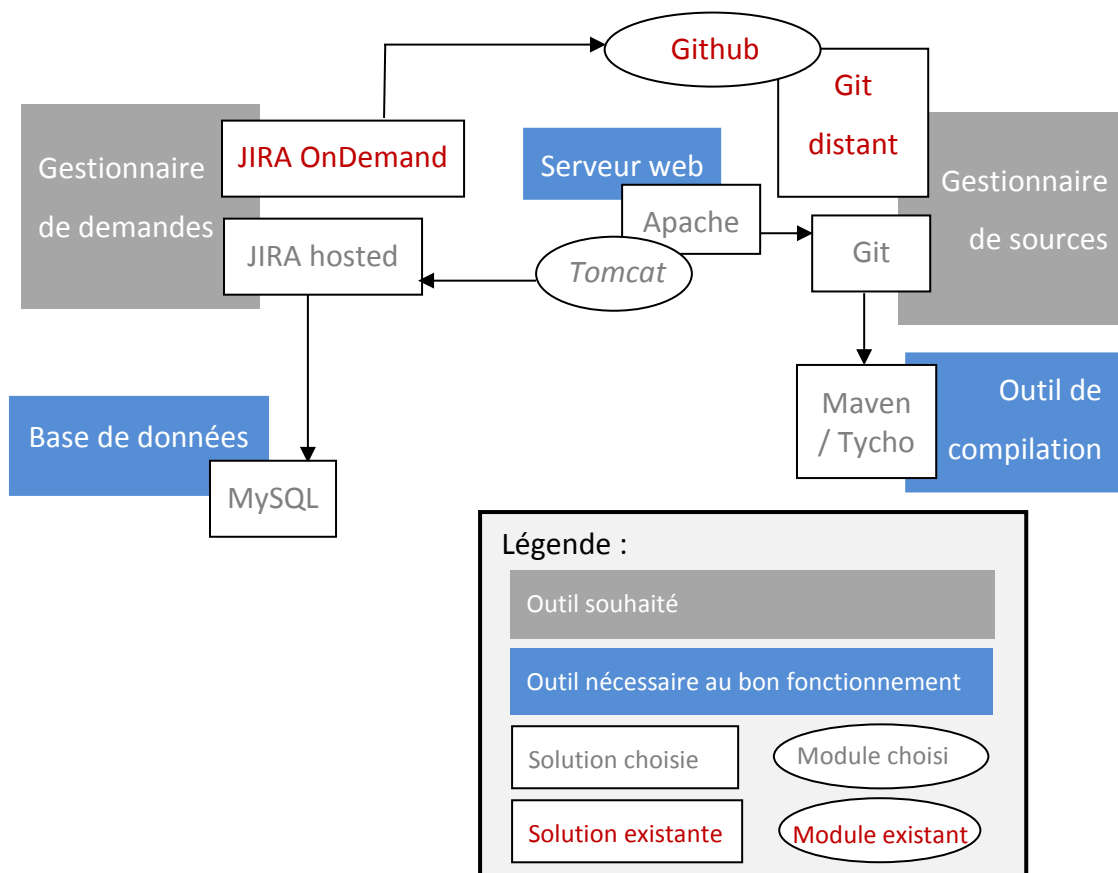


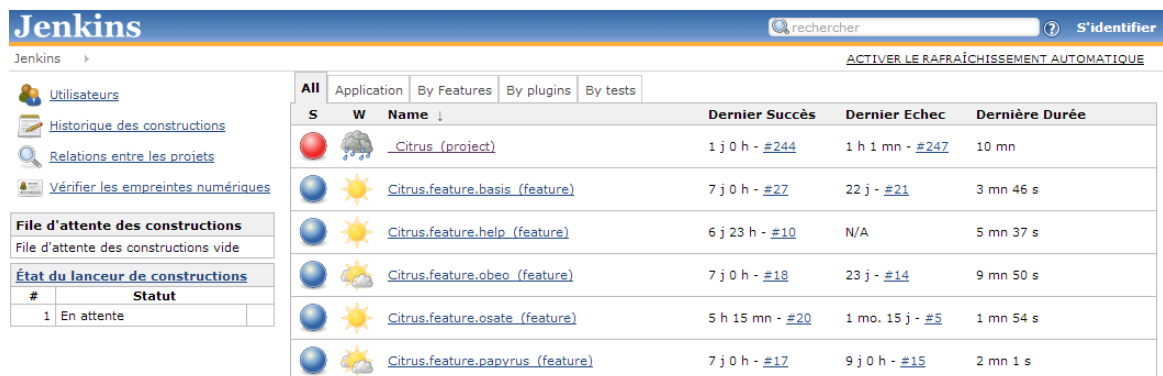
Figure 36 : Composants du S.I. en construction : mise en place d'un outil externe de compilation : Maven

2.4. Aide au développement

2.4.1. Générer automatiquement le produit

L'automatisation de la génération obtenue précédemment passe par la mise en place d'un outil d'intégration continue. Différentes solutions existent sur le marché, cependant deux d'entre elles sont les plus répandues et gratuites : Hudson et Jenkins.

Initialement ce sont les mêmes produits qui naissent tous les deux d'Hudson lorsqu'il était porté par la communauté OpenSource. Oracle a décidé d'acheter Hudson. La communauté OpenSource n'étant pas en accord avec l'orientation proposé par cette société et ayant des craintes sur la gratuité de l'outil, elle a décidé de créer Jenkins il y a environ un an. Entre Oracle et la communauté OpenSource, Solutions Isonéo a pensé que Jenkins serait sûrement plus maintenu, plus évolutif et adaptable aux autres outils portés majoritairement par la communauté OpenSource, outils que la société utilise en priorité car ils présentent un faible coût. Jenkins est un outil web proposant une interface graphique simple pour le configurer (Sitographie [9]).



The screenshot shows the Jenkins web interface. On the left, there's a sidebar with navigation links like 'Utilisateurs', 'Historique des constructions', 'Relations entre les projets', and 'Vérifier les empreintes numériques'. Below these are sections for 'File d'attente des constructions' and 'État du lanceur de constructions'. The main area displays a table of builds, filtered by 'All'. The table has columns for status (S), icon (W), name, last success, last failure, and last duration. The builds listed are for the 'Citrus' project and its various features.

S	W	Name	Dernier Succès	Dernier Echec	Dernière Durée
		Citrus (project)	1 j 0 h - #244	1 h 1 mn - #247	10 mn
		Citrus.feature.basis (feature)	7 j 0 h - #27	22 j - #21	3 mn 46 s
		Citrus.feature.help (feature)	6 j 23 h - #10	N/A	5 mn 37 s
		Citrus.feature.obeo (feature)	7 j 0 h - #18	23 j - #14	9 mn 50 s
		Citrus.feature.osate (feature)	5 h 15 mn - #20	1 mo. 15 j - #5	1 mn 54 s
		Citrus.feature.papyrus (feature)	7 j 0 h - #17	9 j 0 h - #15	2 mn 1 s

Figure 37 : Interface graphique Jenkins

Jenkins est un outil qui est connectable à plusieurs gestionnaires de sources comme Git, SVN ou CVS. Il est également paramétrable selon plusieurs outils de compilation comme Ant, Maven2 ou Maven3. L'outil a ainsi accès à tous les éléments nécessaires à la compilation du code et il suffit de le paramétrer pour programmer des compilations suivant les besoins. Certaines sociétés préfèrent compiler les sources toutes les nuits, d'autres à chaque fois qu'un « commit » est fait sur le gestionnaire de sources. Solutions Isonéo ayant configuré le « pom parent » de Maven3 pour compiler, lancer les tests et

générer le produit de Citrus, Jenkins sera capable d'automatiser ces tâches simplement en utilisant vers ce fichier pom.xml.

Dans un premier temps j'ai configuré une tâche dans Jenkins pour qu'il compile l'intégralité du projet Citrus lors d'une demande manuelle.

a) *Lancement manuel d'une compilation*

Une tâche, nommée « job » en anglais, correspond à une action à déclencher pour Jenkins. Solutions Isonéo a une tâche « _Citrus_(project) » qui correspond à la compilation de la totalité du projet.







	All	Application	By Features	By plugins	By tests
	W	Name ↓			De
zns		Citrus (project)			1 j
sa		Citrus.feature.basis (feature)			7 j
Amérique		Citrus.feature.help (feature)			6 j
tions		Citrus.feature.obeo (feature)			7 j
vide		Citrus.feature.osate (feature)			5 h
ctions		Citrus.feature.nanovic (feature)			7 j

Figure 38 : *Tâches Jenkins*

Pour cela j'ai dû installer les plugins adéquats afin de pouvoir coupler Jenkins à Git via Github.


	GitHub API Plugin This plugin is a library plugin used by other GitHub r no need to install this plugin manually, although you
	GitHub Plugin This plugin integrates Jenkins with Github projects.

Figure 39 : *Liste des plugins nécessaires à la mise en place de Github avec Jenkins*

Ensuite j'ai configuré la tâche.

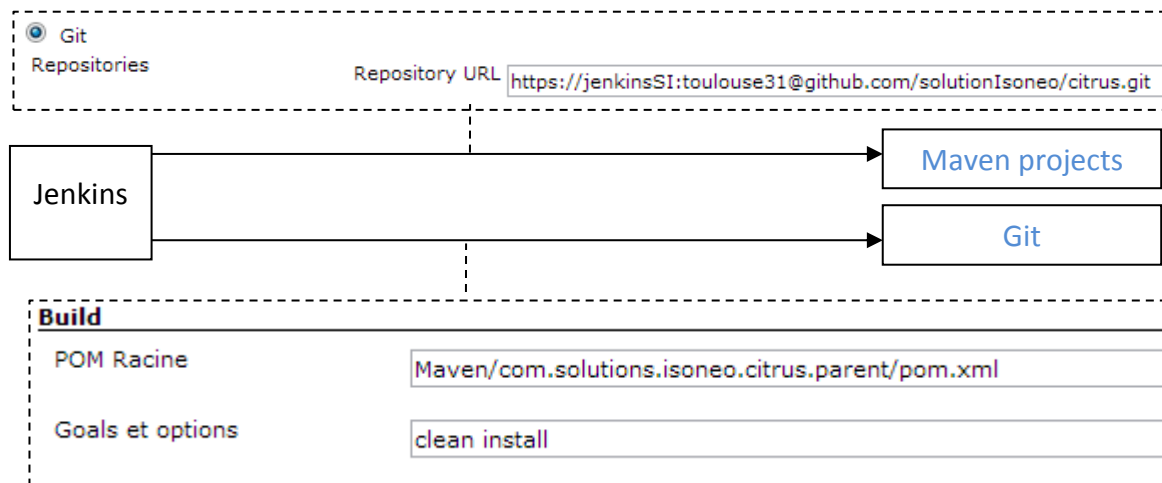


Figure 40 : Schéma des configurations du projet sous Jenkins

De cette façon Jenkins connaît l'URL https du dépôt de sources Git et s'y connecte pour récupérer les sources à compiler via le compte « jenkinsSI » et son mot de passe. Le compte utilisé aurait pu être « solutionsIsoneo » créé pour le lien entre Git et Jira mais ne souhaitant coupler Jenkins qu'au dépôt Citrus, nous avons créé un compte ayant des droits plus restreints. Jenkins peut ensuite, à partir de la racine du projet Git, retrouver le « pom parent » dont nous lui avons donné le chemin ainsi que les phases de compilation à exécuter. Ces quelques informations suffisent à Jenkins pour compiler à la demande d'un utilisateur et générer ainsi le « .product » de Citrus sur le serveur dans les différents dossiers correspondant à chacune des tâches de Jenkins.

b) Liens de l'interface Jenkins

En naviguant sur l'interface de Jenkins, je me suis rendue compte que certains chemins de redirection ne fonctionnaient pas. La difficulté a été de trouver l'origine du problème. Lors de la première compilation, Jenkins intègre de nouvelles icônes comme celle qui permet d'accéder directement au dépôt sur Github :



L'URL à rechercher dans la configuration vers laquelle renvoyait cette icône devait être du type « Github project » mais malheureusement plusieurs paramètres avaient cette dénomination. Après plusieurs tests j'ai trouvé l'adresse vers laquelle renvoyait l'icône Github cependant je devais comprendre l'utilité de la seconde adresse.

2. Mise en œuvre

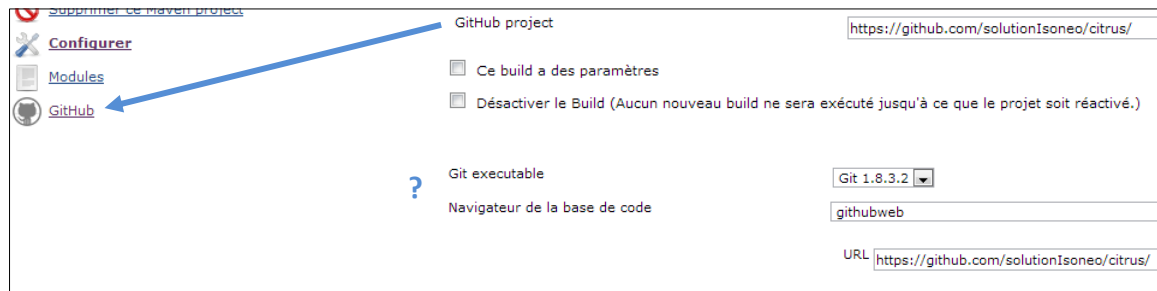


Figure 41 : Lien de redirection vers le projet sous Github

Après plusieurs recherches dans Jenkins, j'ai trouvé que ce paramétrage est utilisé dans la liste de tous les « commits » en rapport avec une compilation de Jenkins.



Figure 42 : Lien « githubweb » dans la liste des « commits »

Ceci apporte une fonctionnalité supplémentaire : il est possible de consulter l'ensemble des lignes ajoutées, supprimées et modifiées de tous les fichiers appartenant à ce « commit ».

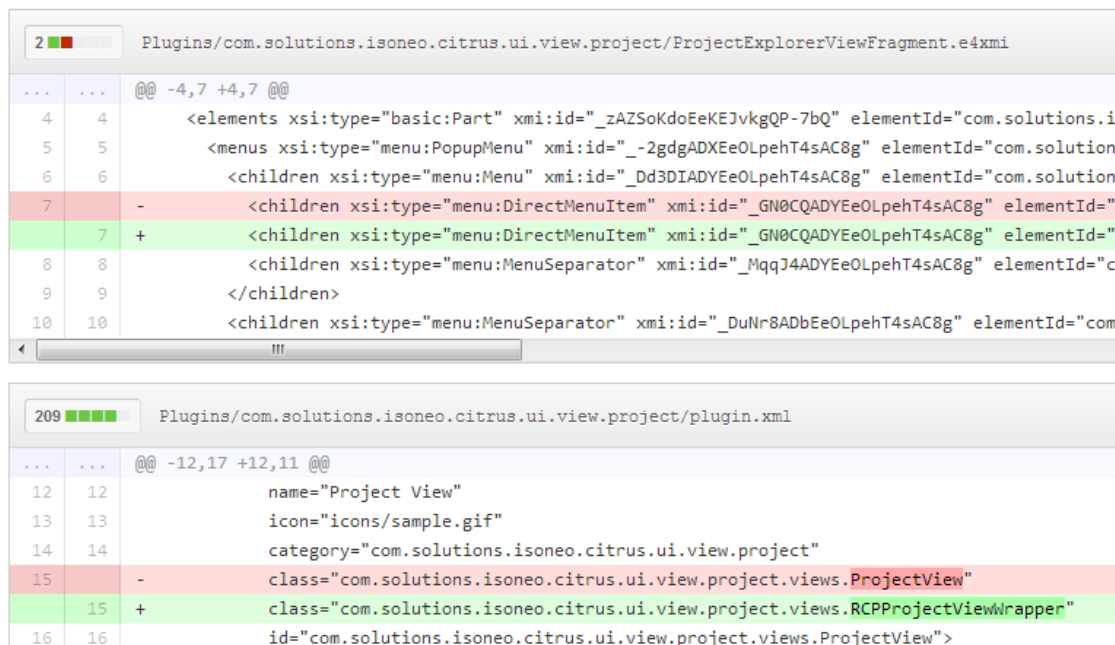


Figure 43 : Exemple de modifications sur des fichiers impactés par un « commit »

c) Gestion des utilisateurs

Dans un second temps j'ai étudié la gestion des utilisateurs. Seul Julien Siega avait besoin d'avoir des droits d'administration sur cette application, les autres programmeurs ou Erwan Deschamps possédant uniquement des droits de lecture.

d) Lancement automatique de la compilation

Pour finir, l'objectif était de lancer une compilation à partir de Jenkins dès qu'un utilisateur faisait un commit.

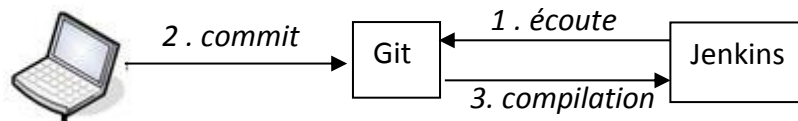


Figure 44 : Schéma de compilation Jenkins

J'ai mis en place une surveillance de Jenkins sur Git en programmant une vérification toutes les 5 minutes de l'existence d'un « commit » durant la dernière période passée.

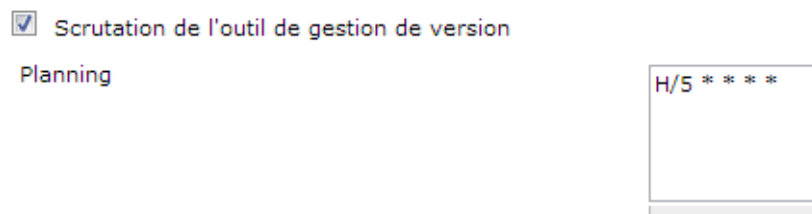


Figure 45 : Paramétrage de la scrutation de Git par Jenkins

Jenkins ne déclenche plus la compilation à la demande de quelqu'un mais informe qu'il y a eu une modification dans la base de code.



Figure 46 : Icône d'information sur le type de lancement de la compilation

J'ai proposé de compiler l'intégralité du projet (project détenant le « pom parent ») mais d'ajouter de nouvelles tâches pour les plugins (modules du « pom parent ») les plus importants et de paramétrer Jenkins afin qu'il ne les compile que si la modification les impacte. La compilation complète permet ainsi de vérifier qu'un module dépendant d'un autre n'est pas impacté par une modification appliquée sur ce dernier.

Le plug-in Git ajouté à Jenkins, permet dans les options avancées d'inclure ou d'exclure des éléments du projet (Included Regions et Excluded Regions).

Dans un premier temps, j'ai pensé que ce paramétrage ne fonctionnait pas car l'intégralité du projet était compilée. J'ai donc décidé de rechercher dans les logs si ce paramétrage impactait bien le comportement avant de lancer la compilation. J'ai alors trouvé les logs du dernier accès à Git pour une tâche. Comme dans l'exemple suivant, les logs contenaient la mention relative aux « excluded paths » témoignant que la configuration fonctionnait correctement :

```
Started on 31-Oct-2013 5:26:11 PM
Polling SCM changes on master
Using strategy: Default
[poll] Last Built Revision: Revision
b1f1fe9f011e04087494eec5b58a32b06b716b2c (origin/master,
origin/HEAD)
Fetching changes from the remote Git repositories
Fetching upstream changes from origin
Polling for changes in
Seen branch in repository origin/AM-12
Seen branch in repository origin/Authentication
Seen branch in repository origin/HEAD
Seen branch in repository origin/RCP
Seen branch in repository origin/master
Seen 5 remote branches
Ignored commit b5c9c848d22e4344f7e597e4129c5ff7a96d0369: Found
only excluded paths:
Done. Took 0.42 sec
No changes
```

D'après ces logs, j'ai pu comprendre que la totalité du projet était compilé quand une modification était identifiée sur le sous-projet lié à la tâche Jenkins.

L'objectif engendrait deux paramétrages :

- ❖ Ecouter uniquement les modifications sur le plugin
- ❖ Ne compiler que le plugin concerné

Pour le premier, j'ai laissé en place la configuration pour inclure la région. Par exemple, avec la configuration suivante, la tâche n'est exécutée par Jenkins que si le plugin « com.solutions.isoneo.citrus » est impacté.



Figure 47 : *Paramétrage Jenkins pour le lancement d'une compilation suivant une modification précise*

Pour le second, j'ai dû modifier le « pom racine » afin que la compilation n'ait lieu que sur le plugin impacté par la modification.



Figure 48 : *Paramétrage Jenkins pour compiler uniquement un plugin du projet*

Désormais le système d'information possède un nouveau composant opérationnel permettant de générer automatiquement le produit de Citrus.

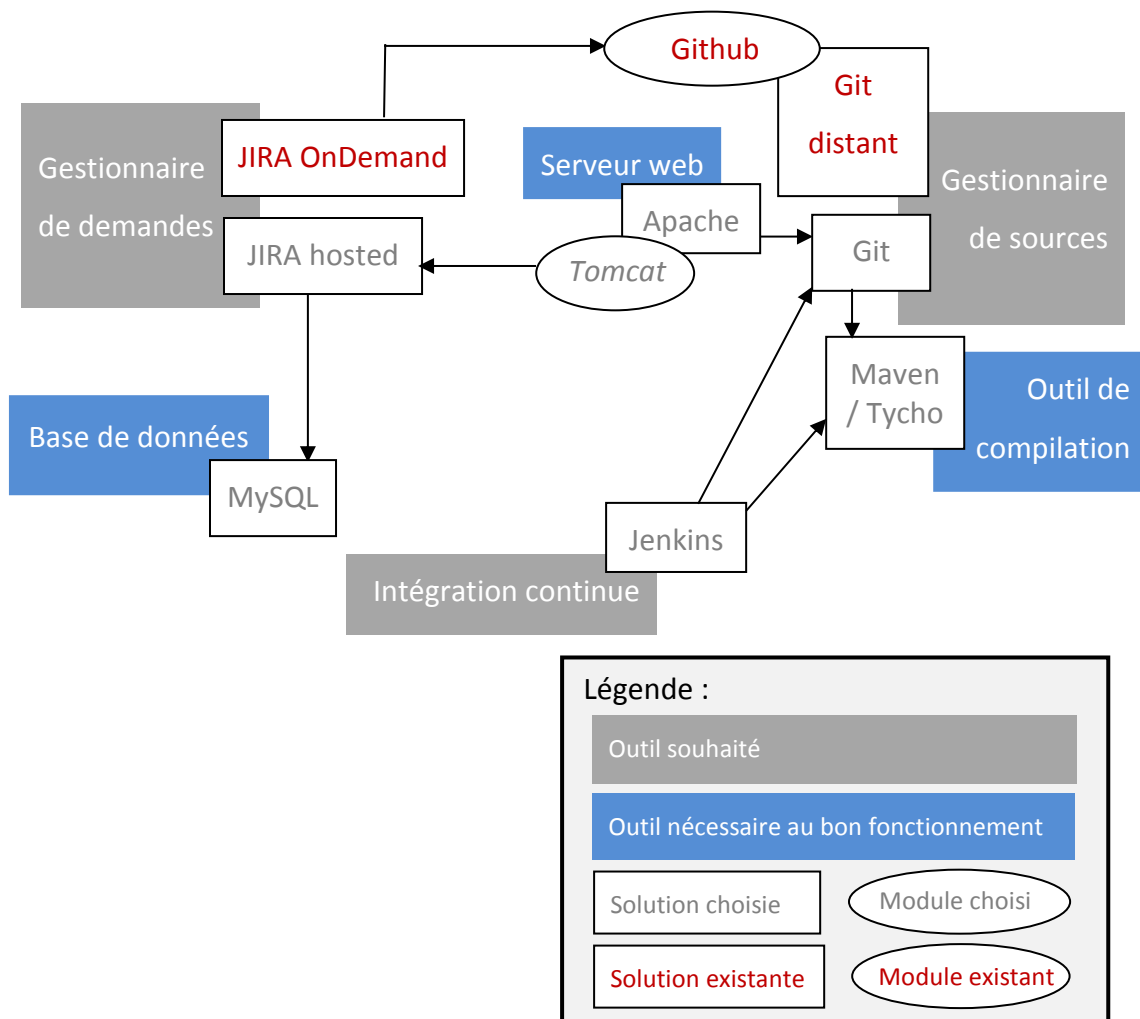


Figure 49 : Composants du S.I. en construction : mise en place de Jenkins

e) Notification par Skype

Afin que les développeurs n'aient pas à regarder le tableau de bord de l'outil Jenkins à chaque fois qu'ils effectuent un « commit », nous avons décidé de mettre en place un système de notification. Jenkins peut, couplé à un serveur mail, envoyer des courriels à chaque fin de compilation. Cependant cela risquait de remplir considérablement la boîte mail des développeurs, notamment celle du responsable en développement, Julien Siega, qui en reçoit déjà un nombre important. Julien Siega m'a alors proposé de mettre en place une notification par Skype.

Cette mise en place présentait rapidement des contraintes d'un point de vue fonctionnel, il fallait que tous les développeurs intervenant sur Citrus possèdent Skype, ce qui était le cas chez Solutions Isonéo mais qui ne serait peut-être pas le cas dans le

futur. Cependant cette contrainte ne fut pas suffisante à Julien Siega pour étudier d'autres possibilités. D'un point de vue opérationnel, cette solution présentait également une principale contrainte : l'application Skype devait être lancée sur le serveur et donc engendrait une ouverture de session permanente. Même si celle-ci ne serait pas accessible car verrouillée, cela signifiait que lors d'un redémarrage de serveur il fallait obligatoire qu'une personne se connecte à la session pour que celle-ci démarre l'application Skype et puisse ainsi rétablir le service. Julien Siega et Erwan Deschamps prirent en considération cette charge.

Pour mettre en place cette solution, j'ai dû, dans un premier temps, créer un compte Skype pour l'outil Jenkins de la société. La possession d'une adresse mail étant nécessaire à la réalisation d'une telle action, un compte de messagerie dédié à l'application Jenkins a été créé.

Ensuite en recherchant sur internet, j'ai trouvé un plugin Jenkins, dédié à la mise en place des notifications automatiques via Skype, cependant celui-ci nécessitait une plateforme 32 bits alors que le serveur de Solutions Isonéo est un système en 64 bits. L'application Jenkins prévoit la possibilité d'ajouter un système esclave et de lui attribuer des caractéristiques différentes du système maitre, cependant cela imposait la mise en place d'une machine virtuelle ou la mise en place de plusieurs logiciels nécessaires à installer en 32 bits avec des risques non négligeable d'impact sur le système d'information déjà mis en place. Julien Siega a confirmé ce risque, et j'ai cherché une solution externe à Jenkins.

Le logiciel « BuildChatBot » semblait adapté au besoin. Développé en python, le logiciel exploite les fonctionnalités de Skype via un langage propriétaire dérivé de python : Skype4Py.

Ce script est basé sur l'existence d'une conversation Skype dans laquelle l'outil va écrire le résultat de compilation des tâches lancées par Jenkins. Cette conversation a donc été initiée par le compte Skype créé pour Jenkins et tous les comptes des différents programmeurs y ont été ajoutés.

BuildChatBot prévoit un script pour identifier cette conversation et passer son identifiant unique en paramètre du script qui va écouter les résultats de compilation lancée par Jenkins pour envoyer ensuite les notifications. Le script principal attend 4

paramètres : l'identifiant unique de la conversation par laquelle il doit notifier les programmeurs, l'adresse du serveur hébergeant l'application Jenkins, l'introduction du message qui sera saisi dans la discussion Skype – dans notre cas [JENKINS] - et la période au bout de laquelle l'outil python doit vérifier les résultats de compilations lancées par Jenkins. Une configuration supplémentaire a dû être appliquée à Jenkins pour que la totalité des compilations soient associées à une notification : Jenkins a été configuré pour ne compiler qu'une seule tâche à la fois. En effet si le paramétrage par défaut était conservé, deux compilations pouvaient donner un résultat quasiment simultanément et une seule notification était envoyée par Skype, une compilation sans succès pouvant ainsi ne donner naissance à aucune notification. Cette période a été initialisée à 15 secondes. Le serveur étant performant et le script python étant de toute façon peu gourmand en ressources, une courte période ne posait pas problème et couvrait également les compilations les plus rapides.

Ce script doit être lancé en arrière-plan via la commande : `python /home/JENKINS/BuildChatBot/buildchatbot.py &`.

Le « & » permet le lancement en arrière-plan : le script poursuivait son exécution même si la fenêtre de lancement de la commande est fermée.

Une réplique de cette configuration (installation de Skype et lancement automatique de l'application avec le compte créé pour Jenkins) a été faite sur les comptes d'Erwan Deschamps et de Julien Siega sur le serveur afin que le minimum d'action soit à faire en cas de redémarrage du serveur.

Un test a été mené pour vérifier que Skype embarquait les identifications de ces conversations actives, afin d'être certain que le lancement du script sur une autre session engendrerait toujours une notification dans la même conversation, ce qui était le cas.

Un nouveau composant est mis en place dans le système d'information de Solutions Isonéo : Skype.

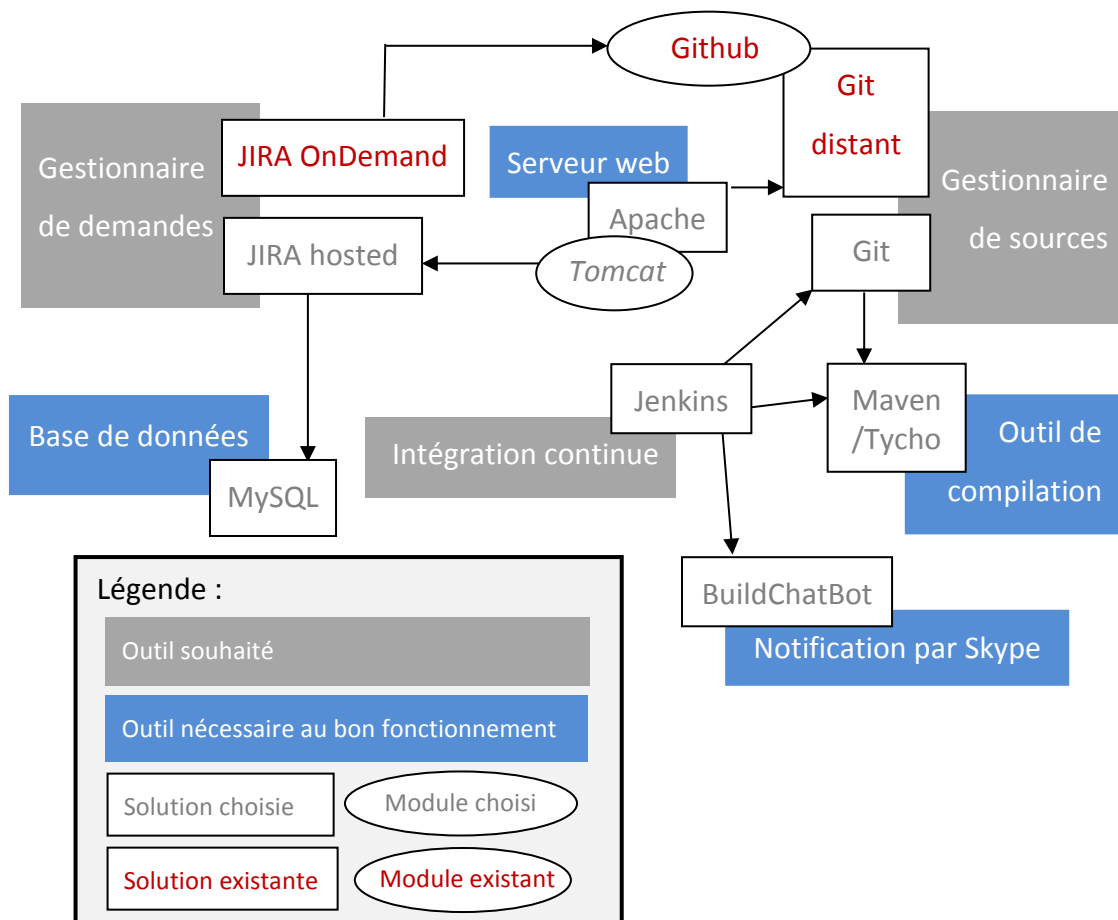


Figure 50 : Composants du S.I. en construction : mise en place des notifications Skype pour Jenkins

2.4.2. Appliquer le contrôle de la qualité du code

Julien Siega souhaitait récupérer les règles de codage, aussi communément nommées règles « checkstyle », en vigueur sur les projets Airbus d'Artal Technologies puis les adapter au projet Citrus. Les règles de codage sont des règles contrôlant la façon de coder comme la gestion des indentations, l'obligation d'avoir certaines clauses dans certaines méthodes ou encore d'ajouter des commentaires Javadoc. Certaines de ces règles seront présentées de façon plus précise par la suite. Pour les mettre en place, Julien Siega souhaitait intégrer « cobertura ». Cobertura est un outil qui permet de faire de la couverture de code mais ne permet pas de vérifier les règles de codage, de plus ses performances sont largement critiquées. Cependant la mise en place d'un outil de couverture de code était une bonne idée. Ce type d'outil permet de donner un

pourcentage de fonctionnalités vérifiées par des tests unitaires sur la totalité du projet. Dans mon précédent emploi, Jenkins avait été couplé avec un outil pour contrôler la qualité du code auquel il était possible de paramétrer la couverture de code et les règles de codage entre autres. J'ai donc proposé de mettre en place Sonar. Cet outil devait répondre également à une autre contrainte : la génération de rapport dans un format transportable. Sonar propose deux extractions, une au format csv, exploitable via excel, donnant par sous-projet le nombre de violations par caractéristique de règle (bloquante, majeure, mineure, etc.) et une au format PDF donnant sous forme plus esthétique les données principales du projet comme le nombre de lignes de code, le pourcentage de tests exécutés et de tests échoués ou encore le nombre de violations de règles de codage.

1	Name	Blocker violation	Critical violation	Rules compliance	Violations	Info violation	Minor violation	Major violation	Array Type
2	com.solutions.isoneo.citrus.Activator			0.0	34			34	C
3	com.solutions.isoneo.citrus.TestPremier			0.0	12			12	C
4	com.solutions.isoneo.citrus.builder.CitrusBuilder			0.0	195			195	C
5	com.solutions.isoneo.citrus.builder.CitrusNature			0.0	59			59	C
6	com.solutions.isoneo.citrus.builder.CitrusWorkspaceController			0.0	72			72	C
7	com.solutions.isoneo.citrus.builder.ToggleNatureAction			0.0	77			77	C
8	com.solutions.isoneo.citrus.builder.AboutHandler			0.0	13			13	C

Figure 51 : Extraction au format csv des métriques dans Sonar

Static Analysis

Lines of code

61,835

39 packages

764 classes

7,353 methods

3.7% duplicated lines

Comments

53.5%

71,078 comment lines

Complexity

3.3

31.8 /class

24,312 decision points

Dynamic Analysis

Code Coverage

0.0%

1 tests

Test Success

0.0%

1 failures

0 errors

Coding Rules Violations

Rules Compliance

0.0%

Violations

93,515



1.2. Violations Analysis

Most violated rules	
Indentation	52,182
Left Curly	9,332
Javadoc Method	7,246
Need Braces	6,018
Parameter Name	4,848

Figure 52 : Extraction PDF des métriques Sonar

Sonar impose une contrainte : le choix d'une base de données dans laquelle l'outil sauvegarde l'intégralité de l'historique et permet ainsi de donner une impression globale d'évolution sur le projet comme la diminution des violations.

Ayant déjà couplé Jira à MySQL, j'ai décidé de créer dans ce système une nouvelle base appelée sonardb avec ses propres noms d'utilisateur et mots de passe.

L'outil Sonar a donc été couplé à l'application Jenkins de Solutions Isonéo et configuré pour vérifier la couverture de code du projet Citrus ainsi qu'y contrôler le respect des règles de codage récupérées des projets Airbus d'Artal Technologies.

Pour réaliser le couplage entre Jenkins et Sonar, il a fallu ajouter le plug-in Sonar dans Jenkins et mettre en place la configuration suivante :

Sonar		
Installations de Sonar		
Nom	<input type="text" value="Sonar MySQL"/>	
Désactiver	<input type="checkbox"/>	
URL du serveur	<input type="text"/>	
Login du compte Sonar	<input type="text" value="sonar"/>	
Mot de passe du compte Sonar	<input type="password" value="*****"/>	
URL publique du serveur	<input type="text"/>	
URL de la base de données	<input type="text" value="jdbc:mysql://localhost:3307/sonar?autoReconnect=true"/>	
Nom d'utilisateur BDD	<input type="text" value="root"/>	
Mot de passe BDD	<input type="password" value="*****"/>	
Driver BDD	<input type="text" value="com.mysql.jdbc.Driver"/>	
Version du sonar-maven-plugin	<input type="text"/>	

Figure 53 : Pointage de la base de données Sonar depuis Jenkins

Afin qu'une analyse Sonar soit effectuée à chaque compilation lancée par Jenkins, chaque tâche a dû être enrichie par l'ajout d'une action à effectuer après la compilation.

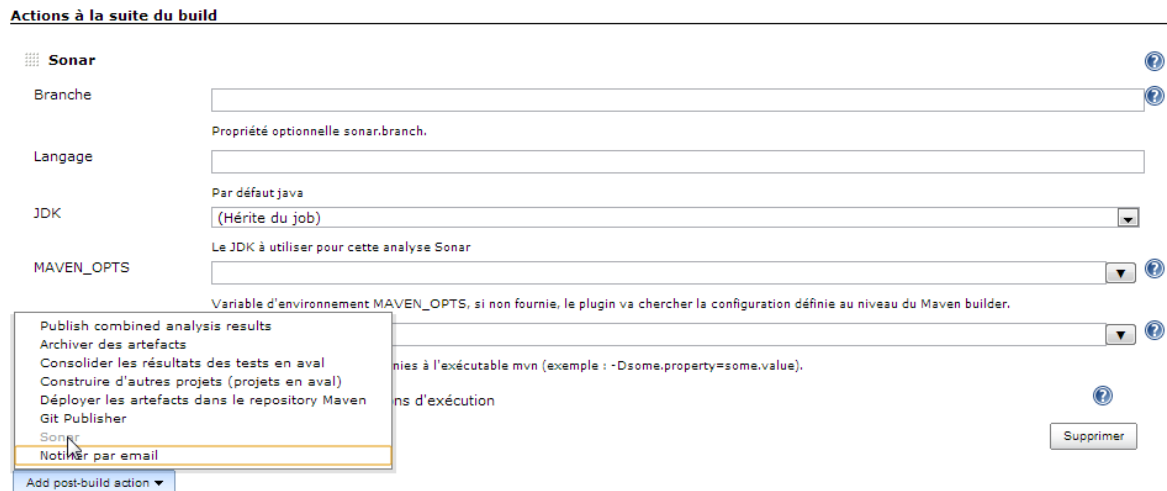


Figure 54 : Lancement d'une analyse Sonar après chaque compilation réalisée par Jenkins

Une fois les compilations effectuées, Sonar propose sur son tableau de bord un récapitulatif des éléments les plus importants.

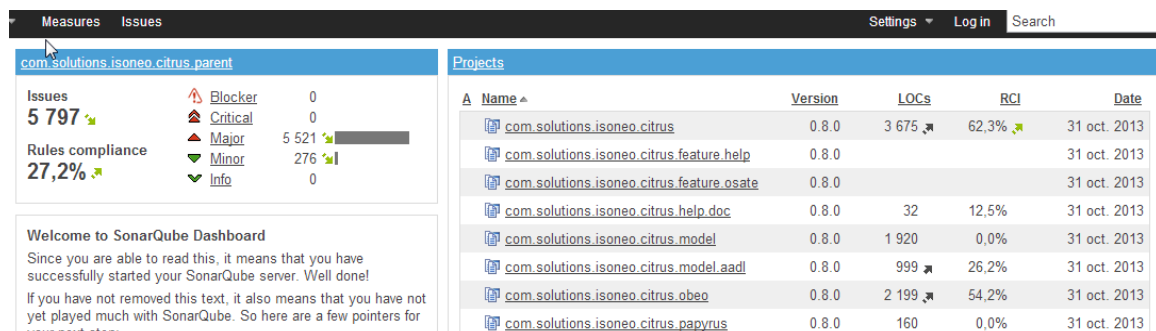


Figure 55 : Tableau de bord Sonar

Le nombre de violations des règles de codage était au départ très élevé. J'ai donc appliqué deux modifications principales à ces règles :

- ❖ Exclusions des sources générées
- ❖ Adaptation des règles à Citrus

En effet, certaines sources du projet Citrus sont générées automatiquement ou récupérées de plug-ins externes à Citrus ; ce code ne peut pas être comptabilisé dans les statistiques Sonar puisqu'elles ne font pas partie du code développé par l'équipe de Solutions Isonéo. Pour cela, Sonar propose dans son paramétrage d'exclure certains chemins de son analyse. J'ai donc ajouté des éléments comme `**/model/base/**` qui

excluent toutes les sources contenues dans `model/base`, qu'elles soient directement dans `base` ou dans un sous-dossier de `base`, sur n'importe quel sous-projet de Citrus. Le nombre de violations a diminué significativement.

Dans un autre temps, j'ai eu la charge d'adapter les règles de codage récupérées des projets Airbus d'Artal Technologies pour les adapter au projet Citrus. Pour cela j'ai mis en place le plug-in « Checkstyle » dans Eclipse en le configurant avec les règles récupérées et j'ai travaillé sur la vérification de chaque type de violation qu'Eclipse surligne en jaune.

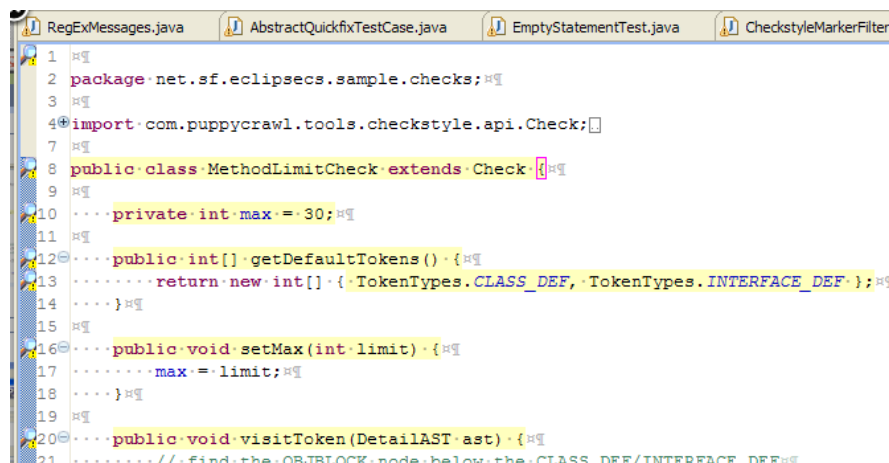


Figure 56 : Remontées des erreurs de style de code par Eclipse.

Eclipse possède un « Code template » et un « Formatter » qui permettent de générer automatiquement du code ou des commentaires Javadoc avec le raccourci clavier `Alt + Shift + J` ou de remettre en forme le code via le raccourci `Ctrl + Shift + F`. J'ai modifié également ces éléments pour que toute action automatique génère des sources en accord avec les règles de codage.

J'ai par exemple travaillé sur l'en-tête des classes afin que chacune d'entre elles soit signées du Copyright de Solutions Isonéo et contienne 4 informations importantes : l'auteur, l'année, le projet et le nom de la classe.

Les règles récupérées possédaient l'en-tête d'Artal Technologies que j'ai dû modifier. Le « Code template » d'Eclipse est également paramétré pour insérer automatiquement un en-tête à la création d'une classe java. Je l'ai donc modifié pour que l'en-tête soit la même que celle attendue par les règles de codage.

Solutions Isonéo a défini l'en-tête suivant qui a dû être ajouté sous format texte au « Code template » afin qu'il l'insère automatiquement à la création d'une classe Java.

```
/**
 *
 * SOLUTIONS ISONEO INC CONFIDENTIAL
 *
 * _____
 *
 * Copyright [2012] - [2013] Solutions ISONEO inc
 * All Rights Reserved.
 *
 * NOTICE: All information, intellectual and technical concepts
contained herein is, and remains
 * the property of Solutions ISONEO inc and its suppliers,
 * if any and are protected by trade secret or copyright law.
 * Dissemination of this information or reproduction of this material
 * is strictly forbidden unless prior written permission is obtained
 * from Solutions ISONEO inc.
 *
 * http://solutions-isonéo.com/
 *
 * Filename : ${file_name}
 * author : ${user}
 * year : ${year}
 * Project : CITRUS
 */
```

Figure 57 : En-tête obligatoire des classes du projet Citrus, inséré lors de la création

L'en-tête est ajouté dans les règles de codage sous la forme d'une expression régulière. Ce type d'expression permet de définir un « patron » afin que celui-ci soit interprété pour contrôler la forme de l'en-tête attendue.

Voilà l'expression régulière qui contrôle l'en-tête insérée ci-dessus :

```
^/\*$\n^ \* $\n^ \* SOLUTIONS ISONEO INC CONFIDENTIAL$\n^ \* \_{18}$\n^ \*
$\n^ \* Copyright \[20[1-9][0-9]\] \- \[20[1-9][0-9]\] Solutions ISONEO inc
$\n^ \* .*$\n^ \* $\n^ \* .*$\n^ \* .*$\n^ \* .*$\n^ \* .*$\n^ \* .*$\n^ \*
.*$\n^ \* $\n^ \* .*$\n^ \* $\n^ \* Filename \: .*$\n^ \* author \: .*$\n^ \*
year \: .*$\n^ \* Project \: CITRUS$\n^ \* \*/$
```

Figure 58 : Expression régulière contrôlant les éléments obligatoires de l'en-tête

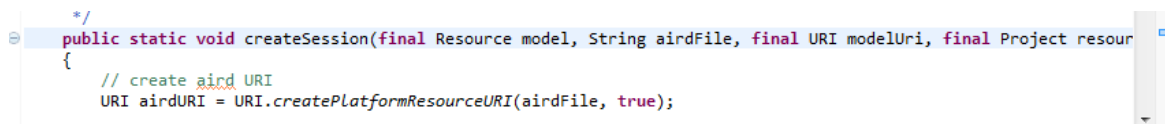
J'ai dû monter en compétences sur les règles des expressions régulières (Sitographie [10]). Par exemple :

```
/** est contrôlé par ^/\*$\n^
    Où ^ : la ligne commence par n'importe quel caractère
    Où \ : permet de ne pas interpréter les caractères spéciaux comme * qui
signifie 'n'importe quel caractère' s'il n'est pas précédé de \.
    Où $ : la ligne doit finir par ce qui le suit, soit \n^
    Où \n^ : retour à la ligne
```

De ce fait `* _{18}$\n^` contrôle que la ligne commence par une *, est suivie d'un espace puis de 18 underscores « _ » et se termine par un retour à la ligne ce qui permet à la ligne `* _____` de l'en-tête de correspondre au contrôle des règles de codage.

Une autre modification qui impactait non pas le « Code template » mais le « Formatter » d'Eclipse fût la gestion de l'indentation de la clause « Throws » pour lever une exception dans les méthodes Java.

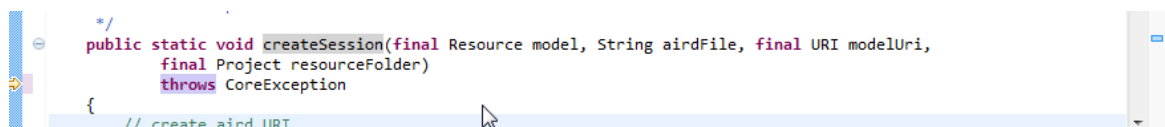
Une méthode Java qui prend en compte une exception possède dans sa déclaration la clause « throws », cependant celle-ci est souvent affichée sur une seule ligne qui oblige les développeurs à se déplacer horizontalement pour consulter la totalité de la déclaration.



```
*/
public static void createSession(final Resource model, String airdFile, final URI modelUri, final Project resour
{
    // create aird URI
    URI airdURI = URI.createPlatformResourceURI(airdFile, true);
```

Figure 59 : Méthode Java contenant une clause « throws » débordant de la fenêtre affichée

Après discussion avec les développeurs, nous souhaitons que la déclaration soit renvoyée à la ligne au niveau de la clause « throws » afin de faciliter la lecture de la classe Java.



```
*/
public static void createSession(final Resource model, String airdFile, final URI modelUri,
    final Project resourceFolder)
    throws CoreException
{
    // create aird URI
```

Figure 60 : Méthode Java contenant une clause « throws » après un formatage

Pour mettre en adéquation le formatage de la classe qui se fait automatiquement par Eclipse et les règles de codage qui prenaient déjà en compte ce renvoi à la ligne de la clause « throws », j'ai dû modifier le « Formatter » d'Eclipse.

Pour cela, j'ai dû rechercher dans la configuration du « Formatter » et modifier le type d'indentation de cette clause.

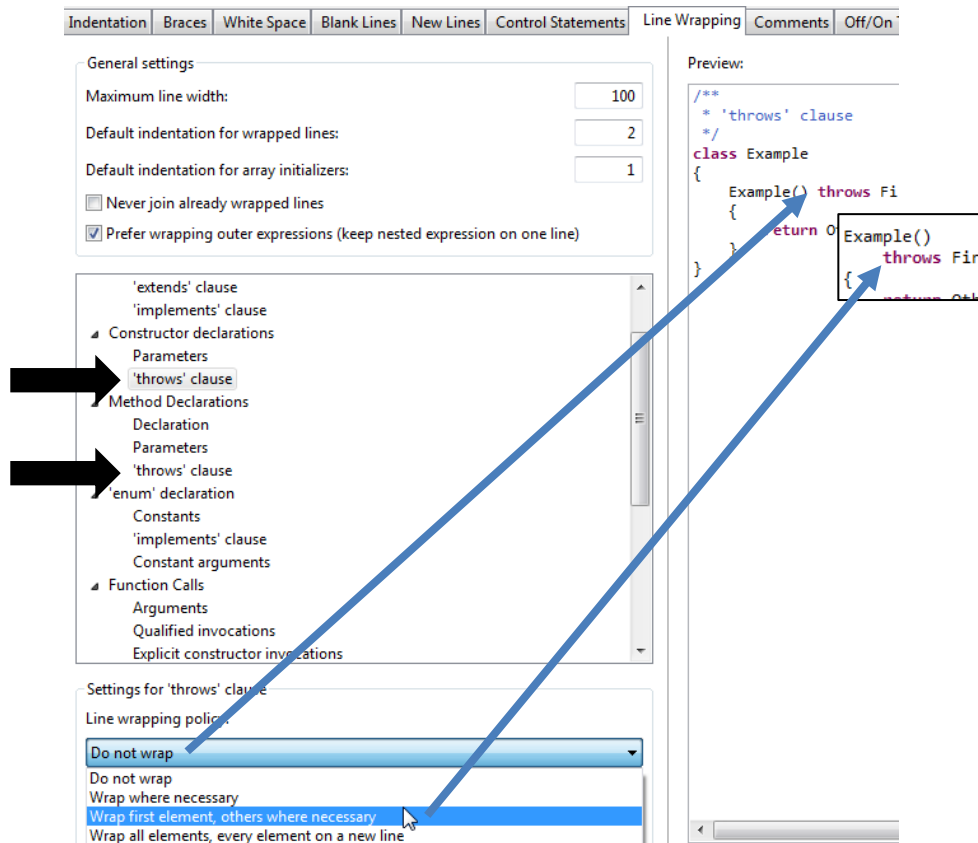


Figure 61 : Modification du « formatter » afin de renvoyer à la ligne la clause « throws » lors du Ctrl + Shift + F

Une fois que les règles de codage ont été totalement adaptées aux besoins des programmeurs du projet Citrus, ces règles ont été ajoutées dans Sonar comme règles par défaut. Sonar est désormais correctement configuré, couplé à Jenkins et son analyse se déclenche automatiquement à chaque compilation réalisée dans Jenkins.

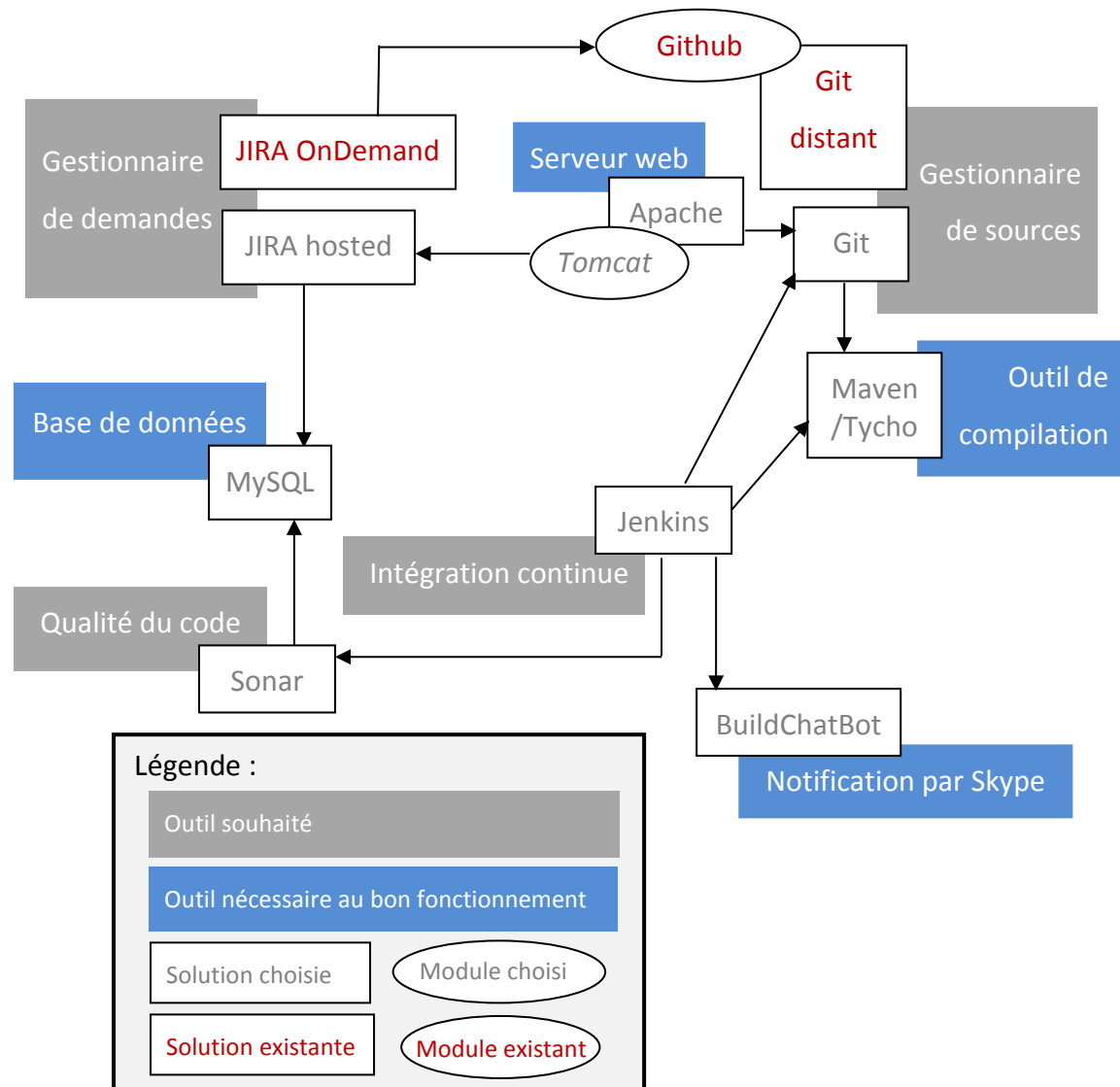


Figure 62 : Composants du S.I. en construction : mise en place du contrôle de la qualité avec Sonar

2.4.3. Centraliser les artefacts « Maven »

Le besoin de centralisation des artefacts Maven n'était pas initialement prévu, cependant un nouveau programmeur est arrivé chez Solutions Isonéo pour participer au développement d'un projet nommé « Asset Manager » en collaboration avec des membres de la société Artal Technologies. Ce projet est basé sur un projet existant chez Artal Technologies et le besoin de récupérer les éléments de configuration utiles à ce projet existant était nécessaire.

Pour compiler le projet existant nommé « Terea », Artal Technologies avait centralisé les librairies nécessaires dans un outil de centralisation d'artefacts « Maven » : Nexus (Sitographie [11]). De même, pour mettre à disposition les composants de « Terea »

nécessaires à la compilation du nouveau projet « Asset Manager », Artal Technologies déposait les librairies « Terea » sur ce même serveur Nexus. Le programmeur en poste dans les locaux de Solutions Isonéo avait besoin d'accéder à ce dépôt Nexus pour compiler le projet « Asset Manager » et poursuivre son développement. Cependant le réseau d'Artal Technologies est limité pour un accès depuis l'extérieur. Nous avons donc décidé de mettre en place un Nexus pour Solutions Isonéo, même si dans un premier temps il n'aurait d'utilité que pour le projet « Asset Manager » et non pas pour « Citrus » (Sitographie [12]).

Ce type d'outil permet d'accéder aux librairies même lorsque l'ordinateur n'a pas accès à Internet. En effet, le principe de la compilation Maven est d'aller récupérer toutes les librairies nécessaires sur Internet, cependant ce système peut diminuer considérablement les performances du réseau pour une société comme Artal Technologies qui compte une centaine de développeurs.

J'ai installé Nexus sur le serveur de Solutions Isonéo. Ce dernier n'étant pas accessible depuis l'extérieur de la pépinière d'entreprises dans laquelle se trouvait la société, nous avons décidé que l'accès au dépôt Nexus serait ouvert à tous. J'ai ensuite demandé aux programmeurs de Solutions Isonéo en charge du développement du projet « Asset Manager » de modifier le fichier « settings.xml » gérant la connexion au dépôt central Maven pour que la connexion s'établisse désormais vers le dépôt Nexus de la société. Internet est cependant nécessaire lorsque le projet requiert une nouvelle librairie qui n'est pas mise en place sur le dépôt central Nexus. Le poste du programmeur n'a toujours pas besoin d'Internet au contraire du serveur hébergeant le dépôt central Nexus.

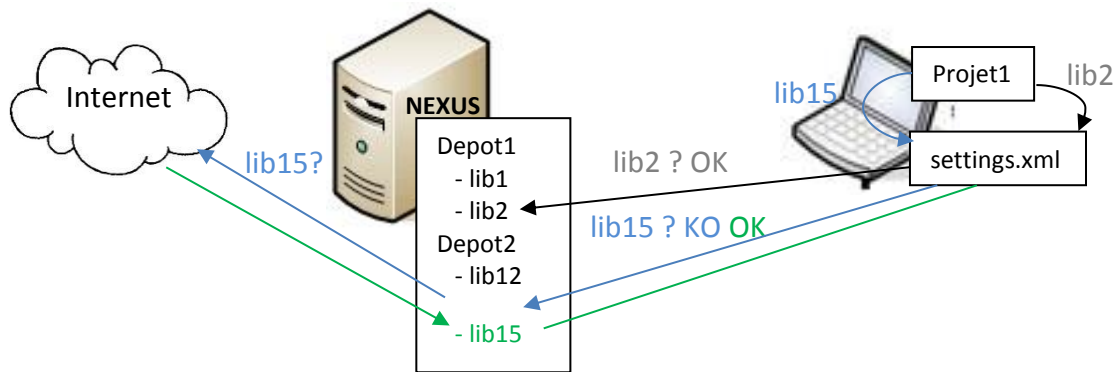


Figure 63 : Fonctionnement de Nexus

Plus précisément il est nécessaire que Nexus pointe vers des dépôts très utilisés comme le dépôt Maven central (Sitographie [13]) dont j'ai parlé plus haut. L'application Nexus, comme Maven, n'est pas capable de localiser les librairies quand elles ne sont pas disponibles sur le dépôt central Maven (<http://search.maven.org/#browse>), excepté si on lui fournit une localisation précise. De ce fait, si nous adaptions Nexus pour le projet Citrus, nous devrions rajouter dans les dépôts accessibles par Nexus la liste des « repositories » configurés dans le « pom.xml » du projet parent qui sont entre autres :

```
<repository>
  <id>obeo</id>
  <layout>p2</layout>
  <url>http://marketplace.obeonetwork.com/updates/od61/</url>
</repository>
```

Pour le projet « Terea », nous avons un problème : les librairies générées par la compilation du projet « Terea » nécessaires à la compilation du projet « Asset Manager » étaient stockées sur le dépôt Nexus d'Artal Technologies et non pas sur Internet ; de ce fait il n'était pas possible de les récupérer dans le dépôt central Nexus de Solutions Isonéo avec une configuration conventionnelle. Julien Siega ayant un compte sur le dépôt Nexus d'Artal Technologies, nous avons décidé de tenter un couplage entre le Nexus de Solutions Isonéo et celui d'Artal Technologies. Cette configuration n'a pas fonctionné. Après plusieurs examens de la situation, nous nous sommes rendu compte que le mot de passe saisi par Julien Siega était erroné. Cependant Nexus n'avait pas levé le problème car le compte de Julien Siega existe bien dans la base des utilisateurs du Nexus d'Artal Technologies. Tous les artefacts étaient donc téléchargés mais étaient tous

corrompus. Nous avons alors saisi le bon mot de passe, les nouvelles librairies à télécharger l'étaient correctement mais toutes les anciennes restaient corrompues. En effet Nexus vérifie l'arbre de ses dépendances et non pas l'état de ses librairies. Pour le logiciel, les dépendances étaient correctement téléchargées et utilisables par les programmeurs alors qu'elles étaient corrompues. J'ai donc vidé le dépôt physique lié à Nexus, situé dans « main-repo », pour relancer une compilation. Toutes les librairies ont été à nouveau téléchargées et exploitées avec succès. Au fur et à mesure que le projet « Asset Manager » grandissait, nous avons besoin de plus en plus d'éléments déposés sur le dépôt central Nexus d'Artal Technologies et les programmeurs ont alors rencontrés à nouveau le même problème sur un dépôt en particulier : toutes les dépendances d'un sous-projet de « Terea » ne pouvaient pas être récupérées et exploitées. Après plusieurs essais, je me suis mise en relation avec le responsable du projet « Terea » côté Artal Technologies. Le problème de connexion à ce dépôt provenait apparemment de la configuration côté Artal Technologies mais personne n'était capable de le résoudre. La personne avec laquelle j'étais en contact m'a informé qu'Artal Technologies possédait deux sites ayant chacun leur dépôt central Nexus et qu'aucun couplage n'avait été effectué entre ces deux entités car cela engendrait de trop gros problèmes de performance du réseau. J'ai donc interrompu toutes les connexions vers le dépôt Nexus d'Artal Technologies pour répliquer la configuration qu'ils avaient mis en place pour tous les dépôts externes à « Terea ». En effet le projet Terea comme « Citrus » utilise des composantes de plug-ins externes comme Obeo, OSGi, etc. Pour que le projet « Asset Manager » puisse utiliser les librairies propres au projet Terea, nous avons décidé de demander à Artal Technologies l'envoi de toutes les librairies pour les déposer sur le dépôt Nexus de Solutions Isonéo. Cette configuration a cependant un principal inconvénient : lorsque « Terea » subira des modifications majeures, le projet « Asset Manager » ne sera peut-être plus capable de compiler avec ces dernières évolutions car il utilisera toujours des anciennes librairies : celles qui ont été rapatriées sur le dépôt central Nexus de Solutions Isonéo. Les membres des projets « Terea » et « Asset Manager » ont été informés et devront se tenir mutuellement au courant des majeures évolutions pour en analyser les impacts.

Le gestionnaire d'artefacts « Maven » nommé Nexus est mis en place dans le nouveau

système d'information de Solutions Isonéo.

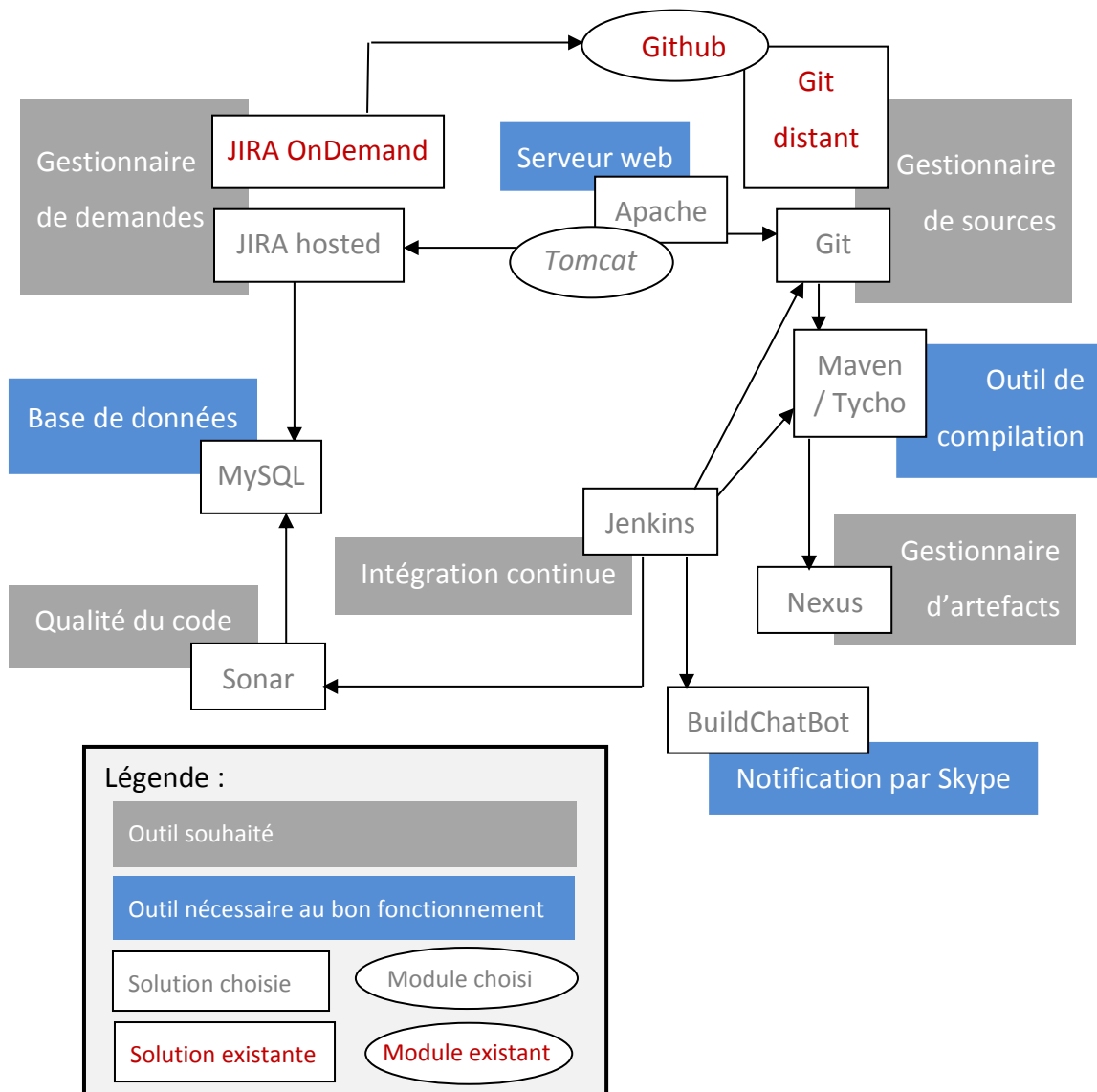


Figure 64 : Composants du S.I. en construction : mise en place de la centralisation des artefacts avec Nexus

2.5. Accompagnement aux utilisateurs

2.5.1. Centraliser la documentation

A mon arrivée, la société Solutions Isonéo possédait très peu de documentation. Mon arrivée faisant naître un nouveau projet interne à la société et le projet en charge de développer Citrus se déroulant en collaboration avec un partenaire, j'ai organisé plusieurs réunions pour que les membres de la société s'accordent sur la création d'un

référentiel documentaire. L'objectif étant de définir des lieux de stockage paraissant logique pour tous et facilement maintenables.

Nous avons donc décidé de séparer la documentation interne et celle concernant Citrus qui devaient être accessible aux salariés de CMC Electronique. Afin que la documentation ne soit pas simplement stockée sur nos machines, nous avons choisi dans un premier temps de la sauvegarder dans le gestionnaire de source : Git. Un nouveau dépôt a donc été créé pour la documentation interne : `solutionIsonéo/documentation`.

Dans un second temps j'ai dû définir les types de documents nécessaires à une capitalisation efficace. Pour cela j'ai travaillé avec Julien Siega qui connaissait les documents utilisés par les programmeurs et chefs de projet informatique d'Artal Technologies :

- ❖ Tool Procedure (TP) : Ce document doit correspondre à une procédure pour un outil. J'ai par exemple créé beaucoup de documents dans cette catégorie pour les manipulations possibles sous Jira. Le titre du document doit identifier clairement l'action qu'il décrit et l'outil qu'il concerne mais aussi sa catégorie. Ainsi toutes les « Tool Procedure » doivent désormais respecter le format : « TP – NomDeLOutil – Action ».
- ❖ User Reference Manual (URM) : Comme son nom l'indique ce document est à destination des utilisateurs et constitue le guide de référence d'un outil. Il permet aux utilisateurs de l'outil une fois développé mais aussi aux partenaires tout au long de l'avancement du projet de le consulter pour parcourir les principales fonctionnalités et leur utilisation. Comme ce document pourra être consulté et récupéré par les partenaires et clients, nous avons décidé d'insérer également les initiales de la société : « InitialesDeLaSociété – NomDuProjet – URM – Version.Edition ». Le numéro d'édition doit être incrémenté à chaque fin de Sprint si le document a été livré et approuvé. Le numéro d'édition augmente entre deux livraisons pour tracer qu'il a subi un certains nombres de mises à jour.

Ce document a engendré plus de désaccord entre les membres de Solutions Isonéo que ceux de la catégorie « Tool Procedure ». Etant le guide de référence d'un outil, il ne doit exister qu'un seul « URM » par projet. Cependant à mon arrivée et dès le début de la construction du système d'information s'est posé la question suivante : « Les composantes du système d'information mise en place pour le projet Citrus et pour répondre aux conditions du partenaire CMC Electronique font-elles partis de Citrus ou de Solutions Isonéo en tant que projet interne ? ». Afin de prendre en considération le point de vue de notre partenaire, nous avons décidé d'avoir deux URM pour le projet Citrus afin que le système d'information dans lequel évolue l'outil soit clair et approuvé par CMC Electronique. Le titre a donc été enrichi de la mention [IT] pour « Information Technology ». Chez Solutions Isonéo il peut donc exister deux URM par projet. L'URM[IT] est à destination des nouveaux programmeurs du projet Citrus. Ils peuvent ainsi facilement et rapidement prendre en main l'environnement de développement, l'architecture du projet et les vérifications obligatoires avant de déposer le code (compilation Maven et vérification de la construction du produit).

Il est important que les nouveaux membres de la société puissent consulter ces documents et avoir une vue d'ensemble des catégories et de leur utilisation. J'ai donc décidé d'organiser le dépôt Git `solutionIsonéo/documentation` suivant les dossiers : « Tool Procedure », « User Reference Manual » mais aussi « Models » pour héberger les « templates » de chaque catégorie de document, respectant un certain formalisme ainsi que la charte graphique de la société.

Un document a été créé également dans ce dernier dossier afin d'expliquer quelles sont les catégories existantes et leur utilisation.

Ainsi il était facile pour un nouvel employé de prendre en main rapidement la gestion documentaire de Solution Isonéo. De plus, cette organisation simplifie également l'ajout de documents et leurs contrôles. En effet il est simple de voir qu'un nouveau dossier a été créé dans la racine mais que son modèle et l'explication de son utilisation n'est pas faite dans le dossier « Models ». Git permettant d'identifier la personne ayant réalisé la

création ou la modification, il est facile de demander à la personne concernée de compléter correctement l'environnement documentaire.

2.5.2. Diffuser l'information

Les deux projets « Asset Manager » et « Citrus » prenant de plus en plus d'ampleur, certains programmeurs d'Artal Technologies sont intervenus.

Chaque programmeur intervient avec son vécu et une vision du fonctionnement des projets façonnée par ses précédents emplois. Les échanges entre les membres de Solutions Isonéo sont facilités par la proximité et la disponibilité des gens qui peuvent répondre rapidement à une question orale alors que répondre à un mail est une action plus lourde.

Pour apporter aux programmeurs d'Artal Technologies une vision du fonctionnement des projets, je leur ai adressé l'URM[IT] de Citrus expliquant le système d'information en construction et les impacts de leur développement : problème sur la compilation lancée par Jenkins, fonctionnement de Git qui permet de réaliser un « commit » pour conserver son code sans le déposer et éviter de ralentir la totalité du projet ou encore l'architecture du projet avec l'existence d'un projet parent. Les programmeurs côté Solutions Isonéo étant très peu disponibles avec la mise en place de nombreuses réunions avec le partenaire, j'ai visé à créer dans le mail, ainsi adressé à l'équipe toulousaine, une union permettant aux programmeurs d'Artal Technologies de se sentir impliqués dans l'équipe comme s'ils étaient dans les mêmes locaux. Je me suis également mis à disposition de ceux-ci pour pouvoir faire relais et ainsi comprendre les problématiques d'un programmeur lors de son arrivée sur un nouveau projet, les attentes qu'il peut avoir d'un système d'information et confronter cette vision avec la mienne, celle d'un chargé de système d'information depuis 5 années.

Pour ne pas que cette apprentissage soit trop lourd et soit au final bâclé par les programmeurs, j'ai envoyé plusieurs mails contenant les règles les plus importantes ou des rappels concernant l'organisation des sources dans le projet lorsque cela était mal fait.

J'ai également eu un travail de diffusion d'information à faire lorsque j'ai mis en place les règles de codage. Il était incohérent de demander à un programmeur de corriger

l'intégralité du projet Citrus pour l'adapter à ces règles. De plus celles-ci avaient été récupérées d'Artal Technologies et adaptées en fonction de certaines sources prises au hasard sur le projet, certaines adaptations étaient donc peut-être encore à faire. J'ai créé une procédure, la plus simplifiée possible pour que chaque programmeur puisse mettre en place les règles de codage sur son Eclipse. J'ai ensuite demandé à chacun des programmeurs de me faire des retours sur les règles en vigueur afin que nous puissions discuter des dernières adaptations à établir. A chaque nouvelle adaptation je transmettais un mail à chaque membre du projet pour souligner les modifications établies et demander ainsi le chargement des nouvelles règles dans leur propre espace de travail. De cette façon, tous les sous-projets sur lesquels travaillaient les différents programmeurs se voyaient adaptés au fur et à mesure aux règles de codage. Voyant que de moins en moins de retours m'étaient faits, j'ai alors envoyé un mail en demandant aux programmeurs qui ne l'auraient pas fait d'appliquer les règles de codage en insistant sur le fait que le nombre de violations de ces règles seraient vérifiées au prochain audit de Citrus.

Durant la mise en place des règles de codage sur le projet Citrus, le projet « Asset Manager » se mettait en place. La procédure a été envoyée en même temps aux membres de ce projet qu'aux membres du projet Citrus afin qu'ils puissent naturellement se poser la question des règles à utiliser. De ce fait, les programmeurs d'« Asset Manager » ont pu utiliser les règles de codage de « Terea » dès le lancement du projet et l'adaptation s'est ainsi faite naturellement.

La diffusion de l'information passe également par un vocabulaire commun et la prise en compte des différents lecteurs du document. J'ai ainsi réadapté l'URM[IT] complété entre temps par des programmeurs afin d'agrandir les impressions écrans ajoutées et de préciser certains éléments car le document peut être lu par un programmeur ayant déjà quelques années d'expérience en développement Java mais il peut également être lu par un nouvel arrivant tout juste diplômé et n'ayant encore jamais travaillé sous Eclipse.

La mise à niveau de ce document a pu être vérifiée par Erwan Deschamps - plutôt axé aujourd'hui dans le commercial - en charge d'adapter ponctuellement l'URM de Citrus au format xml en vue de pouvoir être consultée depuis Eclipse.

2.5.3. Sécuriser l'information

Les gestionnaires de sources comme Git considèrent les documents écrits comme Word ou excel comme des fichiers binaires, il n'est donc pas capable de les « fusionner » si deux personnes ont travaillé en même temps sur le même élément. L'outil risque donc d'écraser le travail d'une personne avec celui d'une autre.

Des logiciels dits de « GED » (Gestion électronique de documentation) sont capables de gérer les modifications sur un document écrit comme le gestionnaire de versions le fait avec des sources. De plus il est possible de sécuriser ces documents et d'établir des droits différents à chacun des membres. Solutions Isonéo pourrait avoir accès à l'intégralité des documents alors que les membres de CMC Electronique n'auraient accès qu'aux documents concernant Citrus.

De nombreux outils permettent ce type de gestion comme SharePoint proposé par Microsoft ou deux concurrents du monde OpenSource : Nuxéo et Alfresco. D'après les retours que j'ai pu obtenir de personnes travaillant avec ce type d'outil, Nuxéo est le moins coûteux et le plus simple à mettre en place, cependant n'étant pas la priorité de Solutions Isonéo nous ne sommes pas allés jusqu'à la mise en place de cette solution.

2.6. Gestionnaire d'authentification

Solutions Isonéo étant une société naissante, aucun annuaire n'existait à mon arrivée. Les solutions installées se multipliant, j'ai proposé de mettre en place un annuaire très simple afin de réduire le nombre de mots de passe par utilisateur. Ainsi chaque salarié a saisi un mot de passe dans la base de données de l'annuaire et peut se servir du même sur toutes les applications de la société.

2.6.1. Mettre en place un annuaire

Beaucoup d'entreprises de taille moyenne et importante possèdent un annuaire et utilisent le protocole LDAP (Lightweight Directory Access Protocol) pour le mettre en place. J'ai également utilisé cette solution (Sitographie [14]).

Tout annuaire LDAP est organisé comme un arbre possédant une ou plusieurs entrées, aussi appelé index, qui permet ensuite d'accéder à plusieurs ensembles clés / valeurs.

L'annuaire mis en place pour Solutions Isonéo possède un seul index nommé `isoneo` qui permet ensuite d'accéder à tous les utilisateurs identifiés par un nom d'utilisateur unique comme `ebardaji`.

Il est possible d'ajouter une interface web pour gérer le LDAP appelée `phpldapadmin`, cependant cette mise en place n'étant pas prioritaire, la tâche a été créée dans Jira mais elle n'a pas été réalisée avant mon départ.

La gestion de l'annuaire reste donc pour l'instant administrable uniquement en ligne de commande chez Solutions Isonéo. Lors de la création, j'ai ajouté chacun des salariés de la société via la commande `adduser nom_d_utilisateur` et leur ai demandé de venir saisir leur mot de passe.

Une commande existe aussi pour voir tous les ensembles présents pour une entrée dans l'annuaire : `ldapsearch -x -b 'dc=isoneo' '(objectclass=*)'` ou pour consulter les détails d'un ensemble : `ldapsearch -x -b 'dc=isoneo' 'cn=ebardaji'`.

Un document a été créé pour que les membres de Solutions Isonéo puissent maintenir cet annuaire après mon départ.

2.6.2. Coupler l'annuaire avec les applications

L'objectif de la mise en place précédente fut de coupler cet annuaire aux applications déjà en place afin que les utilisateurs ne possèdent qu'un seul mot de passe pour toutes les applications.

a) Jenkins

Grâce à un plug-in ajouté à la plateforme Jenkins j'ai pu coupler l'application au LDAP. La configuration requiert de pointer vers le serveur nommé `Walle.isoneo`, d'insérer le nom d'utilisateur et le mot de passe du compte d'administration du LDAP pour autoriser la consultation de celui-ci – dans notre cas ce compte est « admin » - puis de fournir

l'entrée du LDAP (isoneo) ainsi que le filtre pour rechercher les éléments qui se trouvent associés à l'entrée – dans notre cas, l'intégralité des éléments : `uid={0}`.

☒ LDAP

Serveur

DN racine
☐ Autoriser un rootDN vide

Base pour la recherche d'utilisateurs

Filtre pour la recherche d'utilisateurs

Base pour la recherche de groupes

Group search filter

Group membership filter

DN du gestionnaire

Mot de passe du gestionnaire

Disable Ldap Email Resolver ☐

Figure 65 : Configuration du LDAP dans Jenkins

L'attribution des droits se fait ensuite dans une matrice. Si l'utilisateur ajouté n'est pas dans l'annuaire LDAP alors celui-ci apparaîtra précédé d'une icône de sens interdit et l'utilisateur ne pourra pas se connecter à l'application.

☒ Sécurité basée sur une matrice

Utilisateur/groupe	Global				
	Administer	Read	RunScripts	UploadPlugins	Config
 ebardaji	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
 jenkins	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
 jlasalle	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 jsiega	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Anonyme	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 NonLDAP	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Utilisateur/groupe à ajouter:

Figure 66 : Ajout d'un utilisateur dans Jenkins inconnu du LDAP

b) Jira

Jira comme Jenkins possède une interface graphique pour configurer cette connexion, les mêmes données sont saisies pour accéder au contenu de l'annuaire.

Server Settings

Name: * LDAP server

Directory Type: * OpenLDAP
Making a selection will automatically enter default values

Hostname: * Walle.isoneo
Hostname of the server running LDAP. Example: ldap.example.com

Port: * 389 ☐ Use SSL

Username: cn=admin,dc=isono
User to log in to LDAP. Examples: user@domain.name or uid=admin,ou=users,dc=example,dc=com

Password:

LDAP Schema

Base DN: dn=isono
Root node in LDAP from which to search for users and groups

Additional User DN: uid={0}
Optional. If specified, the user will be searched for in this DN as well as the Base DN.

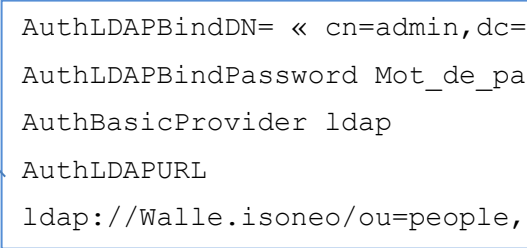
Figure 67 : Configuration du LDAP dans Jira

c) Git

Le couplage entre Git et LDAP ne se fait pas de façon graphique et est donc plus difficile à prendre en main.

Comme vu précédemment Git utilise un serveur apache pour pouvoir être accessible via le protocole http. Pour cela il utilise un fichier GIT.conf positionné dans la configuration d'Apache. La configuration précédemment mise en place doit être modifiée afin que Git puisse consulter le contenu du LDAP via le compte « admin » : l'utilisation du fichier « passwd » contenant les mots de passe cryptés est remplacé par plusieurs paramètres permettant d'interroger l'annuaire LDAP.

D'autres modifications ont lieu notamment sur les paramètres « AllowOverride » et « Require all granted » que la configuration LDAP surcharge.

```
Alias /git/ « /home/GIT/repositories/ »
<Directory /home/GIT/repositories>
    DAV on
    Options Indexes FollowSymlinks MultiViews
    AllowOverride None -> AllowOverride AuthConfig
    Require all granted
    AuthType Basic
    AuthName "Git"
    AuthUserFile /home/GIT/passwd
    
    AuthLDAPBindDN= « cn=admin,dc=isoneo »
    AuthLDAPBindPassword Mot_de_passe_admin
    AuthBasicProvider ldap
    AuthLDAPURL
    ldap://Walle.isoneo/ou=people,dc=isoneo.uid
    Require valid-user
</Directory>
```

est remplacé par

Après plusieurs essais cette configuration engendrait l'erreur « auth.failed » informant que le compte saisi n'avait pas les droits pour accéder à l'application. Après quelques recherches, le module `mod_authnz_ldap.so` permet d'ajouter le couplage entre Apache et LDAP. Je rencontrais le même problème que précédemment : un lien était manquant entre les modules autorisés et les modules disponibles d'Apache. Après création du lien via la commande `ln -s /etc/apache2/mod-available/authnz_ldap.load /etc/apache2/mod-enabled/authnz_ldap.load` la connexion à Git se faisait via les comptes disponibles dans l'annuaire.

d) Sonar et Nexus

Les applications Sonar et Nexus possèdent des droits en consultation ouverts à tous. Seul un compte d'administration est créé pour le paramétrage de l'application. Aucune connexion au LDAP n'était donc nécessaire.

LDAP gère désormais les identifications nécessaires pour toutes les applications du système d'information de Solutions Isonéo.

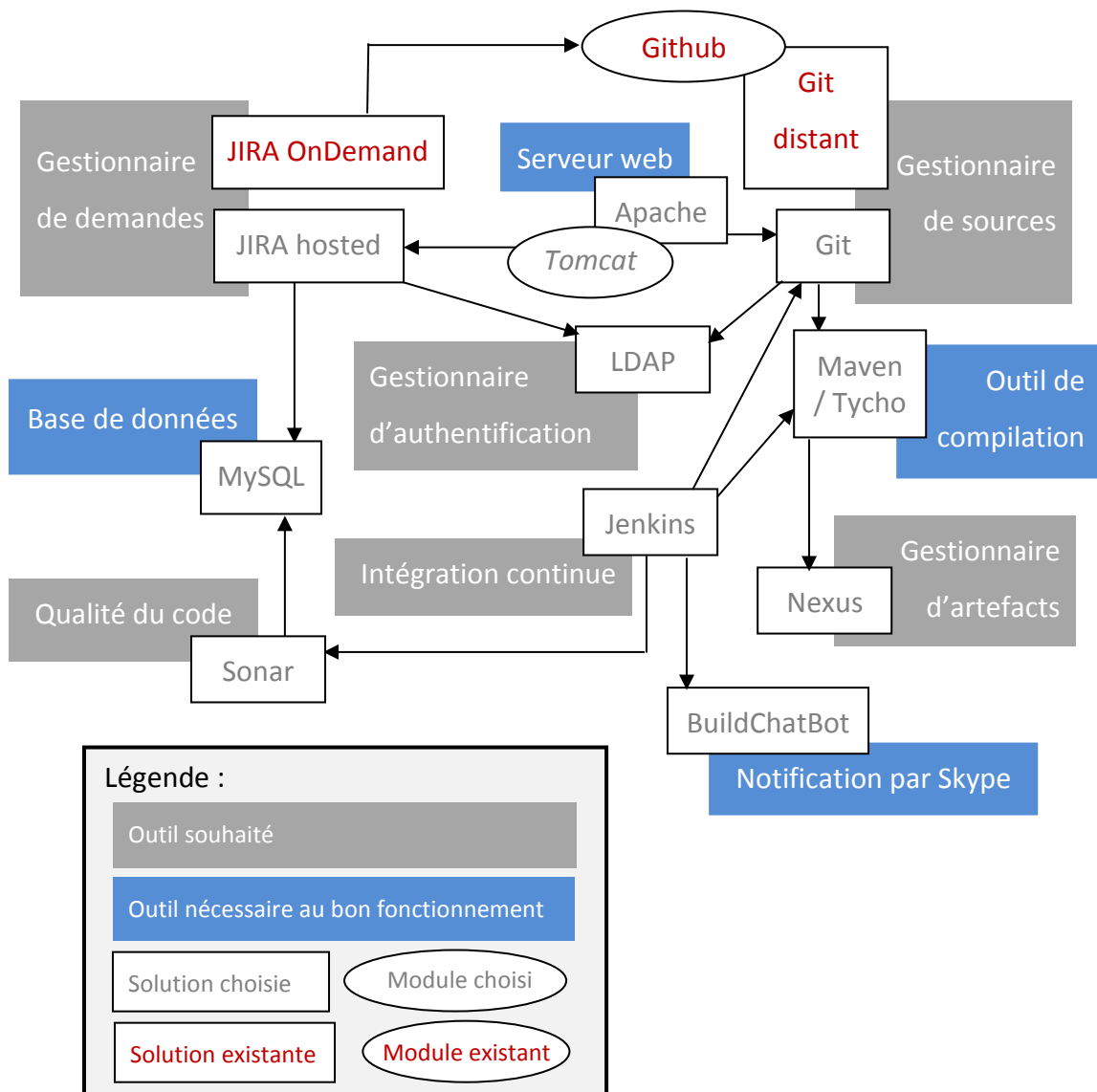


Figure 68 : Composants du S.I. en construction : mise en place de la gestion des authentifications avec un LDAP

2.7. Serveur

Mon travail sur le serveur a pris du retard car Solutions Isonéo étant une société naissante, elle possède de fortes contraintes de budget. Ce dernier doit de plus être contrôlé par les deux personnes membres de la société mère ce qui impacte considérablement le temps de traitement.

Toutes les applications présentées ci-dessus ont d'abord été testées et paramétrées sur mon poste de travail fonctionnant sous Windows avant l'obtention du serveur à la fin du

mois de mai et avec lui, la possibilité d'implémenter le système d'information de façon durable.

2.7.1. Centraliser le système d'information

a) Configuration matérielle du serveur et choix d'un système d'exploitation

J'ai participé au choix du serveur. Il devait posséder un mécanisme de réplication du disque dur principal. Ce procédé est appelé RAID. Il devait également posséder beaucoup de mémoires car les applications Java demandent beaucoup de performance de la part de la machine. Et pour finir il devait posséder une grande capacité de stockage, choisie à 1To soit 1000Go pour pouvoir l'utiliser sereinement pendant plusieurs années.

Grâce à cet achat, je me suis également rendue compte des problématiques que pouvaient rencontrer une « start-up » ainsi que la réalité du coût de tels investissements. Ces aspects étaient pour moi inexistantes ayant travaillé dans des sociétés de services faisant partis de grands groupes et comptant au minimum 150 employés.

Ce serveur a été demandé sans système d'exploitation afin de réduire le coût. Cela engendrait une forte montée en compétences pour ma part car je devais me rapprocher du métier d'administrateur système que je n'avais jamais entrevu.

Comme la plupart des serveurs, nous nous étions orientés dès le départ vers une solution Linux (Bibliographie [5]) pour des questions de fiabilité, sécurité, simplicité, efficacité et stabilité. En effet, même si au premier abord la prise en main d'un système Linux paraît déconcertante quand on est habituée à Windows, une fois que le système est maîtrisé, on peut remarquer que la communauté est beaucoup plus active et que le système est beaucoup plus simple et léger que Windows.

J'ai donc recherché un **système d'exploitation** pour le serveur. Le kernel linux est le cœur de tous les systèmes d'exploitation basés sur cette technologie comme Redhat, Debian, CentOS, etc. Ce cœur englobe des outils développés sur GNU qui est historiquement la racine du kernel Linux et des applications comme bind, Apache, proftpd, etc. Cette base est ensuite enrichie par les différents distributeurs mais est très

rarement modifiée : tous les systèmes d'exploitation basés sur Linux possèdent donc la même architecture système et les mêmes outils de base. La différence se situe au niveau de l'interface graphique, plus ou moins adaptée à un utilisateur novice ou expérimenté et à un système de configuration différent bien que celui-ci pilote les mêmes comportements du système d'exploitation.

Mes recherches se sont donc organisées autour de deux points :

- ❖ Un système d'exploitation adapté au monde professionnel et à faible coût
- ❖ Un système d'exploitation compatible avec l'intégralité des outils du système d'information à mettre en place

Il existe 5 principales distributions basées sur Linux : Redhat, CentOS, Fedora, Ubuntu et Debian.

Initialement Redhat distribuait deux versions de son système d'exploitation : une « entreprise » pour une utilisation professionnelle et une pour une utilisation de particulier. A partir de Redhat Linux 9, la société qui l'édite a décidé de déléguer la version pour les particuliers à la société Fedora et ne propose désormais que sa version professionnelle pour laquelle le support est très coûteux. Cette solution a donc été écartée.

CentOS récupère quasiment le contenu de la version professionnelle distribuée par Redhat pour la recompiler et l'estampiller « CentOS ». Son support est à moindre coût mais très peu développé.

Fedora, comme dit précédemment, est plutôt mis à disposition des particuliers, tout comme Ubuntu.

La dernière solution était Debian : une distribution très déployée dans le monde professionnel comme particulier, dont la communauté est très active et dont ni le support, ni la distribution ne sont payants.

Avant d'opter pour une distribution, j'ai dû vérifier la compatibilité de chacune d'entre elles avec les solutions composant le système d'information (Sitographie [15] pour l'étude de Jira). Le tableau récapitulatif suivant ainsi que les avantages et inconvénients de chacune des distributions ont été présentés aux membres de Solutions Isonéo.

	DEBIAN	REDHAT	CENTOS	FEDORA
JENKINS	X	X	X	X
JIRA	Plus difficile mais faisable	X	X	?
SONAR	X	X	X	X
ECLIPSE JUNO	7.0.0	RHEL 6.4 ou 5.9	X	X
GIT	X	?	X	X
JAVA	X	RHEL 6.4 ou 5.9	X	?
MAVEN3	X	?	X	?
ALFRESCO	X	?	X	?
NUXEO	X	X	?	?

Tableau I : *Compatibilité des solutions composant le S.I. avec les systèmes d'exploitation basés sur Linux.*

Pour des sociétés ayant des besoins très spécifiques et des outils demandant beaucoup de performance il est logique de payer le support auprès de Redhat qui se consacre au monde professionnel. Cependant Solutions Isonéo ayant de fortes contraintes de budget ainsi qu'un important potentiel d'évolution mais pas forcément des besoins spécifiques à demander au système d'exploitation du serveur, nous avons choisi de déployer Debian version 7.0.0 nommée Wheezy sur le serveur avec des données cryptées pour plus de sécurité.

Cette mise en place a cependant demandé une montée en compétences générale de ma part sur le système d'exploitation Debian. Pour cela, j'ai dans un premier temps consulté le guide de la version précédente de Debian : Wheezy (Bibliographie [6]).

b) Maîtrise du système d'organisation de Debian

Une fois l'installation faite (Sitographie [16]), j'ai dû appréhender l'organisation du système (Bibliographie [7]).

Toutes les distributions linux proposent la même organisation des données qui est très différente de celle proposée par Windows. Un dossier `home` contient les données de tous les utilisateurs possédant une session pour se connecter au serveur. Le serveur Solutions Isonéo a été initialisé avec 4 comptes : ebardaji, jsiega, edeschamps et user afin que les

membres importants de la société possèdent leur propre compte, qu'une session permette la mise en place du système d'information (ebardaji) et qu'un utilisateur qui peut être assimilé au compte « invité » de la distribution de Microsoft permette de se connecter en ayant des droits très restreints sur la plateforme. Ce dernier compte est également une solution dans le cas où Erwan Deschamps ou Julien Siega oublieraient leur mot de passe, afin de pouvoir se connecter au système et de pouvoir accéder au compte « root » pour faire des modifications. En effet les distributions Linux proposent un compte « root » accessible via un terminal depuis n'importe quelle session ouverte sur la machine. Evidemment ce compte « root » possède un mot de passe.

En suivant de nombreuses recommandations (Sitographie [17] et Bibliographie [8]), chaque application composant le système d'information s'est vu créer un dossier dans le répertoire `home`.

Le système possède ensuite à sa racine 5 autres dossiers principaux qui possèdent chacun une partition dédiée :

- ❖ root dont le chemin correspond à la racine, soit « / », pour toutes les applications installées par l'administrateur système (soit quasiment toutes),
- ❖ tmp dont le chemin est `/tmp`, correspondant à un dossier dans lequel va se générer des fichiers temporaires liés aux applications,
- ❖ usr dont le chemin est `/usr`, pour toutes les actions accessibles aux utilisateurs autres que l'administrateur système
- ❖ var dont le chemin est `/var`, pour toutes les variables utilisées par le système
- ❖ boot dont le chemin est `/boot`, pour le démarrage du système

Ayant très peu de connaissances dans le système Debian et le logiciel d'installation proposant de prendre en main automatiquement le partitionnement du disque, je l'ai utilisé. Cependant celles-ci ont été dimensionnées de façon non optimale et afin de les redimensionner correctement j'ai consulté le pourcentage d'occupation de chacune d'entre elles :

- ❖ root initialisée à 40M dont 87% était déjà utilisé
- ❖ home ayant 873G d'espace dont seulement 1% était utilisé
- ❖ var allouée à un espace de 187M dont 93% était déjà occupé
- ❖ usr associée à un espace de 3.4G utilisé à 57%

- ❖ tmp ayant un espace de 339M occupé à 1%
- ❖ boot ayant 175M d'espace utilisé à hauteur de 20%

Il est normal que la partition associée au `home` soit la plus grande, cependant les autres espaces sont sous-dimensionnés. Après plusieurs recherches (Sitographie [18] et [19]) et prises de conseils auprès d'administrateurs système, je souhaitais redimensionner les partitions de la façon suivante :

- ❖ root : 30G
- ❖ var : 10G
- ❖ usr : 10G
- ❖ tmp : 5G qui peut paraître surdimensionné, cependant le serveur hébergeant un grand nombre d'applications Java, celles-ci génèrent souvent des fichiers temporaires.
- ❖ boot qu'il m'était impossible de redimensionner sans une réinstallation complète du système ou une grosse prise de risques sur des actions complexes d'administration. De plus les besoins du `boot` grandissent rarement et n'occupe actuellement que 20% de son espace.
- ❖ home : 820G permettant ainsi de conserver de la place non allouée qui peut être octroyée à une partition au besoin.

Pour repartitionner le disque j'ai utilisé le logiciel `sysresccd` que j'ai gravé sur un CD pour démarrer à partir de celui-ci lors de l'initialisation du bios. Le logiciel exécute ensuite les instructions passées en ligne de commande. Par exemple pour la partition du système de fichiers nommé `/tmp` dont le volume logique s'identifie par `/dev/mapper/Wallee-tmp` :

```
umount /tmp
```

Démonte le système de fichiers afin de s'assurer qu'il ne sera pas modifié durant la manipulation

```
e2fsck -f /dev/mapper/Wallee-tmp
```

Vérifie l'intégrité du volume

```
lvextend -size +5G /dev/mapper/Wallee-tmp
```

Ajoute 5Go au volume logique Wallee-tmp

```
resize2fs -p /dev/mapper/Wallee-tmp
```

Force le système de fichiers à s'étendre pour utiliser la place attribuée précédemment

```
e2fsck -f /dev/mapper/Wallee-tmp
```

Vérifie l'intégrité du volume après redimensionnement

```
mount /tmp
```

Remonte le système de fichiers afin qu'il soit à nouveau accessible par le système

Cette manipulation a été utilisée pour chacune des partitions à redimensionner afin d'obtenir des espaces cohérents.

c) Organisation du réseau

Une fois le système d'organisation de Debian pris en main, Solutions Isonéo devait attribuer une place pour son serveur dans le réseau.

La société se trouve dans une pépinière d'entreprises dont l'organisation réseau a été décrite dans la première partie de ce mémoire. Le serveur n'étant pas doté d'une carte wifi, nous avons décidé que contrairement aux postes de travail, le serveur serait branché de façon filaire au réseau Internet.

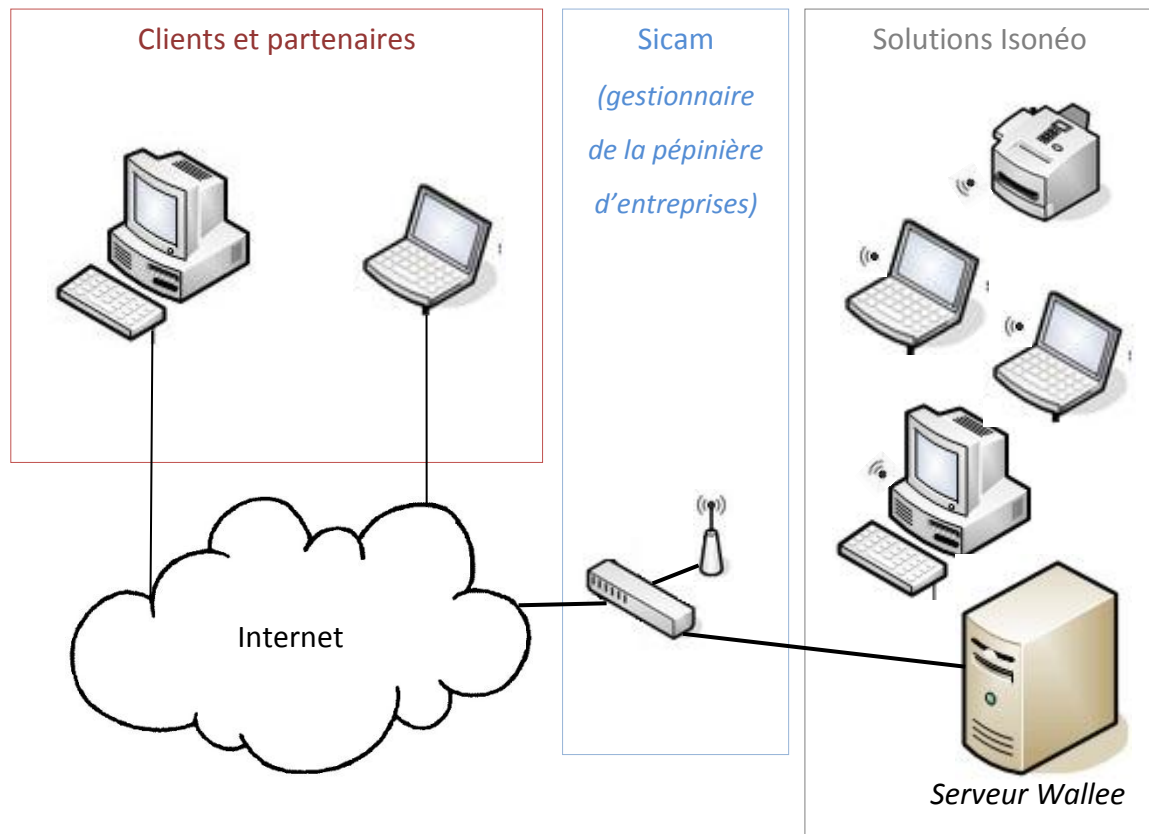


Figure 69 : Schéma du réseau exploité par Solutions Isonéo avec son serveur

Le routeur wifi permettant l'accès au réseau Internet est administré par la Sicam. Tous les postes de Solutions Isonéo se trouvant sur le même sous réseau, le serveur est accessible par son adresse IP. Cependant le routeur wifi était configuré comme dans le cadre d'une utilisation domestique. De ce fait les adresses IP étaient allouées dynamiquement aux postes. Le serveur pouvait donc potentiellement changer d'adresse IP à chaque redémarrage. Même si un serveur n'est pas arrêté ou redémarré fréquemment, en cas de panne électrique ou de nécessité d'un redémarrage pour la finalisation d'une installation, le serveur peut alors être associé à une nouvelle adresse IP et dans ce cas les membres de Solutions Isonéo sont obligés de rechercher cette nouvelle adresse IP pour accéder aux applications qu'il héberge. Ce fonctionnement n'étant pas durable, j'ai décidé de fixer l'adresse IP du serveur. Les autres postes qui seraient associés à la même adresse IP se retrouveraient en conflit mais le routeur est capable de résoudre dynamiquement le problème en allouant une nouvelle adresse IP au poste concurrent.

Pour cela j'ai modifié la configuration réseau qui se trouve dans le répertoire `etc` à la racine du système de fichier de Debian. Dans celui-ci se trouvait un fichier nommé

interface appartenant au dossier `network`. J'ai ajouté à cet élément le bloc suivant permettant de fixer l'adresse :

```
auto eth0
    iface eth0 inet static
        address 192.168.1.145
        netmask 255.255.255.0
        broadcast 192.168.1.255
        network 192.168.1.0
        gateway 192.168.1.1
```

L'IP associée à `address` est l'adresse à fixer pour le serveur. Le masque réseau (`netmask`) aide la machine à trouver les autres postes présents sur le même réseau. Si l'adresse IP à contacter est en dehors du masque alors la passerelle (`gateway`) est utilisée. L'adresse de diffusion (`broadcast`) est utilisée pour diffuser un paquet à toutes les machines accessibles via le même sous-réseau. L'adresse du réseau (`network`) est toujours une adresse réservée et permet de donner la première adresse du sous-réseau.

Le serveur était désormais accessible via une adresse IP fixe 192.168.1.X par les membres de Solutions Isonéo. Les personnes à l'extérieur de la pépinière d'entreprise n'ont cependant pas accès au serveur comme il serait souhaitable. J'ai alors contacté la personne en charge du réseau de la pépinière d'entreprise auprès de la Sicam. Cette personne n'ayant aucune qualification sur la configuration du routeur, j'eus libre accès à l'espace d'administration de celui-ci. Dans le souci de respecter les autres personnes de la pépinière d'entreprises, nous avons cependant communiqué vers la personne en charge du réseau auprès de la Sicam sur chacune des modifications et des impacts qu'elles pouvaient avoir afin d'obtenir une validation de sa part avant toute mise en place. La modification principale à faire était d'associer un nom à l'adresse IP 192.168.1.X afin que le serveur puisse être accessible depuis l'extérieur. En effet, le routeur, lié à Internet, propose ainsi à la toile d'accéder à ce nom qu'il résout automatiquement via l'adresse IP fournie pour transférer la communication vers le serveur de Solutions Isonéo. Pour la mise en place d'une telle ouverture, je devais obtenir l'aval de Julien Siega et Erwan Deschamps qui devaient eux-mêmes en référer à d'autres personnes. La priorité étant fixée sur le développement du produit Citrus, des

réflexions concernant l'ouverture du système d'information vers l'extérieur n'étaient pas prioritaires. Cette action n'a donc pas été effectuée avant mon départ.

d) Apprentissage sur le système d'installation de Debian

Le serveur devant être relié à Internet, je me suis demandée comment le sécuriser (Sitographie [20]). Debian est très souvent infecté par des virus quand il est installé sur un système hébergeant également Windows. En effet Debian est plus robuste que la distribution Microsoft et est également moins répandue donc moins sujette à des attaques. J'ai décidé de mettre en place un anti-virus nommé « clamav » sur le serveur avant de le connecter à Internet. J'ai alors pris en main le système d'installation utilisé sur Debian.

Les distributions basées sur Linux proposent toutes la même logique : la possibilité de compiler soi-même ses sources pour les mettre à disposition sous forme de logiciel gratuit. La communauté Debian teste ensuite ces logiciels et les mettent à disposition via des liens et les classifie ensuite sous des dénominations comme stable, instable, etc. Ces liens sont configurés dans le fichier `/etc/apt/source.list` et est appelé à chaque fois que les commandes de base de Debian sont utilisées. Pour Jenkins, j'ai ajouté un lien dans ce fichier afin de pouvoir simplement installer et mettre à jour le logiciel (Sitographie [21]). En effet il est important de noter que la communauté Debian ou toute communauté qui met à disposition des installations via des liens de ce type permettent aux utilisateurs de maintenir à jour facilement leur machine car un simple appel de la commande `aptitude full-upgrade` va vérifier sur les sites si une nouvelle version des applications installées est disponible pour en proposer l'installation.

Clamav étant un antivirus proposé par Debian, il a été facile de l'installer avec la commande `aptitude install clamav`. Pour rechercher un logiciel disponible sur les sites disponibles dans le fichier `sources.list`, Debian propose la commande `aptitude search nom_de_l_entité_recherchée`.

Cependant d'autres logiciels sont plus difficiles à installer comme Jira. La communauté Debian n'apprécie pas réellement les outils Java, qui demandent beaucoup de performance sans en maîtriser tous les aspects. Pour installer Jira sur le serveur, j'ai suivi la documentation officielle Atlassian (Sitographie [22] et [23]) qui suggère de télécharger le fichier binaire d'installation, de le copier sur le serveur puis de l'exécuter, ce qui a

fonctionné. La base de données a également été installée sans problème en suivant ces recommandations (Sitographie [24] et [25]).

Une problématique importante rencontrée lors de l'installation des solutions composant le système d'information fût la mise en place de Java. La librairie n'est pas difficile à installer car elle est disponible via la commande `aptitude install openjdk-7-jdk`. Cependant des logiciels comme Jira demandaient la mise en place de la librairie Java distribuée par Oracle pour laquelle il faut accepter des conditions d'utilisation. Une fois la librairie téléchargée, il faut installer sur le serveur un utilitaire pour interpréter le Java : `aptitude install java-package`. Cela permet ensuite de pouvoir convertir le binaire en fichier debian via la commande `fakeroot make-jpkg jdk-7u25-linux-x64.bin`. Le fichier debian peut ensuite être installé avec l'utilisateur « root » : `dpkg -i jdk-7u25-linux-x64.deb`.

Cependant le serveur possédait désormais deux versions de Java : la version Oracle et celle non-Oracle. Pour sélectionner celle à utiliser par défaut, il faut utiliser la commande `update-alternatives --config java` qui liste les configurations Java installées.

```
root@Wallee:/etc/init.d# update-alternatives --config java
There are 4 choices for the alternative java (providing
/usr/bin/java).
Selection        Path                                                    Priority    Status
-----
0   /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java        1071       auto mode
1   /usr/lib/jvm/j2sdk1.6-oracle/jre/bin/java             315       manual mode
* 2   /usr/lib/jvm/j2sdk1.7-oracle/jre/bin/java             317       manual mode
3   /usr/lib/jvm/java-6-openjdk-amd64/jre/bin/java        1061       manual mode
4   /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java        1071       manual mode
```

Figure 70 : Liste des configurations Java installées sur le serveur Debian

L'étoile indique la version sélectionnée par défaut. La commande `update-alternatives` demande alors de saisir le numéro de la version à utiliser par défaut, dans notre cas, le 2.

2.7.2. Construire un serveur fiable et adapté aux besoins

a) Automatisation des lancements applicatifs

Le serveur doit rester allumé afin que les applications qu'il héberge puissent fonctionner en tout temps. Cependant il est possible qu'à la suite d'une panne électrique ou d'une nécessité de redémarrage, le serveur soit éteint. Il faut donc prévoir qu'au démarrage du serveur, toutes les applications se mettent en route automatiquement.

Certaines applications comme Jenkins ou Jira se paramètrent automatiquement comme « daemon » du système d'exploitation. Un « daemon » est un logiciel qui se lance en tâche de fond dès le démarrage du système. Ces applications sont donc lancées au démarrage dès qu'elles sont installées.

Pour Sonar, ce n'est pas le cas. J'ai dû comprendre comment le système d'exploitation Debian initialise ses applications. Il possède plusieurs niveaux de lancement :

- ❖ 0 pour l'extinction du système
- ❖ 6 pour l'extinction du système par redémarrage
- ❖ 1 pour l'utilisation du système en mode simple utilisateur
- ❖ 2 à 5 pour l'utilisation du système en mode multi-user

J'ai pu trouver le niveau dans lequel se trouvait mon système lors d'un démarrage normal avec la commande `who -r` qui m'a renvoyé le niveau 2. Ce niveau de démarrage par défaut est également spécifié dans `/etc/inittab` où il était associé dans mon cas à `id:2:initdefault.`

Pour utiliser des applications lors de l'arrêt, redémarrage ou lancement, Debian possède un dossier dédié pour chacun des niveaux qui ont la forme de `/etc/rcX.d` où X correspond au niveau. Ces dossiers contiennent ensuite un lien vers un fichier se trouvant dans `/etc/init.d` qui ira à son tour appeler le réel fichier d'exécution de l'application. Dans ce répertoire, un fichier de nom `jenkins` peut être exécuté manuellement via la commande `./jenkins start` ou encore `./jenkins restart`. De même l'application peut être arrêtée via la commande `./jenkins stop`. L'application peut également être lancée directement à l'endroit d'exécution, par exemple, pour Jenkins : `/home/JENKINS/Installation_directory/jenkins.sh start`. Chaque fichier de lancement, se trouvant donc dans `/etc/init.d` peut

posséder un code pour engendrer un comportement différent de l'outil en fonction du niveau depuis lequel il est appelé. Pour créer le lien depuis `/etc/rc2.d` vers `/etc/init.d/sonar` il est possible d'utiliser la commande suivante :

```
update-rc.d sonar start 18 1 2 3 4 5 . stop 01 0 6
```

Cette commande permet de créer tous les liens de démarrage de Sonar nommés `sonar18` et tous les liens d'extinction de l'application nommés `sonar01`. Les numéros 1, 2, 3, 4 et 5 correspondent aux niveaux de lancement pour Debian et 0 et 6 aux niveaux d'extinction.

Les liens étaient effectivement créés cependant aucun lancement de Sonar au démarrage du serveur n'avait lieu. Les droits ont ensuite été donnés au compte « root » sur l'intégralité du dossier d'exécution de Sonar avec la commande `chown -R root /home/SONAR`, l'option « -R » permettant de lancer la commande de façon récursive afin que les droits soient attribués sur la totalité des éléments présents dans le dossier « SONAR ». Les liens étant créés avec une commande exécutée dans l'environnement de « root », les droits sont automatiquement attribués à ce compte. Le script `sonar` se trouvant dans `/etc/init.d` contient les chemins corrects vers le script d'exécution de l'application que Solutions Isonéo utilise. Il l'initialise dans une variable `$JSW = /home/SONAR/bin.linux86_64/sonar.sh`. Ce script contient également l'appel vers ce même script d'exécution : `su sonar -c « $JSW start »`. Cette commande signifie que la commande `/home/SONAR/bin.linux86_64/sonar.sh start` est exécutée avec le compte `sonar`, `su` permettant d'utiliser n'importe quel compte pour exécuter la commande. Or les droits sur le dossier « SONAR » étaient initialisés au compte « root ». Après une attribution des droits non pas au compte « root » mais au compte « sonar » avec la commande `chown -R sonar /home/SONAR`, l'application se lançait correctement au démarrage du serveur.

Toutes les applications se lançaient automatiquement au démarrage du serveur sauf Skype dont j'ai exposé le cas particulier précédemment. J'ai cependant étudié la possibilité de lancer Skype ainsi que le script `buildchatbot.py` au lancement du serveur.

L'application Skype est bien configurée avec le compte « jenkins » de Solutions Isonéo et se connecte donc automatiquement dès le démarrage d'une session. L'application demandant un nom d'utilisateur et un mot de passe pour retrouver les contacts et les

historiques de conversation, il n'était pas possible de la lancer en tant que « daemon » au démarrage du serveur. La contrainte imposant l'ouverture permanente d'une session sur le serveur était donc obligatoire.

La notification concernant le résultat de compilation des tâches jenkins fonctionne avec l'application Skype ainsi qu'avec le script python de BuilChatBot. J'ai donc cherché à lancer, dès le démarrage du serveur, ce script qui ne fait qu'écouter les actions de Jenkins. Le script doit être exécuté par le compte qui a ouvert la session lançant Skype.

Debian propose plusieurs possibilités :

- ❖ `.bashrc` qui est un fichier présent dans chaque session, peut être modifié pour exécuter des éléments à l'ouverture de la session
- ❖ `/etc/rc.local` peut être enrichi et est exécuté au lancement du serveur
- ❖ `bash.bashrc` peut être modifié pour lancer son contenu à l'ouverture de toutes les sessions.

La dernière solution n'est pas envisageable car le script de BuildChatBot serait exécuté à chaque ouverture de terminal. Cela engendre une utilisation non nécessaire de ressources ainsi que des effets de bords non contrôlables car le script python n'est pas conçu pour assumer le multi-instance.

La première solution qui paraît pourtant la plus adaptée car en relation avec la session ouverte, engendre malgré tout une exécution du contenu de `.bashrc` à chaque ouverture de terminal. De ce fait les mêmes effets de bord sont à craindre.

La seconde solution présente un inconvénient majeur : le script serait exécuté au lancement du serveur et non pas au démarrage de la session qui aurait lancé l'application Skype. De ce fait il faudrait que `rc.local` contienne une ligne par utilisateur pour que chacun d'eux lance le script « `buildchatbot.py` ».

Les trois solutions présentent le même inconvénient face à un script python qui n'est pas développé pour une utilisation poussée.

L'interface de Debian m'a finalement fourni une solution : elle propose la mise en place d'applications à exécuter au lancement de la session dans le menu « Startup Applications ».

b) Problème de compilation jenkins

Après la migration de Jenkins sur le serveur Debian, le produit Citrus ne se générait plus et la compilation ne fonctionnait plus.

L'erreur apparaissait lors de la phase de test après une compilation de tous les sous-modules composant Citrus.

Les tests n'avaient pourtant pas été modifiés. L'exécution manuelle de la commande `mvn test` sur le projet fonctionnait pourtant correctement, que celle-ci soit réalisée sur le serveur Debian ou sur ma machine Windows. Après vérification, le cycle Maven « install » exécuté sur le projet Citrus englobe la phase « test-integration » et non pas la phase « test » uniquement. A l'exécution manuelle de cette phase, en effet, la compilation ne fonctionnait pas. Les sources des tests ne pouvaient être la cause du problème puisque la commande `mvn test` aboutissait. En enlevant le dossier « Tests » du projet, je pouvais vérifier que les sources n'étaient pas impliquées et en effet la compilation du projet était effective. Après consultation de la documentation sur les cycles Maven, je me suis rendue compte que le cycle « integration-test » n'était effectué que si un élément de test était rencontré sur le projet. Cette phase engendrait donc bien l'erreur.

L'« integration-test » a pour but de vérifier que la totalité des éléments nécessaires au produit sont disponibles. La modification majeure pouvant engendrer une erreur de compilation était le changement de système d'exploitation. Après consultation des programmeurs, certaines librairies dédiées à Windows étaient utilisées dans le projet Citrus. Après un ajout de l'équivalent de chacune d'entre elles pour Linux, la compilation fonctionnait à nouveau. Cette nouvelle gestion des librairies dédiées à un système d'exploitation requiert la saisie d'informations supplémentaires pour informer Eclipse sur la librairie à utiliser en fonction du système d'exploitation employé.

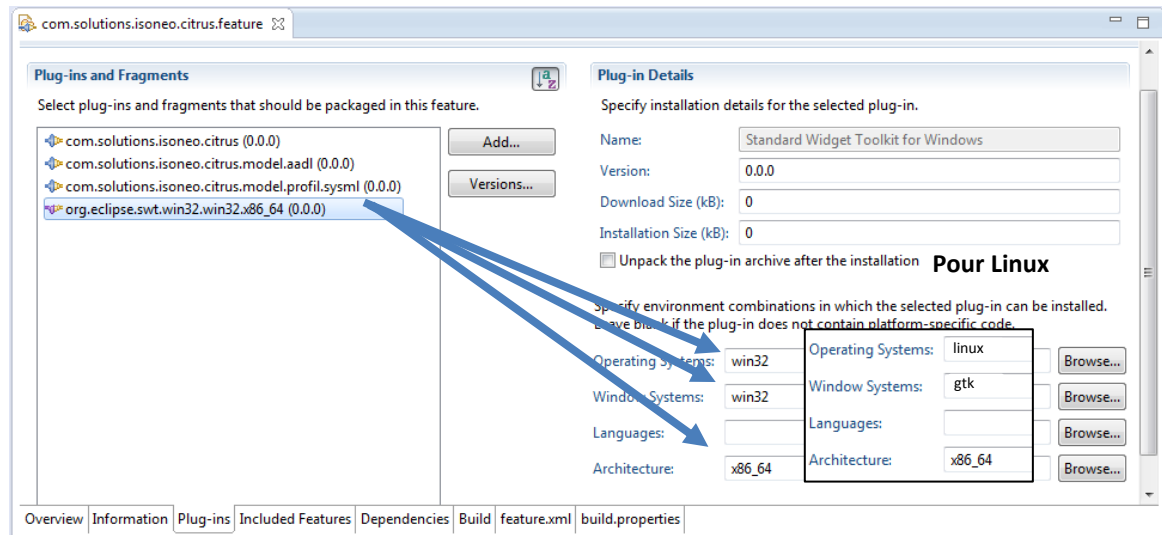


Figure 71 : Saisie d'informations supplémentaires pour la gestion des librairies dédiées à un système d'exploitation

Tous les programmeurs ont été informés par mail de cette nouvelle manipulation à prendre en compte en cas d'utilisation de librairies dédiées à un système d'exploitation. Des rappels ont également été effectués dans la conversation de Skype utilisée pour communiquer sur l'état des différentes compilations réalisées par Jenkins.

3. Résultats, discussions et perspectives

3.1. Système d'information complet mis en place

Solutions Isonéo possède désormais un système d'information centralisé sur un serveur Debian composé des principaux éléments suivants :

- ❖ Jenkins
- ❖ Sonar
- ❖ Jira
- ❖ Git
- ❖ Nexus

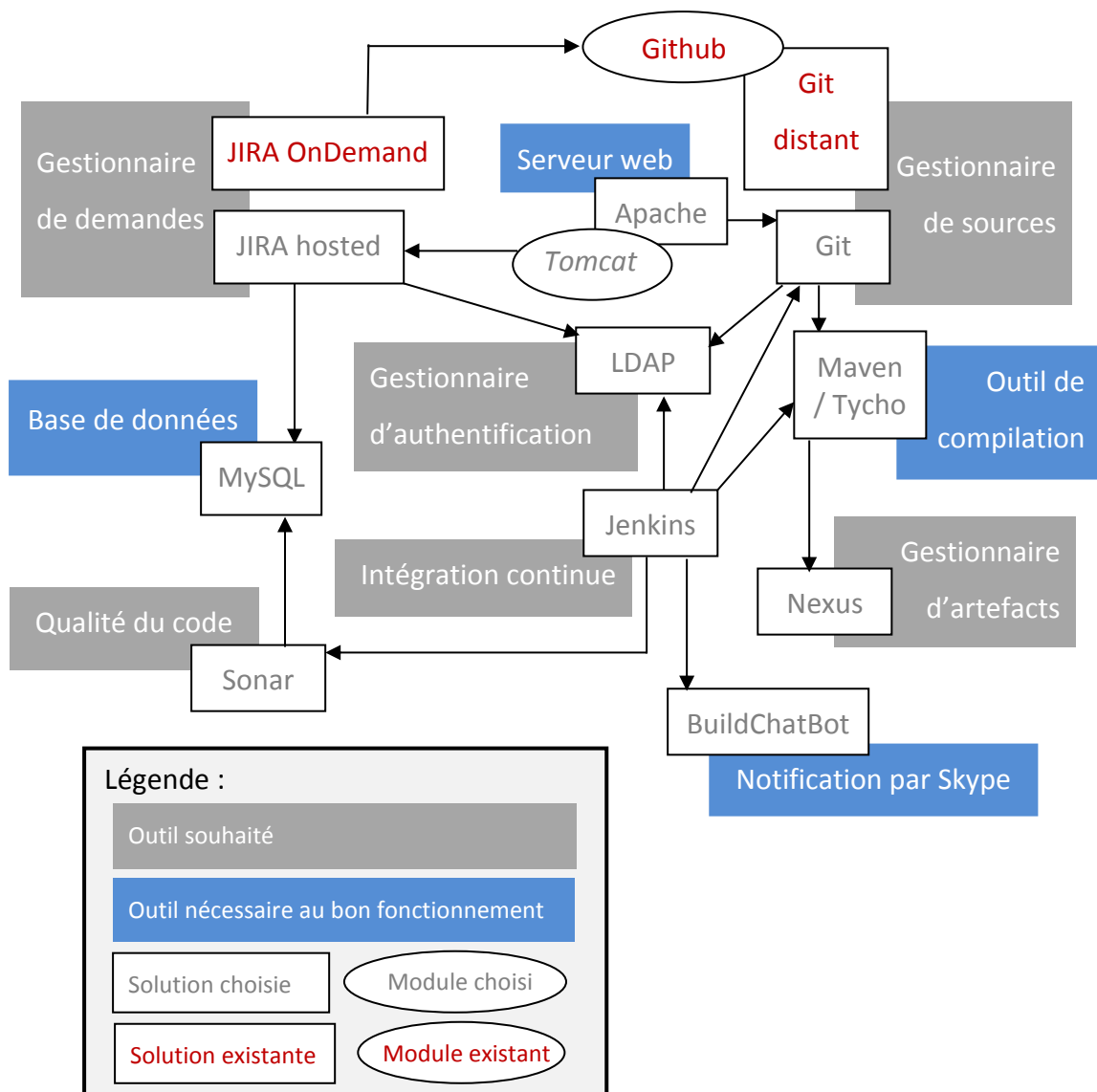


Figure 72 : Composants du S.I. construit

Ce système d'information est complet et répond aux besoins énoncés au début du stage :

- ❖ Respecter les contraintes de gestion locale des applications afin qu'elles ne soient pas accessibles par d'autres personnes grâce à l'installation de « Jira hosted » et de Git sur le serveur interne de la société
- ❖ Optimiser la gestion de projet avec la mise en place de processus et de type d'enregistrements clairs sur le gestionnaire de demandes
- ❖ Automatiser la création du produit grâce à Jenkins et à la mise en place de Maven et de Tycho
- ❖ Evaluer la qualité de code au travers de métriques calculées suite au couplage de Sonar avec Jenkins.
- ❖ Gérer la documentation avec la définition de règles de stockage et d'endroits dédiés
- ❖ Centraliser le système d'information sur un serveur Debian

Un autre besoin est apparu en cours d'année et celui-ci a également trouvé une solution :

- ❖ Centraliser les artefacts Maven avec la mise en place de Nexus

Cependant un besoin n'a été que partiellement réalisé faute de temps et de priorisation :

- ❖ Automatiser la génération documentaire

3.2. Résultats positifs

3.2.1. Evaluer la documentation créée

Toutes les nouvelles personnes entrant sur le projet Citrus installent correctement leur environnement de développement et déposent de moins en moins de sources sur Git engendrant des problèmes sur la compilation du projet : la documentation est donc claire et est écrite dans un langage compréhensible par tous.

Le système d'information du projet Citrus a donc été correctement documenté. Il en a été de même dans une vue plus globale pour le système d'information de Solutions

Isonéo. Cette documentation a été créée avant un départ en congés de plus de deux semaines et a été utilisée par les programmeurs. Celle-ci a permis de reprendre en main facilement le système d'information, le faire évoluer au besoin ou le corriger en cas de problème.

3.2.2. Respecter des règles définies sous Jira

Même si la génération automatique de documents n'a pas été menée jusqu'au bout, des règles d'utilisation de Jira sont désormais définies et tous les enregistrements actuellement sur le gestionnaire de demandes respectent ces conventions.

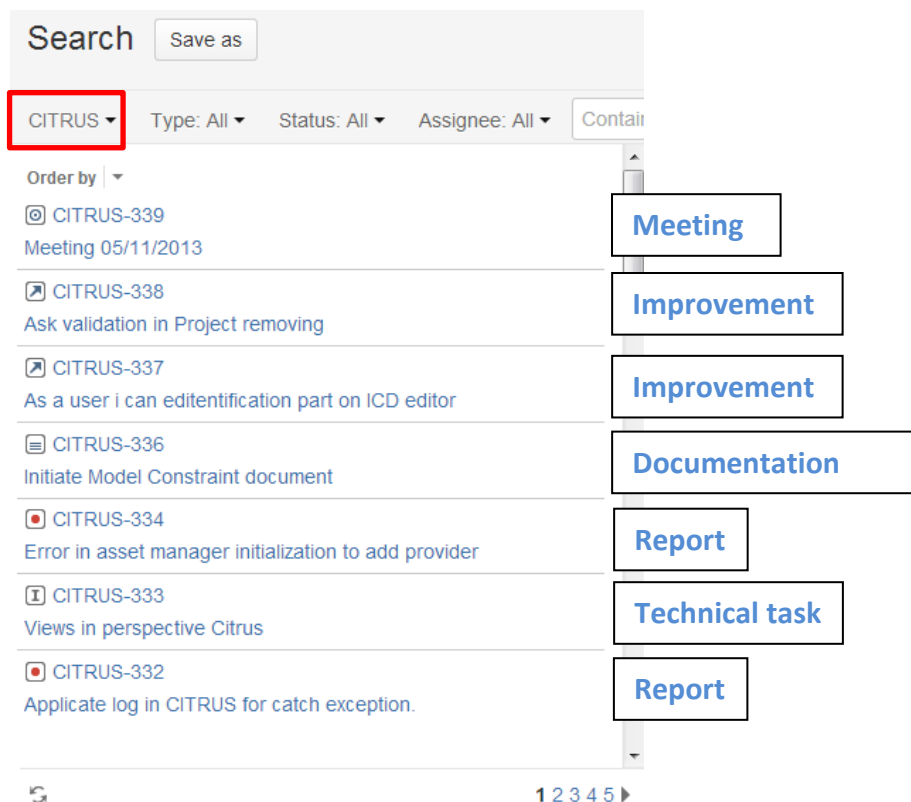


Figure 73 : Recherche sur le projet « Citrus » trouvant des enregistrements respectant la convention établie

3.2.3. Accroître la réactivité de l'équipe

L'équipe de Solutions Isonéo grandissant et ayant peu de moyens financiers, elle englobe désormais des employés d'Artal Technologies. La communication est correctement établie aujourd'hui : la conversation Skype permettant d'informer les programmeurs de l'état des compilations faites par Jenkins permet également aux programmeurs du projet Citrus d'échanger sur l'évolution du projet, ce qui présente un

bon moyen pour tous de participer, d'être informés rapidement des directions prises et de communiquer sur les solutions apportées.

De même une compilation échouée réalisée par Jenkins trouve un rétablissement de la situation dans la plupart des cas en moins d'une demi-journée.

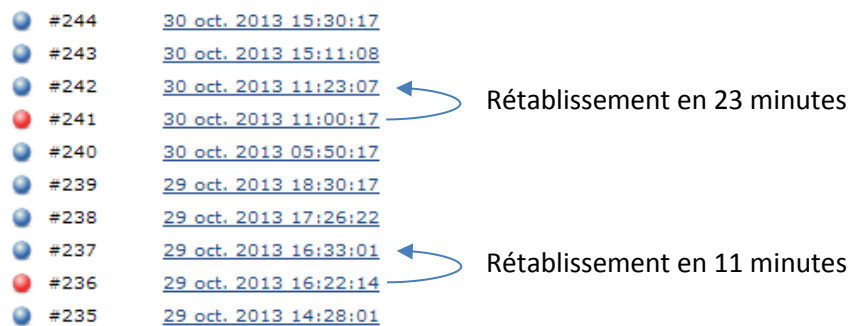


Figure 74 : Historique de compilation du projet Citrus : rétablissement rapide

3.2.4. Respecter des règles de codage

Les nombreux échanges au sujet des règles de codage à mettre en place sur le projet Citrus ont abouti à une version stable qui est appliquée sur le projet. Le nombre de violations de ces règles diminue constamment.

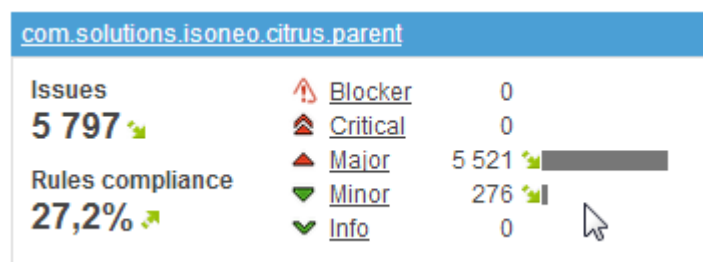


Figure 75 : Diminution du nombre de violations des règles de codage

3.2.5. Etablir une architecture projet

Citrus possède désormais une architecture projet respectée par tous les programmeurs. Celle-ci sera peut être également adoptée par les nouveaux projets comme celui de l'« Asset Manager », prouvant que le choix d'organisation est judicieux.

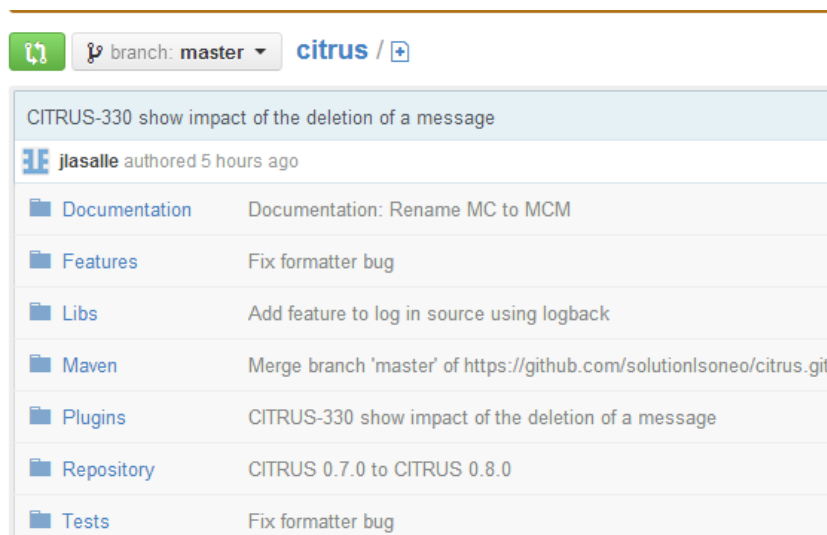


Figure 76 : Architecture du projet « Citrus » en discussion dans le projet « Asset Manager »

3.2.6. Automatiser les lancements applicatifs du serveur

Après un redémarrage du serveur suite à un orage, toutes les applications se sont relancées automatiquement sauf le script de BuildChatBot qui s'est relancé lors de l'ouverture de ma session.

3.3. Génération documentaire partiellement automatique

La génération documentaire n'a pas été menée à son terme faute de temps et de priorisation. Cette automatisation pourrait engendrer un gain important de temps pour les programmeurs ainsi qu'une mise à jour continue des documents en lien avec le développement en cours, ce qui est rarement le cas sur un projet de programmation.

Dans l'optique d'un outil complet, nous pourrions imaginer un outil s'adaptant à la norme devant être suivie par le projet. En effet l'outil pourrait recevoir en entrée un fichier saisi dans un format prédéfini (par exemple Excel) listant les documents à produire et ce qu'ils doivent contenir. L'outil saurait interpréter ces éléments et créer automatiquement la liste des documents ainsi que différents chapitres par élément devant les composer. Afin que les documents générés soient faciles à prendre en main, nous pourrions exiger, au sein du fichier d'entrée, les descriptions de chacun des éléments afin que celles-ci soient insérées au début de chaque chapitre.

Dans un second temps, l'outil proposerait une liste d'éléments pouvant être récupérés. Par exemple, la DO-330 nécessite la récupération de tous les éléments composant un

« Sprint » ce qui est récupérable via Jira ou encore la liste des tests associés à un besoin qui peut se récupérer conjointement avec Jira via l'« Epic » et avec Git via les éléments apparaissant dans l'onglet « Commits ». L'outil proposerait donc l'activation de ces besoins sous forme de case à cocher, demandant ensuite l'emplacement de l'outil qu'il est nécessaire d'avoir pour récupérer automatiquement la donnée souhaitée. Par exemple, l'outil peut proposer de fournir par sous-projet le pourcentage de couverture du code par des tests. Cela permet de vérifier que les sources du projet sont contrôlées par des tests unitaires. L'utilisateur de l'outil pourrait vouloir obtenir cette statistique et en cochant la case concernée, il devrait simplement fournir l'emplacement de Sonar. L'architecture de cette application et le stockage de ses données se faisant au même endroit pour n'importe quelle installation, l'outil pourrait activer le module pour récupérer les statistiques à l'emplacement en question, pointant vers Sonar.

Dans un objectif à court terme, adapté à la DO-330 dans le cas de Solutions Isonéo, il est nécessaire de programmer les plug-ins pour les ajouter aux applications composant le système d'information afin de récupérer les informations nécessaires à l'obtention de la norme. Même si la réalisation n'a pas été faite, j'ai étudié de façon théorique les rôles des différents plugins à mettre en place.

Par exemple le document « Tool Configuration Management Record » a pour but de fournir une vue d'ensemble sur l'évolution du projet avec les différentes compilations et leur état ainsi que le nombre de violations des règles de codage et le pourcentage de couverture de tests lors de chacune d'elle. Le document pourrait avoir le format suivant :

Dans Jenkins	#593							Liste des commits associés
	(lien vers la page Jenkins du build)							n° des commits - Commentaires
								...
								List des éléments de configuration
								com.solutions.... 0.2.0
								...
Résultat de la compilation :	FAILURE :	arrêt (compil failed ou test) + raisons						
	SUCCESS :	passage à SONAR						
Toujours le plugin JENKINS qui se connecte à la BD								
								Vérification des checkstyles
								Minor violations :
								Major violations :
								...
								Informations sur les binaires
								Emplacement
								Noms

Figure 77 : Format possible du « Tool Configuration Management Record » de la DO-330

Pour obtenir ces différentes informations, le plugin à développer devrait rechercher dans Jenkins l'historique des compilations et extraire le résultat de chacune d'elles. Si la compilation est réussie, le plugin devrait se connecter à la base de données de Sonar en lecture seule pour en extraire le nombre de violations des règles de codage ainsi que le pourcentage de couverture de code par des tests. Pour finir le plugin devrait pouvoir lister pour chaque compilation la liste des sous-projets lui appartenant permettant ainsi de suivre la croissance du projet.

Une autre étude intéressante fût celle du « Tool Problem Reports » dont l'objectif est de fournir la preuve que le projet audité maîtrise bien les problèmes rencontrés – qu'ils soient de programmation ou d'intégration - tout au long du développement. Pour cela, Solutions Isonéo doit sauvegarder dans la documentation toutes les compilations échouées enregistrées dans Jenkins même celles rencontrées pour un problème mineur : réseau par exemple. Une compilation échouée engendrerait l'envoi d'un mail à la totalité de l'équipe et/ou la génération d'un enregistrement dans l'application Jira. Le plugin récupérerait alors les informations saisies dans cet enregistrement : description du problème, description de sa résolution contenant toutes les solutions étudiées et la solution résolvant effectivement le problème. Le plug-in insèrerait également dans la

documentation des informations sur le « push » de Git ayant permis de résoudre le problème comme la liste des documents modifiés et les commentaires saisis par le développeur. Le plugin à développer doit donc être capable de trouver des informations sur Jenkins (Sitographie [26]), puis sur Jira pour finalement terminer par Git.

3.4. Axes d'amélioration du système mis en place

Le système d'information mis en place fonctionne correctement, cependant certaines améliorations peuvent être apportées.

3.4.1. Migrer les applications pour respecter totalement les contraintes

Des contraintes ont été imposées par le partenaire afin que les sources soient stockées sur le serveur de Solutions Isonéo et non pas sur un serveur distant comme c'est le cas avec Jira OnDemand ou Git exploité via Github.

Jira et Git ont été installés sur le serveur de Solutions Isonéo cependant ils ne sont pas utilisés : Jira n'a pas été migré depuis Jira OnDemand et Git n'est utilisé que pour le projet « Terea » récupéré depuis Artal Technologies mais ne gère pas les versions de Citrus toujours hébergé sur un serveur Git distant.

3.4.2. Installer un outil de GED

La documentation serait plus facilement accessible et maintenable si un outil de gestion documentaire était mis en place comme Nuxéo. De plus des droits d'accès pourraient être attribués différemment aux membres de Solutions Isonéo et de CMC Electronique.

3.4.3. Améliorer le processus de notification par Skype

Le serveur est configuré pour être au maximum automatisé, cependant les notifications concernant l'état des compilations faites par Jenkins via l'application Skype n'est pas optimal. Il faudrait chercher à mettre en place le plug-in « Skype » dans Jenkins et mettre en place un serveur virtuel esclave en 32 bits pour que la solution soit pérenne et totalement automatique lors d'un redémarrage du serveur.

De même la solution via Skype est la plus dynamique cependant tous les salariés d'Artal Technologies interagissant sur les projets de Solutions Isonéo n'utilisent pas cette application et ne sont donc pas informés de l'état des compilations réalisées par Jenkins.

Il faudrait étudier la faisabilité d'un envoi supplémentaire par mail pour certains utilisateurs.

Ce principal moyen d'échange qui présente de grandes qualités en termes de dynamisme pose également un problème de traçabilité. En effet les échanges sont sauvegardés dans l'historique de conversation dans lequel il est fastidieux de rechercher. Pour pallier à ce problème nous pourrions imaginer un script qui s'exécute sur le serveur toutes les nuits afin de copier la discussion réalisée dans la conversation de Skype durant la journée dans un mail envoyé à tous les programmeurs de Citrus qu'ils aient un compte Skype ou non ; cela pourrait être un moyen pour les salariés n'ayant pas de compte Skype de suivre l'état des compilations ainsi que les échanges de leurs collègues.

3.4.4. Modifier des paramètres de notifications sur Jira

Jira est un outil qui envoie actuellement beaucoup de notifications par mail. Après migration, l'équipe Solutions Isonéo devrait s'employer à désactiver plusieurs notifications afin que la boîte mail de chacun d'eux soit moins remplie de notifications inutiles, ce qui engendrerait une meilleure lecture des notifications utiles.

3.4.5. Ouvrir les applications sur Internet

Toutes les applications du système d'information doivent également être ouvertes après migration aux employés du partenaire CMC Electronique afin que les statistiques générées par les outils du système d'information soient prises en compte dans la productivité et l'évolution du projet Citrus. Par exemple le nombre de violations des règles de codage diminue significativement depuis leur mise en place.

3.4.6. Sauvegarder les éléments générés à la compilation

Jenkins génère automatiquement un produit ainsi qu'un exécutable de l'application Citrus cependant ceux-ci ne sont pas sauvegardés. Il est important afin de répondre à la norme DO-330 de sauvegarder ces éléments sur le serveur de Solutions Isonéo, ainsi que sur celui d'Artal Technologies dans des soucis de redondance.

3.4.7. Exploiter au maximum des éléments du S.I.

Les outils mis en place pour construire le système d'information de la société Solutions Isonéo ne sont pas exploités à leur maximum. Il serait important d'allouer du temps

dans l'exploration des différentes fonctionnalités afin d'obtenir une vue d'ensemble des possibilités du S.I. qui pourraient susciter de nombreux intérêts.

Sonar par exemple pourrait fournir des statistiques plus fines permettant de mettre en avant les violations majeures, mineures, bloquantes, etc. Pour cela, il faut cependant définir chacun de ces termes et y associer des règles de codage.

Conclusion

Le suivi de la méthodologie Scrum apporte une ligne directrice très dynamique dans l'évolution des projets informatiques qui l'adoptent.

Cependant celle-ci peut engendrer une souplesse excessive du projet qui revoit ses orientations à chaque « Sprint review » (Bibliographie [9]).

Le système d'information doit être au service de ses utilisateurs : les programmeurs. Il doit permettre d'automatiser des tâches afin d'accélérer certains traitements ou fournir des résultats témoignant de l'efficacité du projet.

Cependant les utilisateurs doivent être sensibilisés aux nouveaux composants et s'investir. Toute l'équipe étant actrice de la performance du projet évalué par le système d'information, le chef de projet ne doit pas être le seul à s'acquitter de cette tâche.

Après cette année de stage, il me semble aujourd'hui indispensable de rapprocher les programmeurs et les personnes en charge du système d'information afin de :

- ❖ Mettre à disposition des programmeurs et des chefs de projet l'expertise des personnes en charge du système d'information pour proposer des solutions mais aussi éviter la sur-qualité
- ❖ Mettre à disposition des personnes en charge du système d'information, les réels besoins des programmeurs et chefs de projet
- ❖ Mener une politique de changement pour investir l'équipe de développement dans des objectifs communs :
 - Obtenir de meilleurs résultats de performance portés par le chef de projet,
 - Utiliser au maximum les possibilités des outils composant le système d'information permettant aux personnes l'ayant mis en place de concrétiser l'utilité de leur travail
 - Faire évoluer le système d'information vers un état le plus automatisé et adapté possible

Il est cependant nécessaire de ne pas fusionner les deux entités. En effet l'équipe de programmation doit être informée par des communications qui doivent cependant rester rares et pertinentes. L'objectif étant de présenter l'utilité du système d'information sans que sa prise en main n'alourdisse de façon trop importante la charge de travail de l'équipe de développement. Les investissements personnels ne doivent tout de même pas être cachés mais doivent être contrebalancés avec le progrès apporté.

Cette expérience m'a permis d'identifier les limites mais aussi les gains importants de la méthodologie Scrum qui met en avant la communication entre les partis engagés dans le projet. Celle-ci devrait être considérée par le monde des systèmes d'information. L'équipe de développement et l'équipe en charge du système d'information devraient se réunir lors des « Daily Scrum » afin que cette dernière puisse se rendre compte des réels besoins de l'équipe de programmation qui sont en évolution permanente. Dans un monde où la performance possède une place importante, il serait également bénéfique de suivre régulièrement les besoins des chefs de projet.

Au sein de Solutions Isonéo, les composants du système d'information ont été mis en place avec succès, cependant les capacités de ces composants sont au-delà de leurs exploitations actuelles, compte tenu de la taille et des priorités de la « start-up ». D'autre part, des éléments très utiles à la société comme la génération automatique de la documentation n'ont pas été menés à terme faute de temps.

Les gains du système d'information mis en place sont cependant perçus et investis par tous les membres de l'équipe.

Annexes

1. ANNEXE 1	134
2. ANNEXE 2	135

1. Annexe 1

Liste des éléments à produire par Solutions Isonéo pour la DO-330

Table T-2 Tool Development Processes														1st year				
	Objective		Applicability by TQL					Output		Control Category by TQL					Applicable	CC	Who	Comment
	Description	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5				
1	Tool Requirements are developed.	5.2.1.1a	○	○	○	○		Tool Requirements Trace Data	10.2.1 10.2.7	① ①	①①	①①	①①		○	②	Isonéo	
2	Derived tool requirements are defined.	5.2.1.1b	○	○	○	○		Tool Requirements	10.2.1	①	①	①	①					
3	Tool architecture is developed	5.2.2.1a	○	○	○	○		Tool Design Description	10.2.2	①	①	①	②		○	②	Isonéo	Only high level architecture is described
4	Low-level tool requirements are developed.	5.2.2.1b	○	○	○			Tool Design Description Trace Data	10.2.2 10.2.7	① ①	①①	①①						
5	Low-level derived tool requirements are defined.	5.2.2.1c	○	○	○			Tool Design Description	10.2.2	①	①	②						
6	Tool Source Code is developed.	5.2.3.1a	○	○	○			Tool Source Code	10.2.3	①	①①	①①						
		5.2.3.1b						Trace Data	10.2.7	①								
7	Tool Executable Object Code is produced.	5.2.4.1a	○	○	○	○		Tool Executable Object Code	10.2.4	①	①	①	①					
8	Tool is installed in the tool verification environment(s).	5.2.4.1b	○	○	○	○		Tool Executable Object Code	10.2.4	①	①	①	①					

2. Annexe 2

Extraction des informations pour le document « Tool Requirements » de la DO-330

Table T-2 Tool Development Processes																		
Objective			Applicability by TQL					Output		Control Category by TQL					Applicable	CC	Who	Comment
Description		Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5				
T-2 Tool Development Processes																		
1	Tool Requirements are developed.		5.2.1.1a	○	○	○	○		Tool Requirements	10.2.1	①	①①	①①	①①		○	②	Isonéo
								Trace Data	10.2.7	①								
Tool Requirements			Are the requirements used to develop and verify the tool. Tool Operational Requirements, Tool Development Plan and Tool Requirements Standards are the inputs to this process. The Tool Requirements are the primary outputs of this process. It should be include :															
Necessary to define :			<div>- Tool Operational Requirements</div> <div>- Tool Development Plan</div> <div>- Tool Requirements Standards</div>															
Needs									Where					How				
Description of the tool functions and technical features, including modes of operation									Excel existant (EPIC)					Création d'Epic dans Jira + liaison aux User Stories				
User manual (which can be part of Tool Requirements) with user instructions, installation instructions, list of error messages, and constraints.									Manuel mais génération du word automatique					Voir les solutions de transformation des "How To" du Help Content (xml html) Eclipse vers le word ou inversement				
Requirements to describe the users' ability to customize the tool									Manuel mais génération du word automatique					A Ajouter dans le Help Content : "Configuration" + avertissement vers la config dans le "HowTo" en relation				
Functional requirements with the appropriate level of detail (see 5.2.1.2.k)									Jira					User Story et Subtask d'Epic				
Specific requirements (if necessary, for compliance with tool operational environment)									Jira					Nouveau type d'enregistrement : Specific Requirement				
Specific requirements : failure modes and response to inconsistent inputs									Jira					Nouveau type d'enregistrement : Specific Requirement				
Expected responses of the tool under abnormal operating conditions									Jira					Nouveau type d'enregistrement : Soft Behaviour (Bon / Pas bon)				
Interface requirements between the tools in case of collection of tools									Manuel / Considération de 2 collections d'outil : Asset manager et SE Studio					Sur User Story : Ajout de composants : SE Studio / Asset Manager Keyword : "Interface" pour être obligé de la lier à User Story de l'autre collection				

Références

1. Bibliographie

- [1] BLOCH L., 2005, Systèmes d'information, obstacles et succès. Vuibert.
- [2] MESSENGER ROTA V., TABAKA J., 2013, Gestion de projet : vers les méthodes agiles. Eyrolles.
- [3] VANNIEUWENHUYZE A., 2013, Scrum – une méthode agile pour vos projets. Eni Eds.
- [4] DE LOOF N., HERITIER A., 2009, Apache Maven. PEARSON.
- [5] NEGUS C., 2007, Linux Bible 2007 Edition, Boot up to Ubuntu, Fedora, KNOPPIX, Debian, SUSE and 11 other distributions. John Wiley & sons, Inc.
- [6] HERTZOG R., 2011, Debian Squeeze. Eyrolles.
- [7] F. KRAFFT M., 2006, The Debian system: Concepts and Techniques. Open Source Press.
- [8] PINKALL POLLEI R., 2007, Debian 7: System Administration Best Practices. PACKT.
- [9] MORISSEAU L., 2012, Kanban pour l'IT, une méthode d'amélioration des processus complémentaire de Scrum. Dunod.

2. Sitographie

- [1] PROGRAMME SA2GE. « Le projet SA²GE ».
Disponible sur : <http://www.sa2ge.org/index.php/fr/a-propos/le-projet> (consulté le 1 mars 2013)
- [2] Scrum ALLIANCE. « Scrum is an innovative approach to getting work done ».
Disponible sur : http://scrumalliance.org/pages/what_is_scrum (consulté le 11 mars 2013)
- [3] ECLIPSE FOUNDATION. « Product Configuration Editor »
Disponible sur :
http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.pde.doc.user%2Fgui%2Ftools%2Feditors%2Fproduct_editor%2Feditor.htm&cp=4_3_2_4 (consulté le 28 février 2013)
- [4] SCOTT C. « Pro Git ».
Disponible sur : <http://git-scm.com/book/fr/> (consulté le 20 février 2013)

- [5] KAUFFMANN J. « Installation d'un serveur Git avec Apache sous Linux »
Disponible sur : <http://blog.freelan.org/2011/09/02/installation-dun-serveur-git-avec-apache-sous-linux/> (consulté le 21 juin 2013)
- [6] ECLIPSE FOUNDATION. « Buckminster, Component Assembly Project ».
Disponible sur : <http://www.eclipse.org/buckminster/> (consulté le 14 mars 2013)
- [7] VOGEL L. « Eclipse Tycho – Tutorial for building Eclipse Plugins and RCP Applications »
Disponible sur :
<http://www.vogella.com/articles/EclipseTycho/article.html#productbuild>
(consulté le 4 mars 2013)
- [8] APACHE SOFTWARE FOUNDATION. « Maven documentation ».
Disponible sur : <http://maven.apache.org/guides/index.html> (consulté le 5 mars 2013)
- [9] KAWAQUCHI K., MARTINEZ I. « Using Jenkins ».
Disponible sur : <https://wiki.jenkins-ci.org/display/JENKINS/Use+Jenkins>
(consulté le 15 mars 2013)
- [10] JAN G. « Website about Regular Expressions ».
Disponible sur : <http://www.regular-expressions.info/> (consulté le 5 août 2013)
- [11] SONATYPE INC., MOSER M., O'BRIEN T., VAN ZYL J., BRADICICH D., CASEY J., CSERVENAK T., DEMERS B., FOX B., FROEDER M., HAMMAR A., SEDDON R. and XU J.
« Repository Management with Nexus »
Disponible sur : <http://books.sonatype.com/nexus-book/reference/index.html>
(consulté le 3 juillet 2013)
- [12] HOLBREICH A. « Installation and Configuration of Sonatype Nexus »
Disponible sur : <http://alexander.holbreich.org/2012/10/sonatype-nexus-setup/>
(consulté le 3 juillet 2013)
- [13] APACHE SOFTWARE FOUNDATION. « The Central Repository ».
Disponible sur : <http://search.maven.org/#browse> (consulté régulièrement durant l'année 2013)
- [14] SERVER WORLD. « Configure LDAP server ».
Disponible sur : http://www.server-world.info/en/note?os=Debian_7.0&p=ldap
(consulté le 10 juillet 2013)
- [15] ATLASSIAN. « Supported Platforms »

Disponible sur :

<https://confluence.atlassian.com/display/JIRA052/Supported+Platforms>

(consulté le 29 avril)

[16] L'EQUIPE DE L'INSTALLATEUR DEBIAN. « Manuel d'installation pour la distribution Debian GNU /Linux ».

Disponible sur : <http://www.debian.org/releases/stable/amd64/> (consulté le 3 juin 2013)

[17] OSAMU A. « Référence Debian ».

Disponible sur :

<http://www.debian.org/doc/manuals/debian-reference/index.fr.html> (consulté le 7 juin 2013)

[18] LEWIS AJ., SISTINA SOFTWARE INC., RED HAT INC., TERRASCALE TECHNOLOGIES INC., and RACKABLE SYSTEMS INC. « LVM How To »

Disponible sur : <http://tldp.org/HOWTO/LVM-HOWTO/reducelv.html> (consulté le 20 mai 2013)

[19] DELL INC., « PowerEdge T420 – Technical guide »

Disponible sur : <https://www.advania.is/library/Files/1.-Vefverslun-PDF/Midlaegar-lausrnir/PowerEdge/dell-poweredge-t420-technical-guide.pdf> (consulté le 11 novembre 2013)

[20] REELSEN A., FERNANDEZ-SANGUINO PENA J. « Securing Debian Manual ».

Disponible sur : <http://www.debian.org/doc/manuals/securing-debian-howto/index.fr.html> (consulté le 7 juin 2013)

[21] WILSON K. « Installing Jenkins on Ubuntu »

Disponible sur :

<https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu> (consulté le 15 mai 2013)

[22] ATlassian Confluence. « Installing Jira on Linux ».

Disponible sur :

<https://confluence.atlassian.com/display/JIRA/Installing+JIRA+on+Linux> (consulté le 17 juin 2013)

[23] ATlassian Confluence. « Installing JIRA from an Archive File on Windows, Linux or Solaris ».

Disponible sur :

<https://confluence.atlassian.com/display/JIRA/Installing+JIRA+from+an+Archive+File+on+Windows%2C+Linux+or+Solaris> (consulté le 17 juin 2013)

[24] ATlassian Confluence. « Setting Up a MySQL Database on Linux for JIRA ».

Disponible sur :

<https://confluence.atlassian.com/display/JIRA/Setting+Up+a+MySQL+Database+on+Linux+for+JIRA> (consulté le 17 juin 2013)

[25] ATlassian Confluence. « Connecting JIRA to MySQL ».

Disponible sur :

<https://confluence.atlassian.com/display/JIRA/Connecting+JIRA+to+MySQL> (consulté le 17 juin 2013)

[26] KAWAQUCHI K., GLICK J. « Extend Jenkins ».

Disponible sur : <https://wiki.jenkins-ci.org/display/JENKINS/Extend+Jenkins> (consulté le 3 juin 2013)

Liste des figures

Figure 1 :	Organigramme haut niveau	11
Figure 2 :	Différents panneaux composant l'IDE Eclipse	15
Figure 3 :	Modèle des méthodes agile.....	17
Figure 4 :	Interface web de Jira	19
Figure 5 :	Composant du système d'information existant : « Jira OnDemand »	20
Figure 6 :	Schéma de fonctionnement de Git	21
Figure 7 :	Interface web de Github	22
Figure 8 :	Composants du S.I. existant : prise en compte de « Git distant ».....	23
Figure 9 :	Environnement de génération de Citrus	27
Figure 10 :	Représentation du réseau fourni par la Sicam et utilisé par Solutions Isonéo.....	31
Figure 11 :	Organisation des besoins selon des aspects techniques.....	34
Figure 12 :	Plug-in « Greenhopper » de Jira OnDemand	35
Figure 13 :	Schéma explicatif du tableau de bord du Sprint 4 du projet Citrus	37
Figure 14 :	Liens entre les types d'enregistrement existants et nécessaires au plugin « Greenhopper »..	38
Figure 15 :	Vue du plug-in « Pivot » via le tableau de bord Jira	39
Figure 16 :	Extraction au format excel fournit par le plug-in « Pivot ».....	39
Figure 17 :	Organisations des éléments dans Jira	43
Figure 18 :	Interface graphique d'administration des « issue type scheme » de Jira	45
Figure 19 :	Onglet « Commits » d'un enregistrement Jira	46
Figure 20 :	Mise en place du lien entre Jira et Git exploité via Github	46
Figure 21 :	Composants du S.I. en construction : mise en place du lien entre « Jira OnDemand » et « Git distant » via « Github »	47
Figure 22 :	Composants du S.I. en construction : mise en place de « Jira hosted »	50
Figure 23 :	Etude des besoins du « Tool Requirements » de la DO-330.....	51
Figure 24 :	Eléments à mettre en place dans Jira pour répondre à une partie des besoins pour la DO-330.	54
Figure 25 :	Différence d'architecture entre Git en ligne et Git hébergé sur le serveur de Solutions Isonéo	56
Figure 26 :	Architecture et système de Git	57
Figure 27 :	Vue de l'administrateur et de l'utilisateur sur un même dépôt Git	58
Figure 28 :	Composants du S.I. en construction : prise en compte de « Git » hébergé sur le serveur de la société	61
Figure 29 :	Conversion d'un projet en projet Maven via l'interface d'Eclipse.....	62
Figure 30 :	Correspondance entre les « goals » Maven et ses plugins	64
Figure 31 :	Architecture du projet Citrus, référence pour tous les projets de Solutions Isonéo.....	65
Figure 32 :	Enchaînement des compilations à partir du pom parent	67

Figure 33 :	Rapport de compilation Maven en ligne de commande.....	68
Figure 34 :	Génération du produit.....	70
Figure 35 :	Le cycle de compilation par défaut Maven.....	72
Figure 36 :	Composants du S.I. en construction : mise en place d'un outil externe de compilation : Maven	72
Figure 37 :	Interface graphique Jenkins.....	73
Figure 38 :	Tâches Jenkins	74
Figure 39 :	Liste des plugins nécessaires à la mise en place de Github avec Jenkins.....	74
Figure 40 :	Schéma des configurations du projet sous Jenkins.....	75
Figure 41 :	Lien de redirection vers le projet sous Github.....	76
Figure 42 :	Lien « githubweb » dans la liste des « commits ».....	76
Figure 43 :	Exemple de modifications sur des fichiers impactés par un « commit »	76
Figure 44 :	Schéma de compilation Jenkins	77
Figure 45 :	Paramétrage de la scrutation de Git par Jenkins	77
Figure 46 :	Icône d'information sur le type de lancement de la compilation	77
Figure 47 :	Paramétrage Jenkins pour le lancement d'une compilation suivant une modification précise	79
Figure 48 :	Paramétrage Jenkins pour compiler uniquement un plugin du projet	79
Figure 49 :	Composants du S.I. en construction : mise en place de Jenkins.....	80
Figure 50 :	Composants du S.I. en construction : mise en place des notifications Skype pour Jenkins.....	83
Figure 51 :	Extraction au format csv des métriques dans Sonar	84
Figure 52 :	Extraction PDF des métriques Sonar.....	84
Figure 53 :	Pointage de la base de données Sonar depuis Jenkins	85
Figure 54 :	Lancement d'une analyse Sonar après chaque compilation réalisée par Jenkins	86
Figure 55 :	Tableau de bord Sonar	86
Figure 56 :	Remontées des erreurs de style de code par Eclipse.	87
Figure 57 :	En-tête obligatoire des classes du projet Citrus, inséré lors de la création	88
Figure 58 :	Expression régulière contrôlant les éléments obligatoires de l'en-tête.....	88
Figure 59 :	Méthode Java contenant une clause « throws » débordant de la fenêtre affichée	89
Figure 60 :	Méthode Java contenant une clause « throws » après un formatage	89
Figure 61 :	Modification du « formatter » afin de renvoyer à la ligne la clause « throws » lors du Ctrl + Shift + F	90
Figure 62 :	Composants du S.I. en construction : mise en place du contrôle de la qualité avec Sonar.....	91
Figure 63 :	Fonctionnement de Nexus	93
Figure 64 :	Composants du S.I. en construction : mise en place de la centralisation des artefacts avec Nexus	95
Figure 65 :	Configuration du LDAP dans Jenkins	102
Figure 66 :	Ajout d'un utilisateur dans Jenkins inconnu du LDAP	102
Figure 67 :	Configuration du LDAP dans Jira	103

<i>Figure 68 : Composants du S.I. en construction : mise en place de la gestion des authentifications avec un LDAP</i>	105
<i>Figure 69 : Schéma du réseau exploité par Solutions Isonéo avec son serveur</i>	112
<i>Figure 70 : Liste des configurations Java installées sur le serveur Debian</i>	115
<i>Figure 71 : Saisie d'informations supplémentaires pour la gestion des librairies dédiées à un système d'exploitation</i>	120
<i>Figure 72 : Composants du S.I. construit</i>	121
<i>Figure 73 : Recherche sur le projet « Citrus » trouvant des enregistrements respectant la convention établie</i>	123
<i>Figure 74 : Historique de compilation du projet Citrus : rétablissement rapide</i>	124
<i>Figure 75 : Diminution du nombre de violations des règles de codage</i>	124
<i>Figure 76 : Architecture du projet « Citrus » en discussion dans le projet « Asset Manager »</i>	125
<i>Figure 77 : Format possible du « Tool Configuration Management Record » de la DO-330</i>	127

Liste des tableaux

<i>Tableau I : Compatibilité des solutions composant le S.I. avec les systèmes d'exploitation basés sur Linux.</i>	<i>..... 108</i>
--------------------------------------------------------------------------------------------------------------------------	------------------