
	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

WP 4.6 - Traçabilité des exigences dans le code


Livrable du au titre du projet	COCLICO
Lot	Sous-projet 4
Tache	4.6
Livrable	L4.6.4 - WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code

Rédacteur(s)	Vérificateur(s)	Approbateur(s)
Cédric TRAN-XUAN		

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011


Documents applicables
Annexe technique au projet COCLICO

Documents de références (pour information)

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011


Gestion des versions

N° de version	Date	Auteurs	Modification apportées
0.9	07/18/11	Cédric TRAN-XUAN	Initialisation du document
1.0	07/21/11	Cédric TRAN-XUAN	Finalisation de la version 1.0

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

Sommaire

1	Introduction.....	5
2	Contexte.....	6
	Traçabilité des exigences dans le code.....	6
	Métriques.....	6
3	Études de solutions.....	8
	Principe général.....	8
	Utilisation de commentaires du gestionnaire de sources.....	9
	Tags dans le code.....	11
	Principe.....	11
	JavaDoclets.....	13
	Tags de commentaire.....	14
	Conclusion.....	14
4	Pistes techniques.....	16
	Implémentation.....	16
	Plugin Maven TagList.....	16
	Développement d'un plugin Sonar.....	16
	Intégration avec le poste de développement.....	18
5	Annexes.....	20
	Références.....	20


	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

1 Introduction

Ce document a pour objectif de décrire une solution permettant d'assurer la traçabilité des exigences dans le code.

Il s'inscrit dans le cadre du projet Coclico et de la participation de BULL qui marque sa volonté d'axer, pour sa forge logicielle NovaForge TM, des développements futurs autour de l'ingénierie des exigences.

Ce document est le livrable de l'atelier WP 4 Coclico (livrable WP 4.6 *Étude d'une solution de traçabilité des exigences dans le code*).

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

2 Contexte

L'ingénierie des exigences est une discipline en forte expansion car elle figure de plus en plus comme un gage de qualité et de respect de la demande client par la maîtrise d'œuvre logicielle ¹. Aussi, voit-on émerger dans le monde du logiciel de plus en plus de suites applicatives facilitant la saisie et le suivi des exigences. Outre le recensement des exigences et leur affinement avec le client, c'est également leur respect dans le produit élaboré qui intéressent les deux parties. Ainsi, est-il important pour la maîtrise d'œuvre d'être capable de prouver que telle ou telle exigence est bien couverte par le produit. On parle alors de « **couverture des exigences** ». La « **traçabilité des exigences** » désigne, quant à elle, la faculté à conserver la trace des liens entre les exigences et les différents artefacts utilisés par l'ingénierie logicielle.

La traçabilité des exigences peut donc se décliner à plusieurs niveaux parmi lesquels :

- le lien entre les exigences et les cas d'utilisation
- le lien entre les exigences et les tests fonctionnels
- le lien entre les exigences et le code
- le lien entre les exigences et les tâches des membres du projet
- le lien entre les exigences et les bogues (suivi des anomalies)
- le lien entre les exigences et les livrables (suivi de versions)

Des outils de modélisation offrent déjà la possibilité de lier des exigences à des cas d'utilisation ([Obeo Designer](#), [Enterprise Architect](#), etc.), d'autres de lier les exigences et les tests fonctionnels ([TestLink](#)).

Dans ce présent document, nous nous intéresserons à la traçabilité des exigences dans le code qui représente le niveau le plus proche entre les exigences et leur implémentation effective.

Traçabilité des exigences dans le code

La traçabilité des exigences dans le code consiste à tracer le lien entre des exigences et des parties du code. Dans cette étude, nous supposons que *les exigences sont stockées dans un référentiel d'exigences* (base de données, fichier plat, etc.) et accessibles via des services permettant leur consultation. La nature et le détail de ces services (REST, JSON, Web-Services, API Java, etc.) ne fait pas partie de ce document. On pourrait par exemple se référer au [service OSLC \(Open Services for Lifecycle Collaboration\) des exigences](#) promu notamment par Coclico.


Par ailleurs, cette étude restreindra son champ d'investigation sur la traçabilité des exigences dans du code Java TM qui est l'un des langages principaux utilisés par les projets menés à bien par Bull et donc un des centres d'intérêts de Bull.

Métriques

La traçabilité des exigences dans le code n'a d'intérêt que si l'on peut en extraire des informations intéressantes permettant d'améliorer le suivi des exigences dans un projet.

Les informations ou métriques qui ont été retenues afin d'améliorer la gestion des exigences sont

¹ http://www.processimpact.com/reqs_book/reqs_book_toc.html

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

les suivantes :


- liste des exigences couvertes par le code source : avec notamment la capacité d'expliquer le lien entre une exigence et le code (i.e. visualiser le lien entre une exigence et les classe Java TM implémentant la réponse à une exigence).
- liste des exigences non couvertes par le code
- liste d'exigences déclarées dans le code et qui ne seraient pas enregistrées dans le référentiel des exigences. Cette liste permet notamment d'identifier des éventuelles anomalies dans la gestion des exigences (exigences obsolètes implémentées dans le code, code non conforme au référentiel d'exigences, etc.)

De ces listes, on peut en déduire un certain nombre de statistiques qui peuvent être exploitées par la gestion du projet :

- nombre total d'exigences couvertes par le code
- nombre total d'exigences non couvertes par le code
- pourcentage d'exigences couvertes par le code
- pourcentage d'exigences non couvertes par le code
- nombre « d'anomalies » de traçabilité détectées (exigences obsolètes dans le code, etc.)

Par ailleurs, si l'on applique la traçabilité des exigences dans le code dans les tests unitaires du code, on peut également en déduire les métriques suivantes :

- pourcentage de couverture des exigences par les tests unitaires

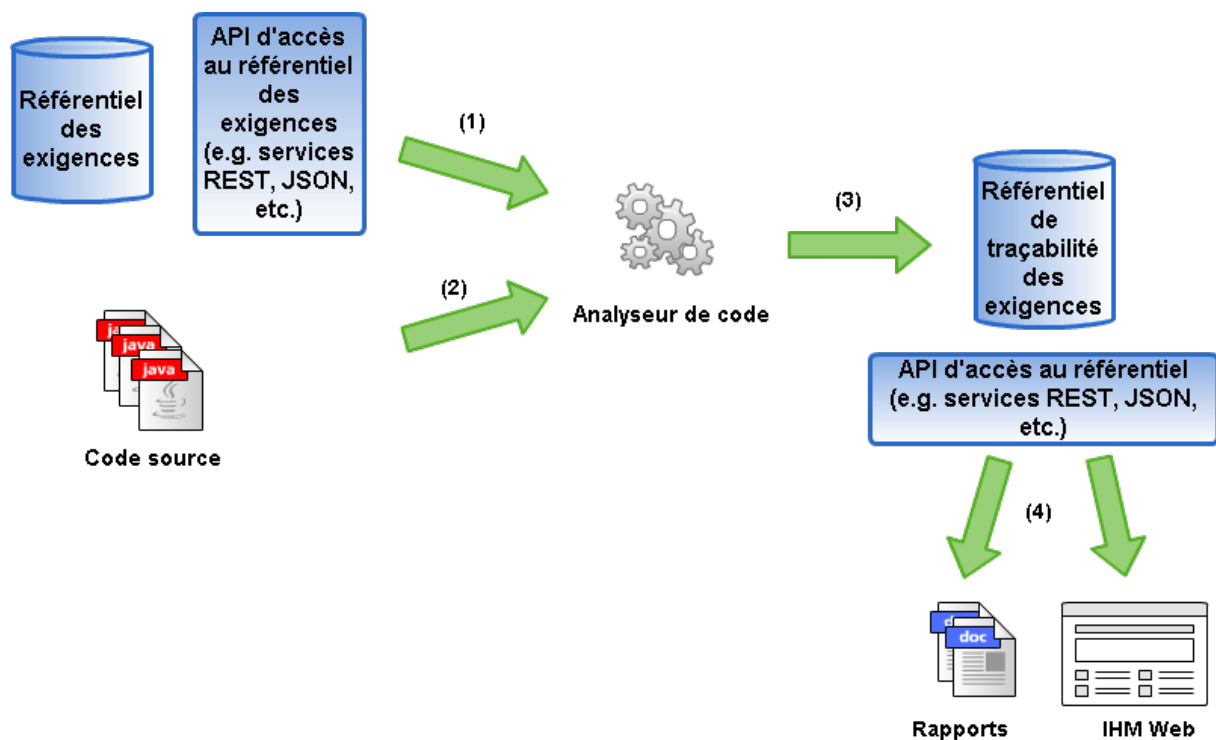
	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

3 Études de solutions

Principe général


De part sa nature, une exigence doit être unique ² et possède un identifiant unique qui permet de la distinguer d'une autre exigence. On peut donc se servir de cet identifiant pour faire le lien entre l'exigence et une partie de code. Par ailleurs, l'extraction des métriques décrites dans le chapitre *Métriques* nécessite l'utilisation d'un outil d'analyse de code qui permettra de faire la relation entre le code et l'identifiant de l'exigence. La volumétrie du code et du référentiel d'exigences pouvant se révéler importante, il convient que l'analyse de code et l'extraction de la traçabilité s'effectuent de façon ponctuelle (tâches planifiées, tâches d'intégration continue, etc.) et que le résultat soit stocké afin, par contre, de permettre une consultation des métriques à tout moment.

L'« architecture générale » pour la mise en œuvre de la traçabilité peut se résumer alors avec le schéma suivant :



- (1) : l'analyseur de code récupère la liste des exigences (identifiants, etc.)
- (2) : l'analyseur de code analyse le code pour en extraire les identifiants des exigences qui sont associés aux sources
- (3) : l'analyseur de code alimente un référentiel de traçabilité des exigences qui permet de faire le lien entre le code source et les exigences. A minima, les données à stocker seraient :
 - l'identifiant de l'exigence

² http://www.processimpact.com/reqs_book/reqs_book_toc.html

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

- le nom du fichier source (voire le chemin complet depuis la racine du répertoire racine) ou le nom qualifié de la classe Java TM (i.e. le nom du package + le nom de la classe)
- le numéro de ligne dans le code source ou le nom de la méthode implémentant l'exigence
- (4) : la traçabilité des exigences dans le code est exposée de manière variée (Interface Homme Machine IHM Web, rapports, etc.)

Le nerf de la traçabilité des exigences dans le code repose principalement sur la façon d'associer l'identifiant d'une exigence avec une partie de code. Les solutions présentées par la suite se focalisent sur cette problématique.

Utilisation de commentaires du gestionnaire de sources

Dans le cadre d'un projet, le code source est naturellement hébergé par un gestionnaire de source de type [Subversion \(SVN\)](#), Git ou autres.

Ces gestionnaires de sources permettent d'ajouter un message lors d'un commit des sources. On pourrait donc imaginer que le développeur soit astreint à ajouter un commentaire particulier permettant d'identifier (par convention) que le commit en question a trait à l'implémentation d'une exigence particulière. Par exemple :

```
[REQ-xxx]: commentaire de commit3 4
```

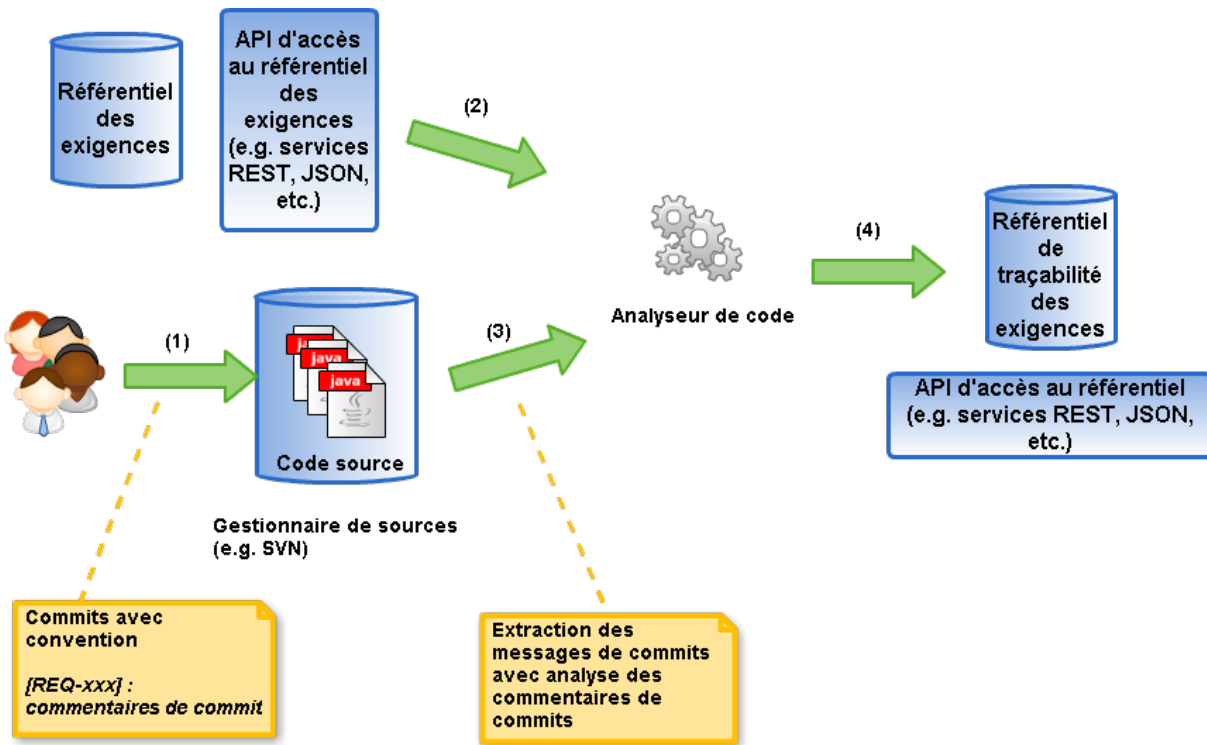
où [REQ-] constituerait le pattern permettant à l'analyseur de code d'identifier la déclaration d'une référence d'exigences

où xxx serait l'identifiant de l'exigence à implémenter.

L'outil d'extraction de traçabilité des exigences serait alors en charge d'analyser les commentaires de commit de code afin d'en déduire le lien entre les exigences et le code.

³ des propriétés similaires existent pour le cas des bogues mais ce sont des propriétés svn particulières implémentées spécifiquement (cf. [Integrating Subversion with your Issue Tracking System](#))

⁴ Un outil comme [Mylyn](#) et son extension svn permet de proposer au développeur un message de commit pré-rempli avec l'en-tête de la tâche Mylyn, facilitant ainsi le travail du développeur




- (1) : les développeurs « committent » code avec une convention de nommage particulière pour les messages de commit
- (2) : l'analyseur de code récupère la liste des exigences (identifiants, etc.) du référentiel des exigences
- (3) : l'analyseur de code analyse les messages de commit pour en extraire le lien avec les exigences. Dans le cas de SVN, il peut également compléter son analyse avec la fonction d'annotation ^{5 6 7} qui permet en plus d'extraire les numéros de lignes changées dans le fichier
- (4) : l'analyseur de code alimente un référentiel de traçabilité des exigences qui permet de faire le lien entre le code source et les exigences

Avantages	
« non-pollution du code »	le code source n'est pas impacté par la gestion des exigences, i.e. la gestion des exigences ne nécessite pas la modification du code source

5 http://wiki.greenstone.org/wiki/index.php/Useful_SVN_Commands#svn_log_and_annotate

6 <http://svn.collab.net/subclipse/help/index.jsp?topic=/org.tigris.subversion.subclipse.doc/html/reference/svn-annotations.html>

7 <http://markphip.blogspot.com/2006/12/subclipse-live-annotations.html>

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011


Inconvénients	
adhérence vis-à-vis de l'outil de gestion de source	adhérence au gestionnaire de sources. La fonctionnalité d'extraction des informations (annotations, etc) dépend du gestionnaire de source. Il faudra donc proposer autant d'implémentations pour l'analyse des messages de commits et annotations que d'outils de gestion de sources supportés dans la forge
consolidation des résultats pouvant être difficile	le gestionnaire de source conservant l'historique de tous les changements, certains peuvent s'avérer obsolètes. On peut ainsi imaginer un commentaire déclarant un lien avec une exigence à un instant t et qu'à un instant t+1, ce code ne soit plus lié à l'exigence (dû par exemple à un refactoring). L'analyse et la consolidation des résultats pour la traçabilité des exigences pourrait s'avérer alors complexe

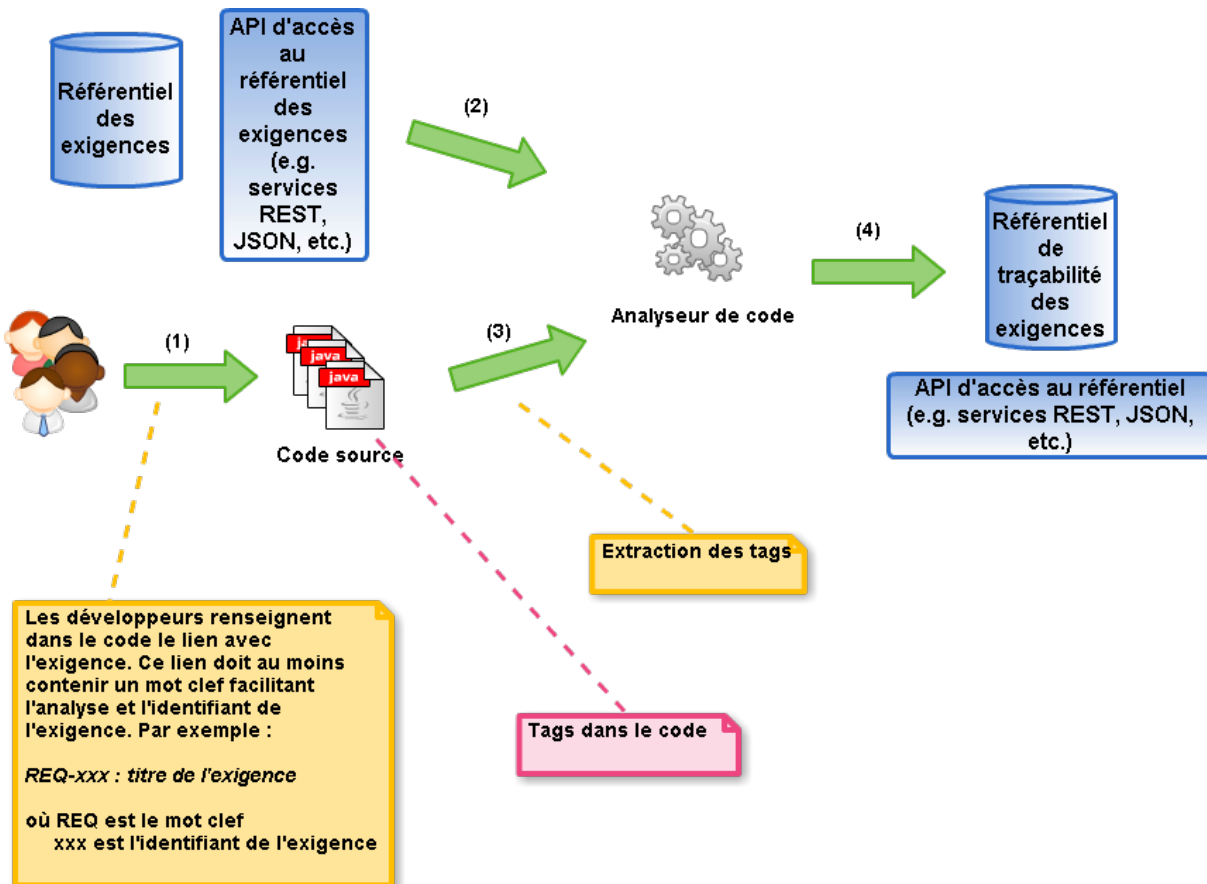
Tags dans le code

Principe


Contrairement à la solution précédente qui repose sur l'utilisation des fonctionnalités du gestionnaire de source, le principe de tags dans le code repose sur la mise en place directe dans le code de tags (ou étiquettes) permettant de relier une exigence à une partie du code.

De façon similaire à la solution décrite dans *Utilisation de commentaires du gestionnaire de sources*, la déclaration d'un tel tag doit contenir au moins un mot clef ou pattern permettant d'identifier la déclaration d'une liaison avec une exigence et au moins l'identifiant de l'exigence. Si l'on souhaite rendre plus « lisible » cette liaison, on pourrait éventuellement ajouter le titre de l'exigence.

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011



- (1) : les développeurs ajoutent un tag de référence de l'exigence dans leur code en respectant une convention de nommage particulière permettant à l'analyseur de code d'identifier le tag comme étant une référence à une exigence
- (2) : l'analyseur de code récupère la liste des exigences (identifiants, etc.) du référentiel d'exigences
- (3) : l'analyseur de code analyse le code source pour en extraire le lien avec les exigences. L'extraction des liens se basant sur le pattern défini par convention
- (4) : l'analyseur de code alimente un référentiel de traçabilité des exigences qui permet de faire le lien entre le code source et les exigences

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

Avantages	
indépendance vis-à-vis du gestionnaire de sources	le tag étant directement dans le code, il y a indépendance vis-à-vis du gestionnaire de source car le traitement s'effectuera directement sur le code source et non via l'intermédiaire d'un outil tiers (qui peut différer d'un projet à l'autre)
analyse de code indépendante du gestionnaire de source	l'analyse de code peut être lancée sans avoir à se connecter au gestionnaire de sources pour récupérer les informations idoines
consolidation des résultats « immédiat »	la présence du tag dans le code conditionne sa liaison avec une exigence. Si le tag disparaît, cela signifie alors que le code n'implémente plus l'exigence. La traçabilité des exigences serait donc plus aisée par rapport à la solution décrite dans le chapitre <i>Utilisation de commentaires du gestionnaire de sources</i> .

Inconvénients	
« pollution du code »	l'insertion d'un tag dans le code peut éventuellement dégrader la lisibilité du code si ce tag est de taille importante. La limitation à l'identifiant de l'exigence et éventuellement au titre de l'exigence peut limiter cet effet

JavaDoclets

Dans le cadre d'un projet Java TM, l'insertion d'un tag pourrait s'effectuer sous la forme d'un commentaire javadoc ou doclet qui pourrait permettre la déclaration de la référence avec l'identifiant de l'exigence.

```


/**
 * Create a new user
 * @param user the user to create
 * @return the created user
 * @throws UserServiceException
 * @requirement USER-0001: le système doit permettre la sauvegarde d'un
utilisateur
 * @requirement BR-0001: email l'adresse mail d'un utilisateur doit être
conforme au pattern d'un email standard.
 */
void createUser(User user) throws UserServiceException;

```

L'analyseur de code pourrait alors se baser sur l'API des [Doclets](#) ⁸ pour réaliser l'analyse et l'extraction des informations. L'analyseur de code pourrait être couplé au [plugin Maven JavaDoc](#) qui offre la possibilité d'ajouter nos propres extensions de doclets ⁹. L'extraction pourrait alors être réalisée lors du lancement d'une tâche Maven particulière, qui pourrait notamment être déclenchée en intégration continue.

⁸ <http://download.oracle.com/javase/6/docs/technotes/guides/javadoc/doclet/overview.html>

⁹ <http://maven.apache.org/plugins/maven-javadoc-plugin/examples/alternate-doclet.html>

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

Avantages	
intégration dans la documentation du code	le tag étant un tag de documentation, la référence à l'exigence ferait alors aussi partie de la documentation javadoc. Par rapport à l'objectif premier, cela constituerait un « plus ». Toutefois, c'est bien la traçabilité dans le code même qui prime
Utilisation d'une API existante	la javadoc offre une API qui permet l'extraction des doclets. Le développement de l'analyseur de code pourrait donc s'appuyer sur cette API ; ce qui éviterait le développement d'un parseur ad-hoc

Inconvénients	
précision de l'implémentation de l'exigence	la précision concernant l'implémentation de l'exigence demeure au niveau de la méthode ou de la classe suivant la disposition du tag. Ceci peut être considéré comme suffisant si les bonnes pratiques de codage sont respectées (méthode à une responsabilité)

Tags de commentaire

Une seconde possibilité consisterait à insérer le tag de référence à une exigence comme commentaire dans le code. Pour le tag, on pourrait prendre une convention du type :

```
// REQ-xxx: <titre de l'exigence>
```

où // serait le caractère de commentaire

où REQ- serait le pattern permettant d'identifier la déclaration d'un tag de référence à une exigence

où xxx serait l'identifiant de l'exigence


Avec la contrainte que le tag soit déclaré sur une seule ligne.

Avantages	
précision de l'implémentation de l'exigence	la précision concernant l'implémentation de l'exigence serait cette fois-ci liée au numéro de ligne dans le code. On pourrait alors « descendre d'un niveau » en n'étant plus cantonné au niveau de la classe ou de la méthode

Inconvénients	
Développement d'un parseur ad-hoc	l'outil d'analyse nécessite le développement d'un parseur ad-hoc (par rapport à la solution <i>JavaDoclets</i>). Toutefois, le pattern proposé étant assez simple, de facto, le parseur devrait l'être également


Conclusion

La solution reposant sur l'utilisation des fonctionnalités du gestionnaire de source offre l'avantage

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

de ne pas « polluer » le code. Toutefois, sa mise en œuvre peut s'avérer complexe car il faut tracer le lien effectif avec l'exigence qui peut disparaître au cours du temps.

La solution consistant à insérer un tag de référence de l'exigence dans le code constitue de notre point de vue la solution la plus simple à mettre en œuvre et sans inconvénients majeurs. La présence du tag dans le code permet à l'analyseur de code d'être lancé sans être connecté au gestionnaire de source et rend par conséquent la solution indépendante de l'outil utilisé pour la gestion des sources. La présence effective du tag dans le code détermine son lien avec l'exigence.

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

4 Pistes techniques

Implémentation

Cette section présente deux pistes techniques qui pourraient servir de support à une implémentation effective.

Les deux pistes adressent en partie l'implémentation de la solution consistant à insérer un tag de référence de l'exigence dans le code. Comme mentionné précédemment, l'étude se focalise sur des projets Java TM et notamment dont la construction repose sur l'utilisation de Maven.

Plugin Maven TagList

La première solution consisterait à enrichir le plugin [Maven TagList](#) permet déjà d'analyser dans des sources Java TM les tags de commentaires courants comme `// TODO` ou `// FIXME`.

L'enrichissement consisterait :

- à ajouter le pattern permettant d'identifier une référence d'exigences (par exemple, `// REQ-xxx`)
- à effectuer un traitement pour enregistrer le numéro de ligne dans laquelle le tag a été déclaré
- à enregistrer en base de données (via par exemple, une API exposée en REST ou Web-Service) le lien entre le code et l'exigence

Avantages	
utilisation d'un plugin maven existant	une partie du code est déjà mis en place (mécanique Maven et parsing du code, lien entre la ligne de code et le tag de référence). Il « suffirait » alors d'enrichir ce code avec les fonctionnalités idoines
possibilité d'intégration dans des sites Maven	le plugin Maven TagList offre la possibilité de générer des rapports Maven qui s'intègre dans un site Maven. Il serait possible d'exploiter cette mécanique pour afficher des rapports exposant la traçabilité des exigences dans le code sur le même modèle

Développement d'un plugin Sonar

[Sonar](#) est un outil open-source d'analyse de métriques de code. Via un plugin ¹⁰, il se couple assez naturellement avec l'outil d'intégration continue [Hudson](#) / [Jenkins](#).


Sonar possède un ensemble d'APIs qui permettent de déclarer de nouvelles métriques ¹¹ et de les exposer en web-services ¹². Par ailleurs, cet outil offre la possibilité de l'étendre via un mécanisme de plugins ¹³. L'extension peut aussi bien porter sur l'introduction de nouvelles métriques que sur

¹⁰ <http://docs.codehaus.org/display/SONAR/Hudson+and+Jenkins+Plugin>

¹¹ <http://docs.codehaus.org/display/SONAR/Extend+coding+rules>

¹² <http://docs.codehaus.org/display/SONAR/Web+Service+API#WebServiceAPI-Metrics>

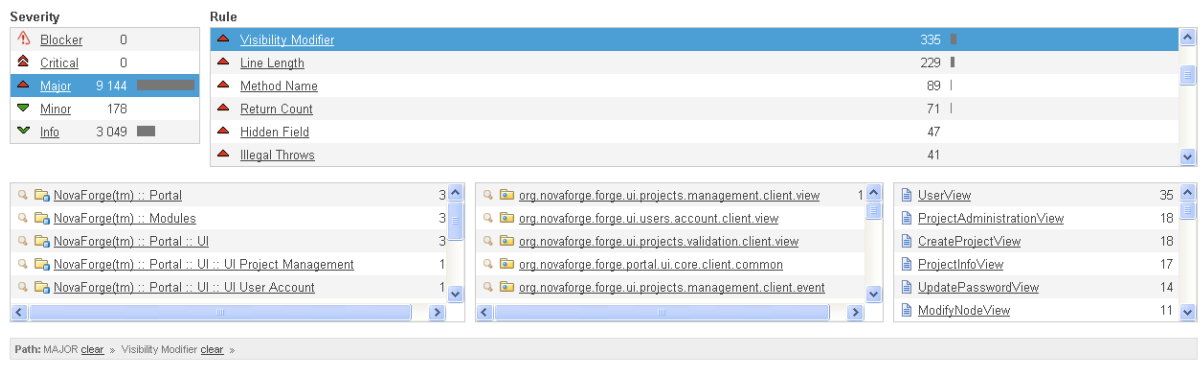
¹³ <http://docs.codehaus.org/display/SONAR/Coding+a+plugin>

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

le rendu de ces métriques dans l'IHM de Sonar. Le stockage de la plupart de ces métriques sont d'ailleurs pris en charge par les APIs bas niveau de Sonar.


Avantages	
stockage des métriques	le stockage des métriques est délégué à Sonar. Le stockage du lien entre le code et l'exigence serait donc pris en charge par Sonar
réalisation d'IHM	le développement d'une IHM exposant la traçabilité des exigences dans le code reposerait sur le guide méthodologique de développement de plugins de Sonar
intégration de l'IHM de traçabilité des exigences	<p>de l'IHM de traçabilité des exigences serait intégrée dans un outil qui expose d'autres métriques liées au code (checkstyle, findbugs, etc.). L'intégration des métriques de traçabilité des exigences avec les autres métriques de code dépasserait donc le cadre de l'ingénierie des exigences et faciliterait l'analyse des axes d'amélioration autour du code</p> <p>plusieurs IHMs de Sonar repose sur le lien entre la ligne de code et la métrique : il serait alors envisageable de proposer une IHM comparable pour gérer la traçabilité des exigences</p>

A titre d'exemple, voici une copie d'une IHM Sonar exposant le lien entre une métrique et les classes java afférentes :

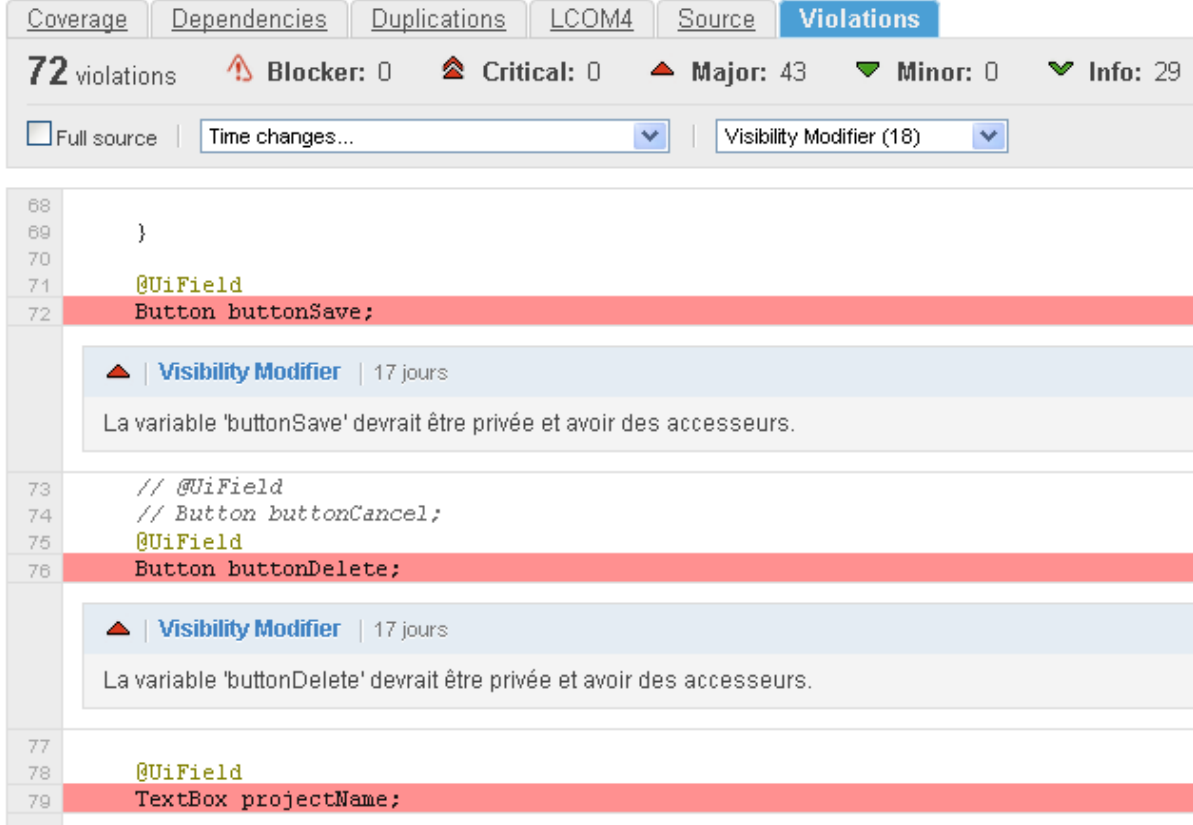


On pourrait imaginer de bénéficier d'une représentation similaire pour la traçabilité des exigences dans le code.

A titre d'exemple également, voici une copie d'une IHM Sonar liant une métrique au code source :

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

org.novaforge.forge.ui.projects.management.client.view.CreateProjectView



Coverage Dependencies Duplications LCOM4 Source **Violations**

72 violations Blocker: 0 Critical: 0 Major: 43 Minor: 0 Info: 29

Full source | Time changes... | Visibility Modifier (18)

```

68
69     }
70
71     @UiField
72     Button buttonSave;

```

Visibility Modifier | 17 jours

La variable 'buttonSave' devrait être privée et avoir des accesseurs.

```

73     // @UiField
74     // Button buttonCancel;
75     @UiField
76     Button buttonDelete;

```

Visibility Modifier | 17 jours

La variable 'buttonDelete' devrait être privée et avoir des accesseurs.

```

77
78     @UiField
79     TextBox projectName;

```

Ainsi, pourrait-on imaginer un autre onglet « Exigences » qui pourrait afficher la référence de l'exigence dans le code.

Intégration avec le poste de développement


Afin de faciliter le travail du développeur en charge de l'implémentation des exigences, on pourrait également imaginer une assistance qui aiderait le développeur à insérer le tag de référence de l'exigence dans le code.

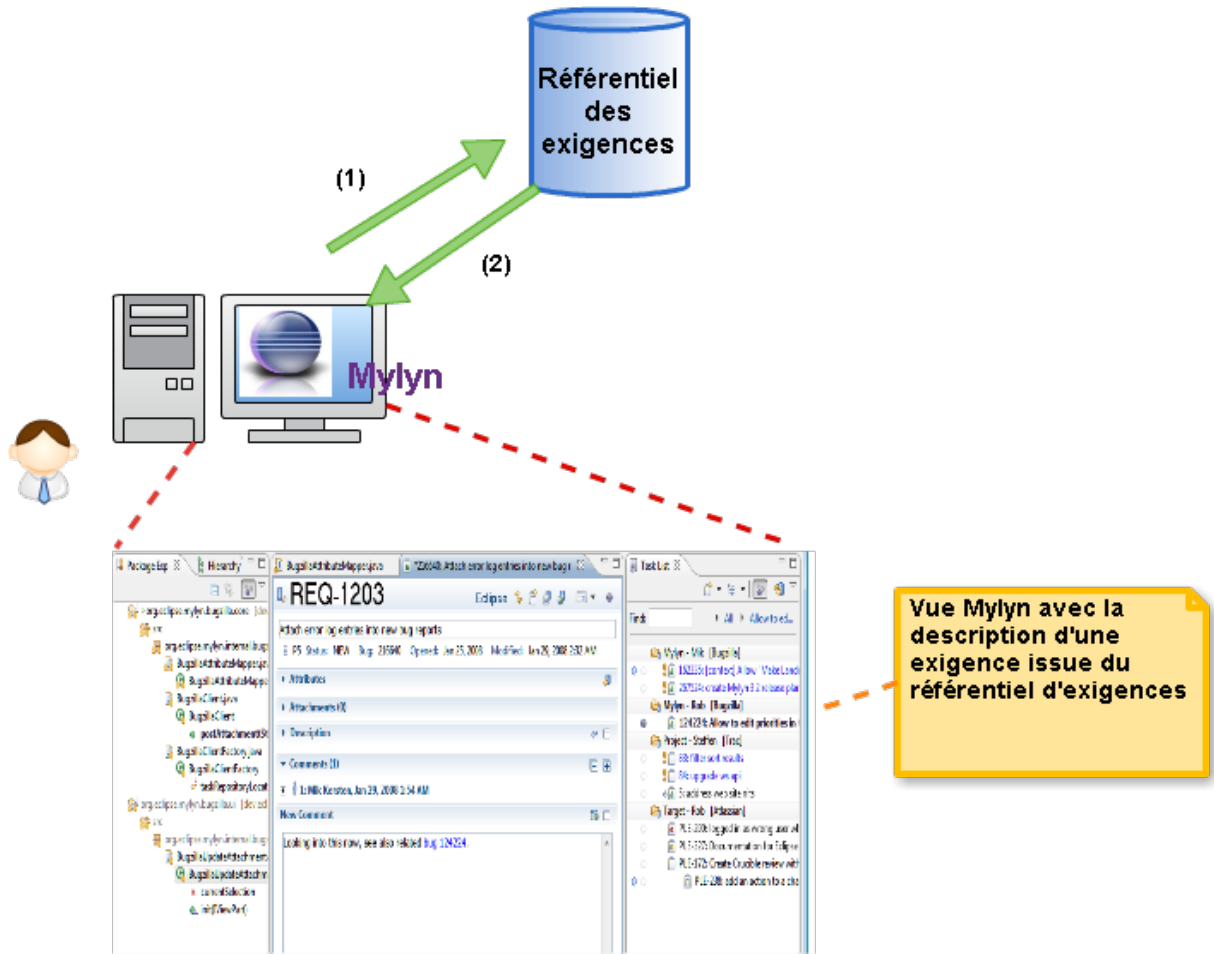
Un plugin Eclipse reposant sur Mylyn pourrait ainsi permettre de :


- se connecter au référentiel des exigences de la forge
- récupérer les identifiants des exigences
- proposer par auto-complétion dans le code d'insérer le tag de référence d'une exigence, réduisant ainsi les erreurs d'insertion d'un tag de référence mal rédigé et non conforme à la convention (problème de casse, de notation, etc.)

Ainsi, le développeur posséderait-il la liste des exigences du projet directement dans son poste de développement et n'aurait pas besoin d'un outil tiers pour s'y référer ¹⁴.

¹⁴ Mylyn permet également d'enregistrer le temps passé sur une tâche. Dans le cas des exigences, cela permettrait de savoir le temps passé par un développeur sur l'implémentation d'une exigence et donc en déduire de nouvelles métriques pour la gestion de projet...

	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011



	Titre du document : WP 4.6 - Traçabilité des exigences dans le code
	Référence : WP 4.6 - Étude d'une solution de traçabilité des exigences dans le code
	Version du 21/07/2011

5 Annexes

Références

Enterprise Architect <http://www.sparxsystems.com/>

Git <http://git-scm.com/>

Hudson <http://hudson-ci.org/>

Java Doclets Overview :

<http://download.oracle.com/javase/6/docs/technotes/guides/javadoc/doclet/overview.html>

Java Doclets <http://download.oracle.com/javase/1.5.0/docs/guide/javadoc/doclet/spec/index.html>

Jenkins <http://jenkins-ci.org/>

Maven TagList Plugin <http://mojo.codehaus.org/taglist-maven-plugin/index.html>

Mylyn <http://eclipse.org/mylyn/>

OSLC (Open Services for Lifecycle Collaboration) : <http://open-services.net/bin/view/Main/RmSpecificationV2>

Obeo Designer <http://www.obeo.fr/pages/obeo-designer-for-is/fr>

Subversion (SVN) <http://subversion.apache.org>

Sonar <http://www.sonarsource.org/>