Geasysecu

Rapport étendu et documentation

Emmanuel Nhan

Aout 2011

Table des matières

Introduction 2
1. État de l'art 2
2. Cahier des charges 3
3. Implémentation de l'extension 5
Pile partagée 6
Implémentation de policy-domain 6
Implémentation de securitypolicy-domain 7
Implémentation de domain 8
Architecture serveur 8
Architecture client 10
4. Gestion de la configuration 10
5. Cas d'utilisation ..... **Erreur ! Signet non défini.**
6. Perspectives d'évolution 11
1. Références bibliographiques 1
2. Glossaire 2

Introduction

La sécurité est une thématique extrêmement importante dans le cadre des architectures SOA et de l'ouverture par une entreprise de son système d'information à des partenaires extérieurs. Il est vital pour une compagnie que l'ouverture de son système d'information soit régit par une ligne métier consistante et cohérente. L'aspect sécurité doit être un des facteurs qui sont là pour garantir que l'ouverture du système d'information suit bien cette ligne. Une des approches possibles pour formaliser cette ouverture est d'utiliser la modélisation des processus. Cela amène à impliquer des business analysts non seulement dans l'élaboration des processus mais aussi dans la définition des contraintes de sécurité s'appliquant sur le système, directement dès la phase de modélisation. Pour cela il faut enrichir les langages graphiques de modélisation, comme BPMN (1) qui ne proposent pas en standard de support pour cet aspect sécurité, et traduire ensuite les informations ajoutées pour une exploitation au runtime : c'est l'objectif de mon PFE.

Le projet a été constitué de deux grandes phases : étude de l'état de l'art et réponse à la question « comment intégrer la sécurité dans BPMN 2.0 ? », puis implémentation de cette réponse sous forme de prototype dans l'éditeur BPMN 2.0 Petals BPM.

1. État de l'art

Le problème posé par le PFE est multidimensionnel. Il concerne à la fois l'aspect visuel et la transformation des modèles. Ces deux aspects induisent le problème de l'outillage.

Pour ce qui concerne l'aspect visuel (5) propose un modèle très simple venant en extension au langage BPMN 1.x. Ces travaux mettent en évidence le concept central de « security requirement », qui est affiné par divers types d'exigences (Privacy, Non-repudiation,…). L'étude associe à chaque exigence un degré d'importance (bas, moyen, haut) et traduit l'association entre les éléments graphiques BPMN et les exigences par des pictogrammes sur les éléments concernés. A l'époque de ce travail, BPMN n'était pas encore destiné à être aussi un langage d'exécution, c'est pourquoi, l'étude est orientée sur l'aspect graphique uniquement. Par ailleurs, le mécanisme utilisé pour étendre le standard BPMN a depuis été modifié et est maintenant bien différent.

En ce qui concerne la traduction (6) propose une extension à BPMN 2.0 et un moyen de traduction des annotations vers des langages standards (XACML, par exemple). Malheureusement, je n'ai pris connaissance de ce dernier que bien trop tard pour pouvoir en tirer bénéfice pour le PFE. Il renferme cependant d'intéressantes idées et il serait possible de l'exploiter à moyen terme.

A un niveau plus technique, des principes de sécurité dans SOA sont définis, notamment par IBM (7). Outre la présentation des solutions IBM, l'ouvrage définit le nécessaire pour comprendre la place de la sécurité dans SOA et met en évidence des concepts, plus liés à l'aspect gouvernance et cycle de vie de l'écosystème SOA (Figure 1). En s'appuyant sur des cas d'utilisation non triviaux pour de grandes organisations, il montre à quelles fins un outillage peut-être développé.

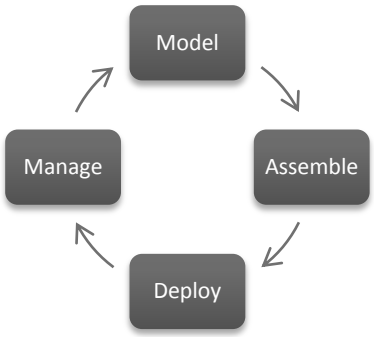

Figure 1- Cycle de vie tiré de (7)

Enfin, des standards ont été définis pour intégrer la sécurité au runtime dans les WS. Deux dialectes XML ont principalement retenu mon attention : les spécifications WS-Policy et WS-SecurityPolicy.

WS-Policy définit un langage flexible permettant d'exprimer des exigences (8). Grâce à une autre spécification, WS-PolicyAttachment, il est possible d'embarquer ces exigences dans les documents WSDL (9) décrivant les WS. Enfin, la spécification WS-SecurityPolicy définit un standard pour exprimer des exigences de sécurité dans le cadre des documents conforme à WS-Policy (10).

Ces standards enrichissent le contrat (décrit en WSDL) entre producteur et consommateur de service pour y intégrer les contraintes de sécurité.

2. Cahier des charges

Suite à cet état de l'art, j'ai décidé que le plugin aurait deux principales fonctions. La première s'inscrit dans l'étape modélisation du cycle de vie IBM (7) et permet d'ajouter les attributs confidentialité, intégrité, non-répudiation, autorisation, authentification et disponibilité, assortis chacun d'un niveau d'importance, appelé aussi niveau d'exigence, sur l'échelle low/medium/high. Ces attributs sont relatifs aux éléments ServiceTask, SendTask et ReceiveTask du modèle BPMN.

La seconde fonctionnalité, qui s'inscrit à l'étape assemble du cycle de vie IBM (7), est axée sur transition vers le runtime. Les solutions Petals Link étant très orientés sur l'utilisation de standards (spécifications de l'OASIS) et de protocoles ouverts (SOAP), j'ai décidé que le plugin rentrerait aussi dans cette logique. L'intégration des technologies WSDL avec le standards BPMN étant de plus en plus aboutie, j'ai considéré pertinent de pousser plus loin et de lier avec les technologies déjà existantes autour des WSDL. Dans le cadre de ce PFE, il s'agit des spécifications WS-Policy et WS-SecurityPolicy. Le but du plugin qui en découle ne sera donc pas de construire un tunnel complètement sécurisé (sécurité de bout en bout) entre service provider et service consummer, mais plutôt de permettre à un utilisateur d'enrichir les WSDL de ses services avec des assertions de sécurité.

L'utilisation du plugin se fait en quatre temps (Figure 2).

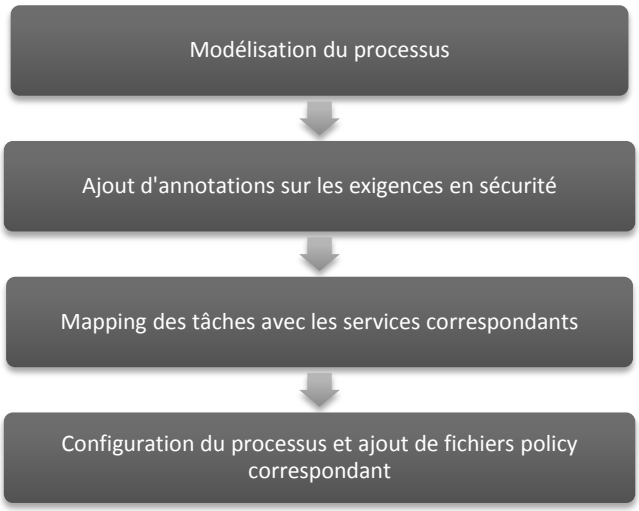

Figure 2 - Workflow d'utilisation du plugin

L'utilisateur initial a un profil plutôt business analyst. Il décrit le processus grâce au langage BPMN 2.0 basique et ajoute des indications sur les besoins en sécurité. Dans ce cas, il associe un niveau d'exigence des attributs de sécurité sur les tâches : c'est l'utilisation de la première fonctionnalité du plugin. L'utilisateur obtient alors un diagramme enrichit par les pictogrammes de sécurité (Figure 3).

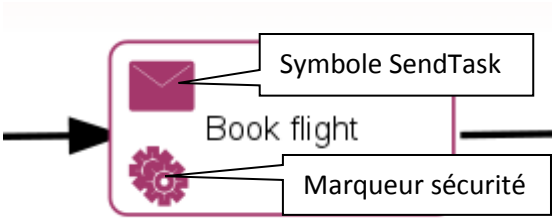

Figure 3 - SendTask avec marqueur de sécurité

Une fois sauvegardés, ces diagrammes permettent de générer des fragments XML intégrant les exigences de sécurité (Listing 1).

```
<bpmn:serviceTask>
    <!-- ... -->
    <bpmn:extensionElements>
        <secu:securitySet>
            <secu:integrity level="medium"/>
    <!-- autres attributs de sécurité -->
        </secu:securitySet>
    </bpmn:extensionElements
</bpmn:serviceTask>
```

Listing 1 - ServiceTask avec extension sécurité spécifiant un niveau médium pour l'exigence d'intégrité

Les deux dernières étapes sont réalisées par un utilisateur averti. Ce dernier va d'abord associer les tâches aux WSDL correspondants, puis ajouter de même les fichiers policy répondant aux exigences spécifiées par le premier utilisateur. Pour cela, l'utilisateur dispose d'un explorateur de policy

permettant de parcourir les exemples de fichiers policy déployées sur le serveur. Grâce à un glisser/déposer, il peut les lier aux tâches concernées. Ces fichiers policy peuvent ensuite être édités par un éditeur type formulaire puis ajustés par un éditeur XML intégré.

Outre l'aspect modélisation, je me suis aussi penché sur l'aspect runtime. L'architecture cible des systèmes d'information après déploiement des processus modélisés dans Petals BPM suivent le schéma par exemple le schéma de la Figure 4. Au runtime, c'est le BC-SOAP du bus Petals qui est chargé d'interpréter les policies dans les WSDL. Cela est déjà possible car le BC-SOAP est construit sur Apache Axis2, pile implémentant les spécifications WS-Policy et WS-SecurityPolicy via d'autres librairies. L'architecture cible intègre d'autres éléments, en particulier les Identity providers, applications spéciales ayant pour rôle de délivrer les identités et les autorisations.

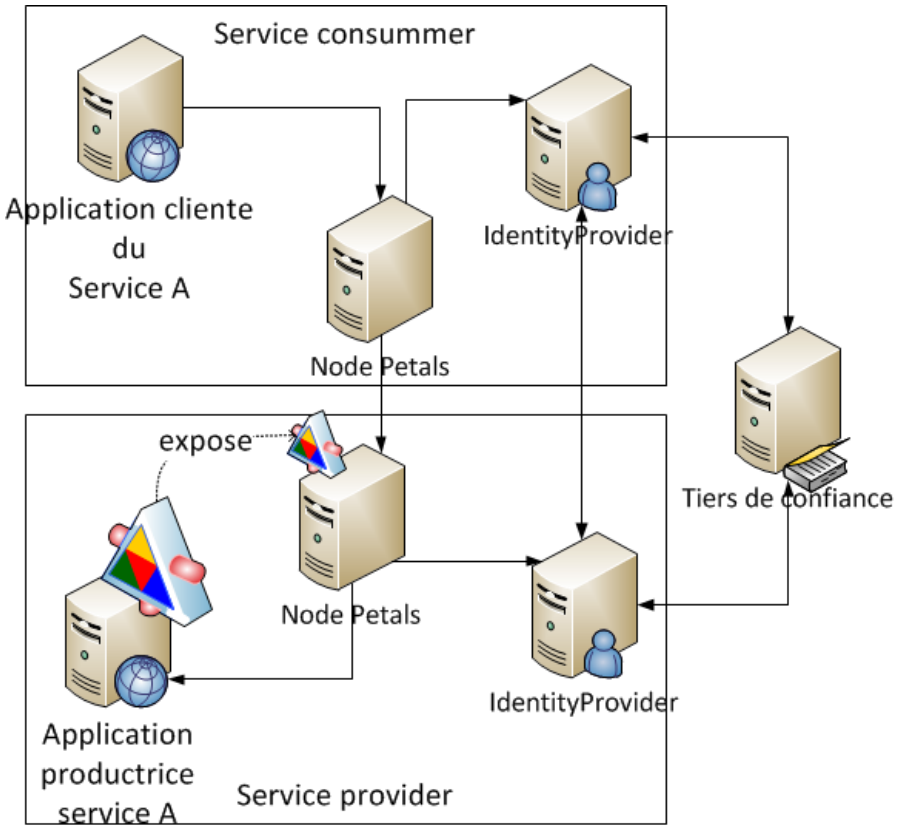

Figure 4 - Exemple d'architecture runtime

3. Implémentation de l'extension

Geasysecu est une application client/serveur écrite entièrement en Java. Le client est une web-application reposant sur Petals BPM et écrite en Java grâce au framework de Google GWT. L'architecture de Geasysecu est directement liée à l'utilisation de ce framework. GWT propose un mécanisme RPC (propriétaire) pour permettre au client et au serveur de communiquer entre eux qui oblige un recours au design pattern Transfert Object. On a donc trois piles : serveur, client et transfert objects.

4. Pile partagée

La pile partagée est le cœur de l'extension. Elle contient l'ensemble des classes modélisant les objets principaux des spécifications WS-Policy et WS-SecurityPolicy ainsi que les classes en charge de l'aspect modélisation. Elle est divisée en trois sous-projets Maven :

- policy-domain ;
- securitypolicy-domain ;
- domain.

Implémentation de policy-domain

Le projet policy-domain consiste en l'implémentation d'un support basique de WS-Policy. Cette implémentation ne reprend pas complètement le schéma décrit par WS-Policy. WS-Policy ne définit en effet qu'un cadre pour manipuler des assertions et des jeux d'assertions sont définis (par exemple, WS-SecurityPolicy). Dans WS-Policy, les assertions provenant de diverses spécifications peuvent se mélanger et se combiner (du moins, en théorie). Pour policy-domain, en revanche, j'ai préféré garder une séparation des concepts de chaque spécification. Ainsi, j'ai introduit un concept de `IPolicyAspect` qui a pour vocation de regrouper toutes les assertions d'une spécification. Tous les aspects sont groupés dans une classe permettant de manipuler les fichiers policy facilement sur le serveur et sur le client via la classe `PolicyFileTO`. Le diagramme des classes UML de la Figure 5 résume ce fonctionnement.

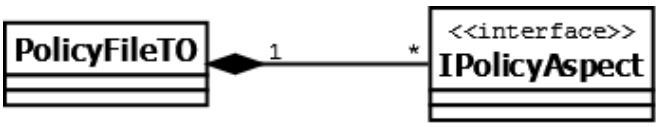

Figure 5 - principales classes de policy-domain

Les instances de `PolicyFileTO` contiennent le nécessaire pour savoir si un fichier policy a été nouvellement créé et son contenu en XML brut, entre autres. Dans le cadre de ce PFE, j'ai uniquement développé la base pour pouvoir utiliser la spécification WS-SecurityPolicy.

Implémentation de securitypolicy-domain

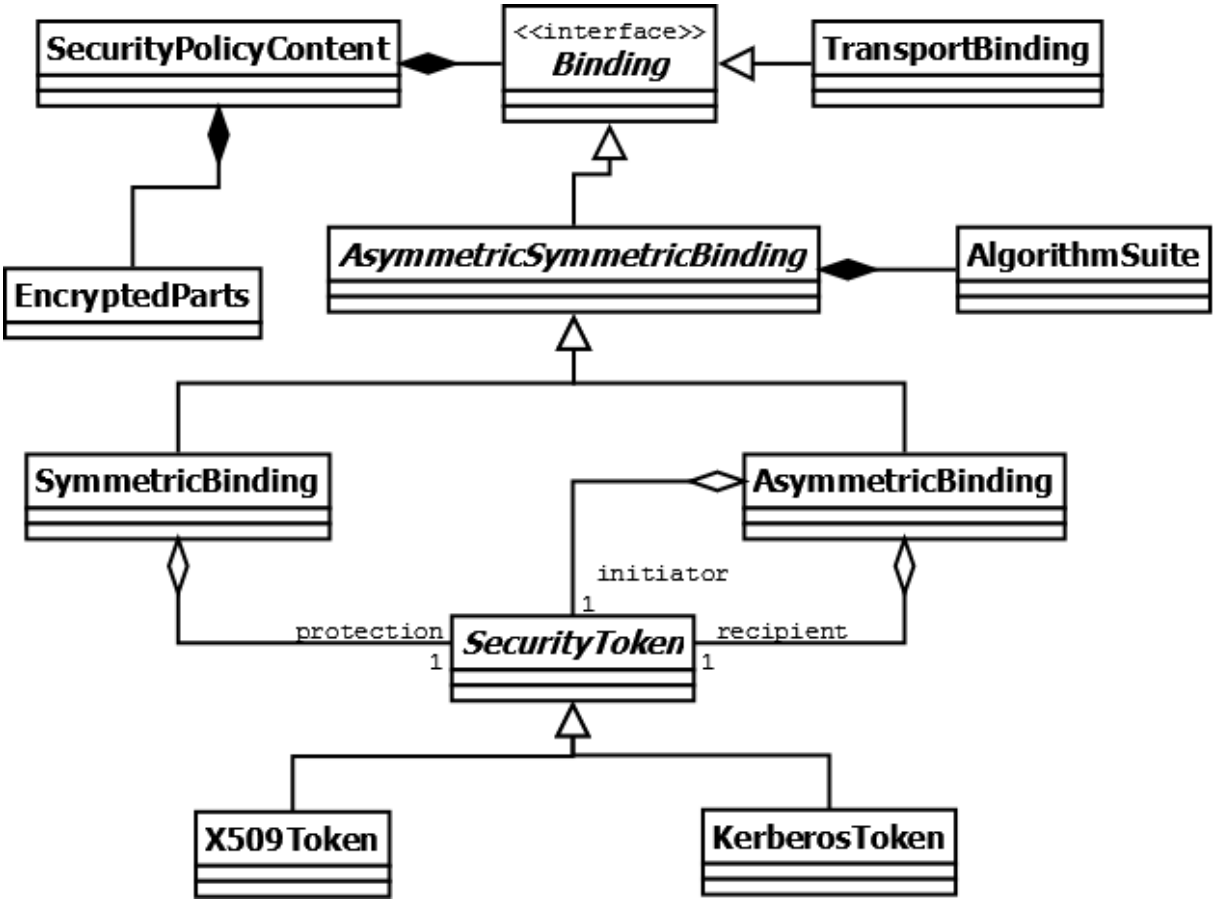

Figure 6 - Diagramme des classes de securitypolicy-domain

Le projet securitypolicy-domain regroupe toutes les classes utiles pour manipuler les assertions de sécurité de WS-Securitypolicy de manière simplifiée. La classe `SecurityPolicyContent` implémente `IPolicyAspect` et concentre tous les aspects sécurité. Dans ce diagramme, on notera l'interface `Binding` qui est là pour permettre l'implémentation de différents mécanismes de sécurité. A ce jour, seul les classes filles d'AsymmetricSymmetricBinding sont implémentées. Le diagramme UML de la Figure 6 illustre les grandes lignes de ce projet.

Implémentation de domain

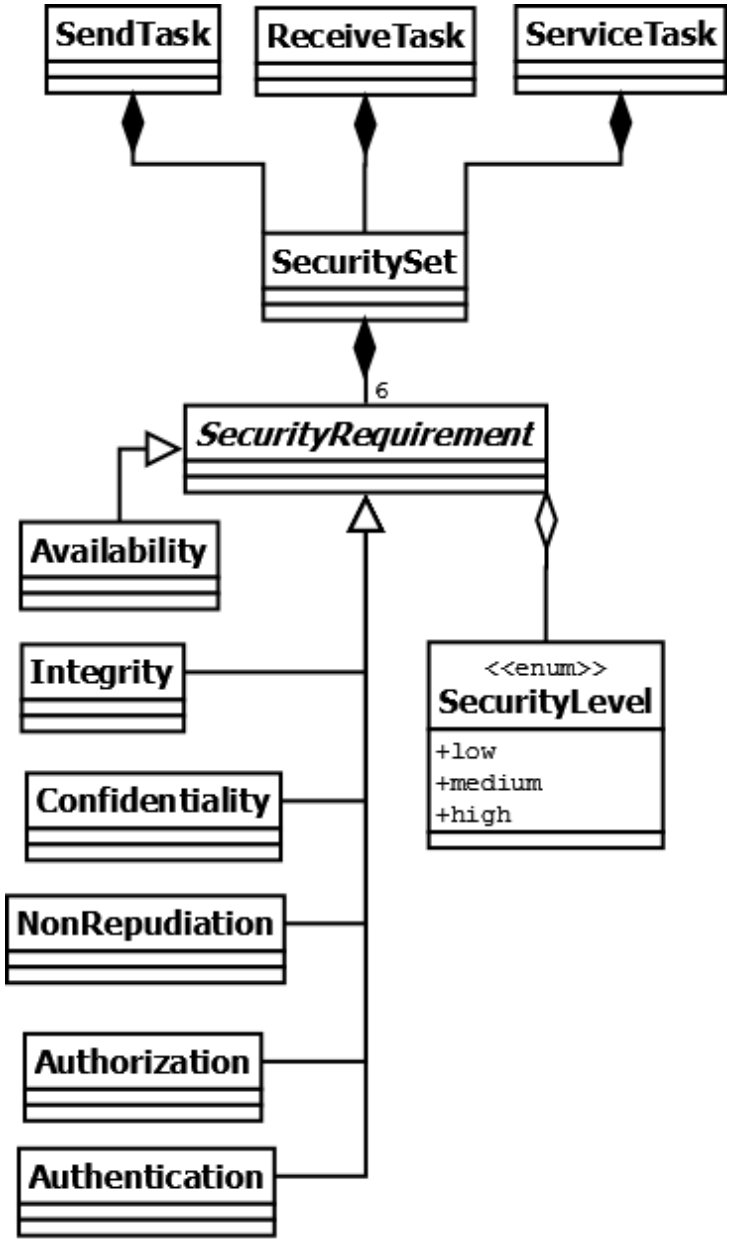

Figure 7 - Diagramme des classes de domain

Le projet domain est le cœur de l'extension, et est destiné à la partie modélisation. Il représente les concepts que l'utilisateur type analyste métier va utiliser pour modéliser la sécurité dans l'éditeur. Le diagramme UML de la Figure 7 résume son architecture et son interfaçage avec les classes du modèle BPMN.

Architecture serveur

La Figure 8 présente l'architecture serveur plus en détail, avec les librairies utilisées, chaque brique étant un projet Maven.

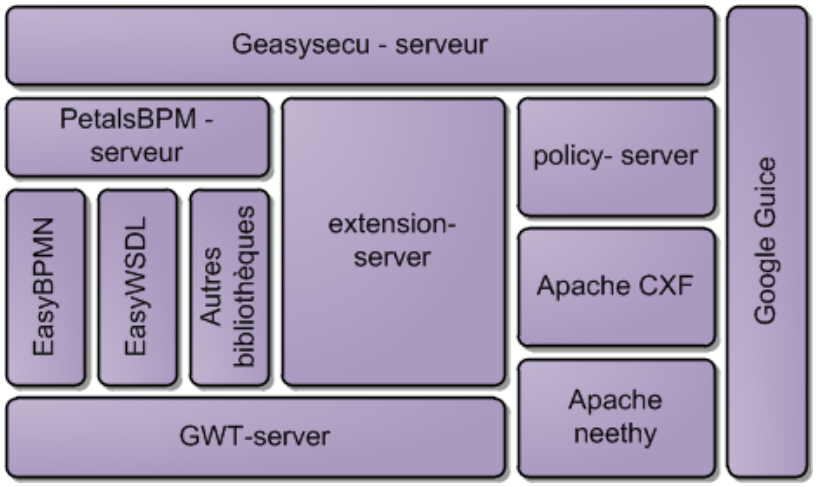

Figure 8 - Architecture serveur

La pile serveur est constituée principalement du composant policy-server. Ce composant a la charge de manipuler les fichiers policy sur le serveur et de les traduire en transfert objects compatibles avec GWT. Il a aussi la responsabilité de la tâche inverse : convertir des Transfert Objects en fichiers policy. Le point d'entrée de ce composant est l'interface `IPolicyRepository`. Dans le cadre du PFE, seule une implémentation système de fichier a été implémentée. Deux autres interfaces (`IServerToClientTranslator` et `IClientToServerTranslator`) décrivent les opérations possibles de traduction (Figure 9).

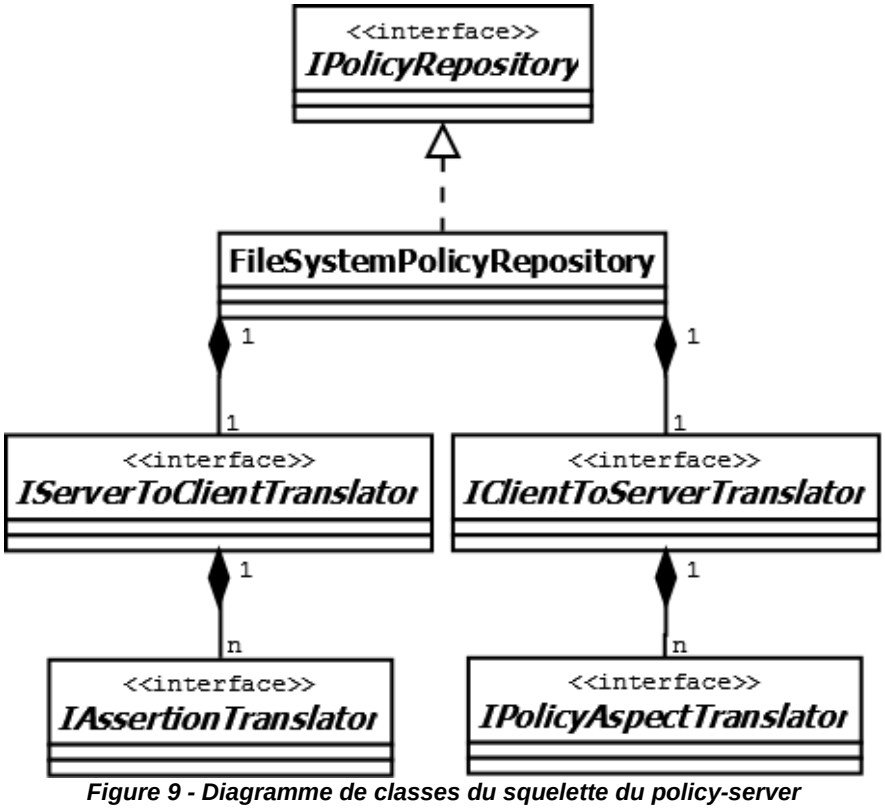

Figure 9 - Diagramme de classes du squelette du policy-server

Les implémentations d'`IAssertionTranslator` ont pour rôle de traduire les assertions parsées par Apache CXF en Transfert Objects correspondant. Les instances d'`IPolicyAspectTranslator` s'occupent de la traduction inverse. Les instances implémentant ces interfaces sont injectées par

Google Guice dans leur conteneur respectif. L'utilisation de ce découplage a pour but de laisser la place à des implémentations pouvant traduire des assertions WS-Policy autre que celles de WS-SecurityPolicy et d'intégrer, par le futur, le plugin avec d'autres produits de Petals Link, comme l'outil de gouvernance.

Architecture client

Côté client, Petals BPM est aussi basé sur une pile de bibliothèques qui ont pour but de produire un studio générique, d'apporter la manipulation de scènes SVG et de gérer de manière uniforme des projets dans l'éditeur (par exemple, un projet BPMN niveau descriptif).

Côté client, l'extension de Petals BPM n'est pas véritablement un plugin dans le sens où elle n'est pas chargeable/déchargeable à chaud (ce fonctionnement n'est pas possible pour le moment avec GWT). La conception est compliquée, d'une part car l'extension n'étend pas le studio mais plutôt un type de projet possible (dans mon cas, un processus exécutable BPMN) et d'autre part parce que la pile client de Petals BPM évolue très vite (les API ne sont pas stables). La partie client se découpe en deux lots : l'édition de policy et l'annotation des diagrammes. Le design pattern MVC est mis en œuvre dans les deux cas. Malheureusement, la partie cliente de Petals BPM n'utilisant pas d'injection de dépendances, l'accès à certains éléments centraux de l'éditeur, comme la factory permettant d'instancier les éléments BPMN est difficile. Ainsi, remplacer cette factory est vite un casse-tête. Ma solution temporaire a donc été de patcher par endroits Petals BPM, puisque les patterns d'injection de dépendance et d'inversion de contrôle ne sont pas encore possibles.

5. Gestion de la configuration

L'outil de construction permettant de construire le projet est Maven.

La gestion de la configuration se fait avec SVN sur la forge de Petals Link. Cependant pour expérimenter, j'ai utilisé git comme gestionnaire de sources en local. Git, grâce à git-svn permet d'utiliser un repository SVN comme serveur de sources pour git.

L'avantage de git est d'autoriser des branches locales qui ne seront pas commitées sur le serveur SVN. L'idée était de faire une branche par fonctionnalité à développer et d'intégrer sur la branche principale une fois chaque fonctionnalité implémentée.

Pour utiliser git avec svn voici quelques commandes de base :

- `git svn rebase` est équivalent à un `svn update`
- `git svn dcommit` est équivalent à un `svn commit`

Enfin, il est nécessaire de connaitre la base des commandes git pour que tout se passe bien :

- `git add` pour ajouter des fichiers au système de gestion des sources
- `git rm` pour en enlever
- `git commit` pour effectuer un commit local
- `git branch` pour travailler avec les branches.

6. Perspectives d'évolution

En l'état (aout 2011), le plugin développé est loin d'être l'outil idéal pour annoter des processus BPMN et y ajouter la prise en compte d'aspects sécurité, puisqu'il s'agit avant tout d'un prototype. En cela, il est perfectible. Sur le plan de l'ergonomie, tout d'abord, l'amélioration du plugin passerait par une manipulation de fichiers policy par une couche de plus haut niveau. Ainsi, l'utilisateur, au lieu d'aller piocher des fichiers policy dans l'explorateur, pourrait indiquer directement une technologie pour la traduction d'une exigence et remplir la configuration de l'exigence.

Au niveau du runtime, le plugin actuel manque de connectivité avec les autres produits tels que le bus Petals. Une interaction avec le Déployeur développé dans le cadre du projet Process 2.0 pourrait par exemple simplifier le déploiement des fichiers policy. L'idéal serait une intégration avec les outils de gouvernance pour aller chercher les politiques déjà déployées.

Enfin, l'aspect « modification d'un existant » est quasiment nul à ce jour. J'ai cependant proposé diverses stratégies pour enrichir des services existants avec de la sécurité dans un autre document.

1. Références bibliographiques

1. **OMG.** *Business Process Model and Notation Version 2.0.* 2011.
2. **Hafner M., Brey R.** *Security Engineering for SOA.* 2009. ISBN 978-3-540-79538-4.
3. **OASIS.** *Reference Architecture for Service Oriented Architecture Version 1.0.* 2008. pp. 88-89.
4. —. Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0. *OASIS.* [Online] 2005. http://www.oasis-open.org/committees/download.php/21111/saml-glossary-2.0-os.html#Identity%20Provider.
5. *A BPMN Extension for the Modeling of Security Requirements in Processes.* **Alfonso Rodriguez, Eduardi Fernandez-Medina, Mario Pianttini.** 2007.
6. *A Security Language for BPMN Process Models.* **Jutta Mülle, Silvia von Stackelberg, Klemens Böhm.** 2011.
7. **IBM.** *Understanding SOA Security design and implementation.* s.l. : IBM redbooks, 2007. p. 14. ISBN 0738486655.
8. **W3C.** Web Services Policy 1.5 - Framework. [Online] 9 4, 2007. http://www.w3.org/TR/ws-policy/.
9. —. Web Services Policy 1.5 - Attachment. [Online] 9 4, 2007. http://www.w3.org/TR/ws-policy-attach/.
10. **OASIS.** WS-SecurityPolicy 1.2. [Online] 07 01, 2007. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html.
11. **Frédérique Biennier, W. Francis Ouedraogo.** *Spécification du méta modèle de sécurité - Projet Process 2.0.* 2011.

2. Glossaire

Authentification (authentication):
Assurance de la véracité de l'identité des pairs ;
Autorisation (authorization):

Assurance que seuls les pairs légitimes peuvent effectuer les interactions et accéder aux ressources;

Confidentialité (confidentiality):
Assurance que les pairs non autorisés ne peuvent pas comprendre le contenu des messages ou des échanges.
Consommateur de service (service consummer) :
Pair qui va être client de services.
Disponibilité (availability):
Assurance que les services sont accessibles — dans le sens où le système est fiable, robuste;
Fournisseur d'identité (identity provider) (4):
Pair de confiance qui assure le service de l'authentification des pairs.
Intégrité (integrity):
Assurance que l'information échangée entre les pairs n'est pas altérée;
Non-répudiation (non-repudiation): Assurance qu'un pair impliqué dans un échange ne puisse nier sa participation au dit échange.
Pair (peer) :
Un pair est un usager du système d'information. Cela peut être une organisation, une personne physique, ou bien une application.
Producteur de service (service provider) :
Pair (applicatif) qui met à disposition un service.