

---

*Mounir Lallali, Prof. Burkhardt Wolff*  
Parc Orsay Université  
4, rue Jacques Monod  
Building H / Room 012

## Projet Génie Logiciel

Date : 22.09.2009

**A faire :** en binôme

**A rendre le :** 17 Décembre 2009

**Forme du compte rendu :**

1. rapport en doc ou pdf à envoyer par e-mail à [mounir.lallali@lri.fr](mailto:mounir.lallali@lri.fr)
2. archive (zip, rar ou tar) du code Lotos à envoyer à la même adresse
3. démo sur machine : date à fixer ultérieurement

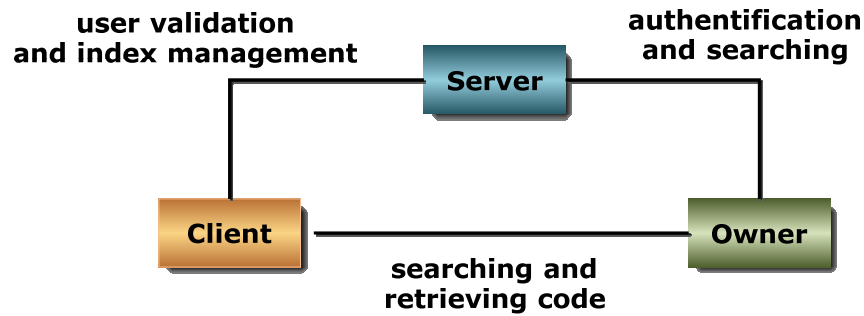
## 1 Le service CodeShare

Il vous est demandé de modéliser un exemple de service Web peer-to-peer qui permet de partager du code source, appelé « **CodeShare Service Network** ». Sa spécification informelle ainsi que sa description WSDL sont partiellement décrites dans le livre : Snell & al., Programming Web services with SOAP, O'Reilly 2002.

La spécification informelle ci-dessous est simplifiée et complétée. Elle a été volontairement laissée très proche de la spécification originale, qui n'a pas été détaillée en vue d'une description LOTOS. Cela signifie qu'il y a besoin d'une réflexion approfondie sur les processus, les ports, et leurs synchronisations avant d'élaborer une architecture et d'écrire la spécification. Comme dans les cas réels, la spécification informelle contient des détails inutiles pour la modélisation.

### 1.1 Spécification informelle

plus qu'un simple service Web, le service **CodeShare** présente plusieurs interfaces qui sont elles-mêmes des services Web. Il y a trois types de composants qui correspondent respectivement : au propriétaire du code (**Owner**), au demandeur de code (**Client**), et au serveur **CodeShare** lui-même (**Server**). Il peut y avoir plusieurs instances de **Owner** et **Client**, et il y en a une seule de **Server**.



Un scénario d'utilisation typique est :

- Les développeurs d'un code décident de le partager publiquement. Ils mettent donc à jour localement leur fichier `index.xml` qui indique les fichiers qu'ils veulent partager.
- Les développeurs s'identifient sur le serveur pour mettre à jour leur entrée dans l'index principal maintenu par le serveur.
- Les développeurs mettent ensuite en fonction leur propre service (`Owner`) avec le nouveau index.
- les utilisateurs qui veulent trouver du code ont deux options :
  - Ils peuvent se connecter directement à un service `Owner` et exécuter une ou plusieurs des trois opérations : `search`, `list` ou `get`.
  - Ils peuvent se connecter au service `CodeShare` (`Server`) et faire des recherches via l'opération `search` qui renvoie une liste de tous les propriétaires qui possèdent du code correspondant à la demande.
  - On notera que toutes les opérations `get` se font via le propriétaire du code partagé.

Chaque utilisateur ou propriétaire doit s'enregistrer auprès du serveur avant de l'utiliser pour la première fois. Un utilisateur doit s'identifier avant toute utilisation du serveur.

## 1.2 Les interfaces

### 1.2.1 CodeShareOwner\_interfaces

- `search` prend deux arguments de type `String`, appelés `Value` et `DcElement`. Cette opération cherche dans le fichier `index.xml` du propriétaire les éléments qui correspondent à `Value` ; `DcElement` correspond (pour simplifier) à certaines caractéristiques du code partagé.
- `list` n'a pas d'arguments.
- `get` possède le même profil que `search`, sauf que le résultat est le code.

Ces trois opérations retournent toutes l'envoi d'une table d'éléments `item`. Un élément est composé de quatre champs `String` qui peuvent être égal à la chaîne vide pour certaines opérations ; les noms de ces champs sont : `path`, `fullpath`, `type`, `code`.

### 1.2.2 CodeShareClient\_interfaces

- **register** prend en argument un identificateur et un mot de passe. L'ensemble des clients du serveur **CodeShare** est modifié.
- **login** prend en argument un identificateur et un mot de passe. Le résultat retourné est une assertion SAML si l'identification est correcte, une erreur SOAP sinon.
- **searchMasterIndex** est la même opération que **search** sauf que la recherche se fait sur le serveur.
- **listMasterIndex** est la même opération que **list** sauf que la recherche se fait sur le serveur.

L'assertion SAML que renvoie l'opération **login** est un standard XML. Elle est créée et signée par l'autorité qui effectue le procédé d'authentification. Le format de cette assertion n'intervient pas dans ce projet.

### 1.2.3 CodeShareMasterIndex\_interfaces

- **register** est la même opération que précédemment, sauf que c'est l'ensemble des propriétaires du serveur **CodeShare** qui est modifié.
- **update** prend l'identificateur du propriétaire et son mot de passe, ainsi que son index mis à jour avec les nouveaux fichiers à partager. L'index principal maintenu par le serveur est modifié.

## 2 Questions

### Question 1

Donner sous la forme de diagrammes de séquences UML les scénarios très simples correspondant à :

- L'enregistrement d'un nouveau client, et la vérification qu'il est effectif.
- L'enregistrement d'un nouveau propriétaire, et la vérification qu'il est effectif.
- Une exécution de **update** avec prologue et observation que la mise à jour a été faite.

### Question 2

Donner l'architecture de l'ensemble dans le cas où il n'y a qu'un **Client** et qu'un **Owner**.

### Question 3

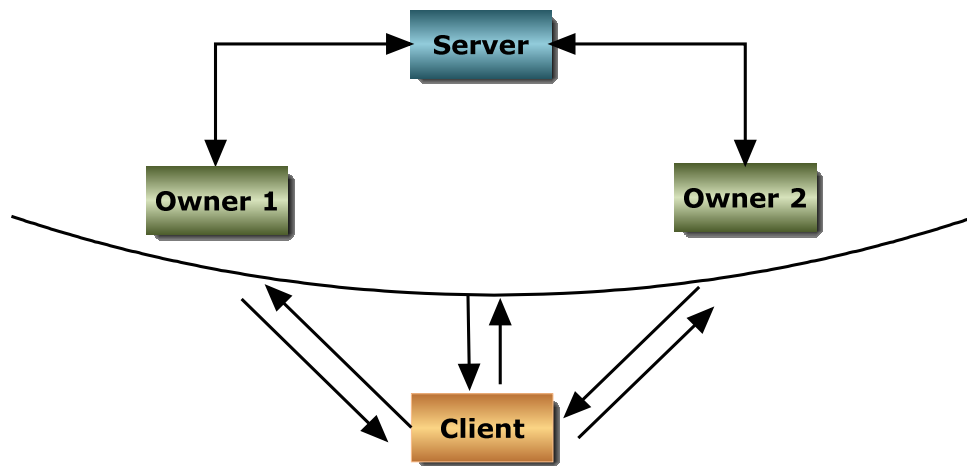
Donner une spécification en LOTOS de l'architecture ci-dessus, en tenant compte des indications suivantes :

- Les types de données utilisés doivent pouvoir se compiler en CADP. On donnera seulement la signature de ceux qui ne sont pas triviaux, avec une explication concise pour chaque opération.

- On utilisera le fait que **Owner** peut interagir en parallèle, sans synchronisation, avec **Client** et **Server**. Il en est de même pour **Server** avec **Owner** et **Client**.

Compiler la spécification en CADP.

#### Question 4



Pour se ramener à un système de transitions fini, on supprime les types de données. On considère que chaque **Owner** a deux codes au maximum. Pour rendre le modèle plus intéressant, on considère qu'il y a deux **Owners** (donc quatre ports à prévoir). Il n'y a toujours qu'un **Client** et qu'un **Server**.

Donner la spécification en LOTOS de base et les automates correspondants construits par CADP.