



Titre du document : Compte-rendu de réalisation : Prototype logiciel permettant l'évaluation de l'interopérabilité dynamique entre forges basé sur la spécification LI-2.1

Référence : LI-2-5-1

Version 1.0 du 30/10/2011

Compte-rendu de réalisation :

Prototype logiciel permettant l'évaluation de l'interopérabilité dynamique entre forges basé sur la spécification LI-2.1

Livrable dû au titre du projet	COCLICO
Lot	WP2 (<i>Interopérabilité et échange de données</i>)
Tâche	2.2 (<i>Spécification d'un format d'échange standard pour l'import/export "à froid" de données dans les forges</i>)
Livrable	LI-2-5-1

Rédacteur(s)	Vérificateur(s)	Approbateur(s)
O. Berger, C. Bouthier	P. Bournai	

Documents de références	
	Annexe technique du projet COCLICO

Gestion des versions

N° de version	Date	Auteurs	Modification apportées
0.1	21/09/2011	O. Berger, C. Bouthier	Première version
1.0	30/10/2011	O. Berger	Version finale

Copyright : © 2010-2011 Institut TELECOM, INRIA

Ce document est diffusé sous les termes de la licence « Creative Commons Attribution 2.0 France ».

Ce document est le produit de travaux effectués dans le cadre du projet COCLICO, avec l'aide financière de collectivités publiques, dans le cadre des appels à projets des pôles de compétitivité SYSTEM@TIC et MINALOGIC.

Table des matières

1 Objectifs de ce document.....	3
1.1 Contexte : tâche 2.5 du projet COCLICO.....	3
2 Interopérabilité dynamique dans les « forges actuelles ».....	4
2.1 Cas d'utilisation envisagés.....	4
3 Intégration de fonctions sémantiques au coeur d'une « forge de nouvelle génération », pour l'interopérabilité dynamique.....	9
3.1 Introduction	9
3.2 Contexte	9
3.3 Réalisations	12
3.4 Résultats	19
3.5 Conclusion	23

1 Objectifs de ce document

Ce document présente un des résultat du sous-projet WP2 « *Interopérabilité et échange de données* » .

Il fait un compte-rendu des activités de réalisation effectuées pour le livrable logiciel LI-2.5.1 du projet COCLICO « *Prototype logiciel permettant l'évaluation de l'interopérabilité dynamique entre forges basé sur la spécification LI-2.1* ».

1.1 Contexte : tâche 2.5 du projet COCLICO

La tâche 2.5 « *Spécification et implémentation d'une interopérabilité dynamique entre forges* » du sous-projet WP2 « *Interopérabilité et échange de données* » est décrite dans l'annexe technique du projet comme :

« En lien avec les sous-projets 1 (vis à vis des rôles et privilèges d'accès), et sous-projet 3 (Indexation et recherche), on spécifiera et on implémentera des fonctionnalités minimales permettant de mettre en œuvre le standard de description des méta-données défini dans la tâche 1 dans un contexte dynamique, pour la communication entre forges.

On spécifiera et on réalisera un prototype permettant d'évaluer la faisabilité de mécanismes permettant à des communautés de structurer un projet sous forme d'un méta-projet réparti sur plusieurs forges éventuellement hétérogènes, et la recherche d'informations distribuées dans des sous-projets sur différentes forges.

La réalisation de ce prototype permettra de spécifier les extensions nécessaires pour l'enrichissement ou l'adaptation du format d'échange standard défini dans la tâche 1 dans un contexte dynamique. »

2 Interopérabilité dynamique dans les « forges actuelles »

Cette première section présente les travaux entrepris sur les « forges actuelles », telles FusionForge ou Codendi, en vue de valider des scénarii d'interopérabilité dynamique, et les résultats obtenus.

2.1 Cas d'utilisation envisagés

Nous présentons tout d'abord les cas d'utilisation qui ont été envisagés :

- agrégation dans un tableau de bord personnel d'événements issus d'autres projets de l'utilisateur ;
- gestion de projets hiérarchiques et distribués et de portails de projets multi-forges.

L'objectif de ces cas d'utilisation est de définir une hypothèse de travail, afin d'ajouter de nouvelles fonctions, ou de valider certaines fonctionnalités déjà présentes dans les forges. Nous ne détaillerons pas l'ensemble des réalisations effectuées pour toutes les fonctions décrites.

2.1.1 Agrégation dans un tableau de bord personnel d'événements issus d'autres projets de l'utilisateur

L'objectif de ce cas d'utilisation est de permettre la visualisation dans une interface unique, sur une des forges où un utilisateur a un compte, des informations (éventuellement privées) liées aux projets de ce même utilisateur sur d'autres forges qu'il utilise également. Idéalement, l'utilisateur décide lui-même des informations qu'il souhaite agréger dans un tableau de bord Web unique, évitant ainsi de se connecter à différentes forges, uniquement pour savoir ce qu'il y a de nouveau dans ses autres projets. La visualisation peut ensuite être identique, quelle que soit la forge sur laquelle l'utilisateur est effectivement connecté, puisque chacune des forges peut aller chercher les informations pertinentes sur les autres.

Nous nous plaçons dans un cas courant pour les contributeurs Open Source, notamment, où une même personne, utilisatrice d'une forge, peut disposer de plusieurs comptes sur d'autres forges (sans lien quelconque entre ces comptes, ni au niveau des opérateurs de ces différentes forges, ou sans unicité des identifiants de login sur ces forges, contrairement à ce qui peut se produire dans une organisation avec annuaire d'entreprise, une fédération d'identité, une coordination entre les administrateurs des outils,...).

On souhaite que l'utilisateur puisse gérer son profil de façon globale pour ajouter d'autres comptes dont il dispose, avec un impact "immédiat" sur toutes les forges où il pourrait se connecter (à la prochaine connexion).

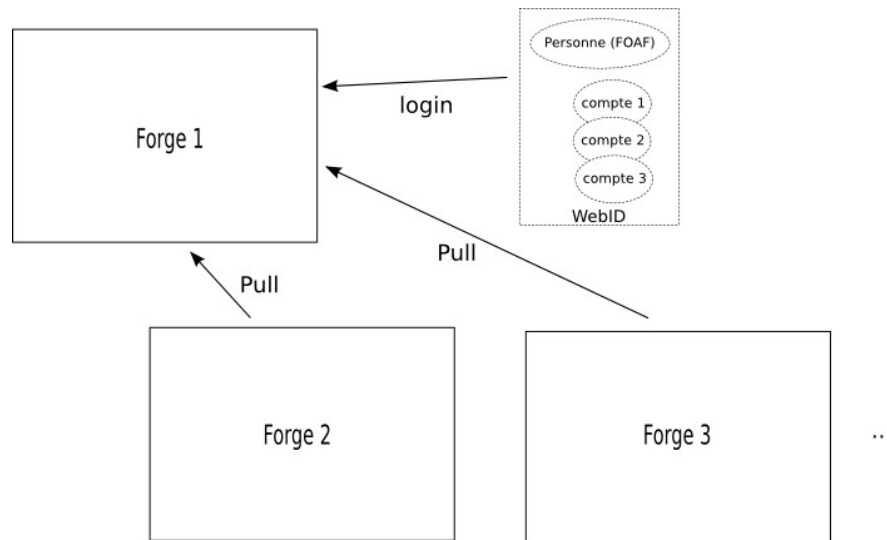


Figure 1: Récupération des informations de l'utilisateur entre forges

2.1.1.1 Mécanismes éventuellement utiles pour l'implémentation

On détaille ici certains mécanismes permettant d'envisager l'implémentation de ce cas d'utilisation :

- un utilisateur (une personne : foaf:Person) a un profil WebID (FOAF + SSL) dont le document FOAF décrit (entre autres choses) les différents *comptes* (sioc:UserAccount) qu'il utilise sur différents services (de forges). C'est ce document FOAF et son certificat client SSL associé qu'il utilise pour se logger sur les différentes forges supportant WebID (cf. livrables WP1 pour plus de détails sur l'authentification via WebID) ;
- à défaut de support de WebID, la description des comptes des forges distantes est gérée dans des préférences ad-hoc de l'utilisateur sur chaque forge. L'inconvénient par rapport à la solution FOAF ci-dessus : une mise-à-jour est à faire sur toutes les forges en cas d'utilisation d'un nouveau compte utilisateur ;
- sur chaque forge, un service de découverte OSLC et de requêtage permet de trouver les éléments à afficher dans les tableaux de bord de suivi (bugs affectés, etc.)
- chaque forge peut découvrir les items du compte distant d'une autre forge à afficher à l'utilisateur. Le compte en question est retrouvé dans le profil FOAF transmis par l'utilisateur à la connexion : l'ajout d'une nouvelle forge et d'un nouveau compte à un profil existant, est complété par un lien avec le « service provider catalog » OSLC si la forge distante est compatible OSLC. Cet ajout est donc répercuté immédiatement sur toutes les sessions ouvertes par l'utilisateur ultérieurement.
- quelle que soit la forge sur laquelle l'utilisateur se connecte, il peut voir apparaître dans "ma page" les items relatifs à ses comptes sur d'autres forges, pourvu qu'une délégation de jeton OAuth ait été initiée entre les différentes forges (OAuth fait partie des préconisations OSLC. Cf. WP1 pour plus de détails sur OAuth).
- Les informations sur la liste des projets distants de l'utilisateur sont soit décrites

dans le profil FOAF (en lien avec l'ontologie planetforge, et SIOC), soit retrouvées (plus sûrement s'il s'agit d'informations privées, qu'on ne souhaite pas transporter dans un profil FOAF public) via un service Web ad-hoc (via le service catalog OSLC, par exemple)

- Les éléments de notification issus des projets (privés) distants sont collectés (en direct ou en asynchrone) via OAuth, pour la gestion des permissions d'accès au nom de l'utilisateur. Les sessions OAuth sont créées lors de la connexion d'un utilisateur, par l'accès aux deux forges émettrice et réceptrice des informations. Par la suite, le jeton OAuth permet aux forges de se connecter aux autres au nom de l'utilisateur pour aller récupérer ses informations, même en dehors de ses connexions (si nécessaire, car processus asynchrone)

Les tableaux de bord intègrent par la suite le support des « avatars » et des prévisualisations « OSLC compact preview », comme dans le cas d'utilisation sur les données publiques des forges.

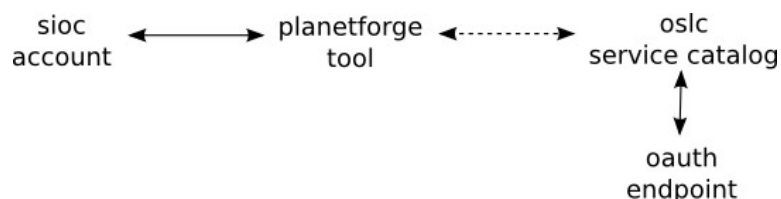


Figure 2: Découverte des endpoints OSLC

La découverte des endpoints OSLC depuis le profile FOAF : nécessite un lien entre les sioc :Space des outils de la forge (Cf. ontologie planetforge) et les endpoints OSLC, via le service catalog.

On envisage également, pour des raisons de performances de la mise en oeuvre, la possibilité de disposer d'un mécanisme de notification du besoin d'aller chercher des informations sur une forge distante, sans pour autant transporter des infos confidentielles, en mode « push », plutôt que « pull » (par exemple via le protocole pubsubhubbub).

2.1.1.2 Contenu des tableaux de bords agrégés

Les infos agrégées dans les tableaux de bords concernent toutes les notifications d'événements dans les projets actuellement présents dans les forges :

- commits
- mise en ligne de documents
- assignations de bugs
- changement d'état de bugs dont le user est l'émetteur
- éléments monitorés
- publication de news de projets,
- etc.

2.1.1.3 Réalisations

La cible générale pour l'intégration des réalisations de ce cas d'utilisation a été prioritairement FusionForge, pour la « forge maître » réalisant l'agrégation des informations distantes. Les informations agrégées provenant d'instances de FusionForge ou de Codendi.

2.1.1.3.1 Tableau de bord d'agrégation pour FusionForge dans "my page"

Un nouveau tableau de bord est implémenté via un nouveau plugin FusionForge (committé sur le trunk) nommé « globaldashboard ». Il intègre la gestion des comptes distants de l'utilisateur (y compris les end-point OSLC, ou les flux RSS, ainsi qu'une découverte dynamique de certaines informations si la forge dispose d'un mécanisme de publication de méta-données RDF).

2.1.1.3.2 Support d'authentification WebID FOAF + SSL

Une ébauche de plugin d'authentification a été commencée, qui est committée sur FusionForge, et nécessitera des travaux additionnels ultérieurs. En attendant, le plugin global dashboard gère les informations de comptes en mode « manuel » avec des ajouts explicites faits par l'utilisateur.

2.1.1.3.3 Client OAuth dans FusionForge

- Un client OAuth est committé dans le plugin FusionForge « oauthconsumer » qui permet de se connecter à des forges distantes pour la récupération des informations à afficher à l'utilisateur, ou pour d'autres usages, comme la publication sur Twitter, implémentée à titre d'exemple.

2.1.1.3.4 "Service" de notification / publication des infos à agréger

Un service a été développé dans FusionForge sur la base des informations présentes dans la page de suivi de l'activité des projets pour publier (y compris depuis un projet privé) sous forme de flux RSS/ATOM les informations qui seront récupérées par les forges distantes (via OAuth).

2.1.2 Gestion de projets hiérarchiques et distribués et de portails de projets multi-forges

L'objectif d'un tel cas d'utilisation est de répondre aux besoins :

- de structurer des projets en une hiérarchie de projets/sous-projets, éventuellement répartie sur plusieurs forges physiques
- de permettre la navigation dans une liste de projets (publics ?) de différentes forges selon différents index :
 - par entités participantes aux projets
 - par tags
 - par catégories ("trove")

Les réalisations sur ce cas d'utilisation n'ont pas été entreprises, car ne représentant pas

un intérêt fort pour les participants en l'état actuel des besoins envisagés pour leurs propres utilisations des forges.

2.1.3 Gestion de projets hiérarchiques inter-forges

L'objectif de ce cas d'utilisation est de supporter la notion de projets – sous-projets, qui permet de structurer des communautés en sous-unités de façon à rendre plus lisible une structuration, et à gérer les privilèges d'accès de façon plus fine.

L'accès aux projets doit refléter cette structuration. Les privilèges peuvent éventuellement être hérités.

Certains éléments peuvent être agrégés dans les tableaux de bords des projets de plus haut niveau.

Dans le cas envisagé, un sous-projet n'a qu'un projet parent au maximum (structure d'arbre, pas de graphe).

La localisation physique d'un projet sur une forge précise est indépendante de sa position dans la hiérarchie des projets : un projet A1 d'une forge A peut avoir comme sous-projets deux projets A2 et B1 respectivement sur les forges A et B.

Les widgets de navigation dans les listes de projets doivent pouvoir être agrégés à travers différentes forges, par exemple pour agréger tags et faire un rendu de nuage de tags global, ou pour les catégories de structuration des projets ("trove"). Ces différents éléments étant modélisés avec des ontologies communes en RDF.

Un « super projet » peut être implémenté comme un projet "presque vide" (sans plugins des différents espaces collaboratifs) à l'exception de plugins dédiés à l'affichage des sous-projets.

Fusionforge disposait déjà d'un plugin "project-hierarchy" permettant de construire une arborescence de projets locaux.

Mais l'implémentation actuelle de ce plugin rend difficile son extension à des besoins concernant des projets distants.

Un prototype a été réalisé pour ce cas, sous la forme d'un plugin « extsubproj », qui implémente la visualisation de sous-projets distants, en intégrant la découverte dynamique de leurs propriétés, récupérées dans les profils DOAP (grâce au plugin « doaprdf » développé par COCLICO) des projets distants.

3 Intégration de fonctions sémantiques au coeur d'une « forge de nouvelle génération », pour l'interopérabilité dynamique

3.1 Introduction

Contrairement à la section précédente, qui introduisait des fonctions d'interopérabilité dynamique entre forges de la génération « actuelle », cette section présente des travaux réalisés dans une optique plus innovante, sur la base d'une « forge de dernière génération », la forge QualiPSo.

3.1.1 Objectif

Cette section fait un bilan de la participation de l'équipe INRIA SCORE au projet COCLICO dans le cadre de la tâche 2.5. Elle décrit les activités menées par l'équipe au sein de ce projet, les résultats obtenus, et ce qu'il reste à faire.

3.1.2 Plan

La première partie de cette section explique le contexte de la participation de l'équipe SCORE à ce projet. Elle décrit les raisons de la participation de l'équipe SCORE au projet COCLICO et le but poursuivi. Elle résume aussi l'architecture de la forge Qualipso, utilisée comme support de réalisation.

La deuxième partie détaille les différentes réalisations dans ce domaine. Elle décrit notamment la liste des fonctionnalités ajoutées, tout en les remettant dans le cadre de la proposition globale.

La troisième partie fait la liste des résultats obtenus par l'équipe SCORE en tant que participation au projet COCLICO, ainsi que des livrables correspondants.

Finalement, la conclusion fait le bilan, par rapport à la proposition globale, de ce qui a été réalisé et de ce qui reste à faire.

3.2 Contexte

3.2.1 Équipe SCORE

L'équipe SCORE mène une activité de recherche sur les forges logicielles depuis de nombreuses années, notamment par ces réalisations comme LibreSource ou, plus récemment, Qualipso. Cette dernière forge, en particulier, a été réalisée lors de la participation de l'équipe, en tant qu'équipe INRIA, au projet européen du même nom. La réalisation de cette forge nouvelle génération a permis de poser de nouvelles questions, et de soulever de nouvelles pistes de recherche. L'une d'elles, en rapport avec le thème de recherche de l'équipe SCORE et de ses travaux, est la piste d'une forge « sémantique ».

Le but de la participation au projet COCLICO est, pour l'équipe SCORE, de mener des activités exploratoires de recherche concernant l'ajout de capacités « sémantiques » à une forge. En particulier, un des avantages prévisibles de l'approche sémantique est la possibilité d'interopérabilité entre forges, par l'utilisation d'un format sémantique comme format commun de données, et par la possibilité d'interactions entre des modèles de données différents. L'exploration de cette piste est une nouveauté, non réalisée dans le

projet Qualipso.

3.2.2 Forge et sémantique

La proposition de recherche de l'équipe SCORE, concernant l'ajout de fonctionnalités « sémantiques » à la forge Qualipso, est de faire en sorte qu'une forge logicielle puisse être une source de donnée pour « le Web des données »¹.

Le Web des données, par opposition au Web des documents, désigne toute la partie du Web qui s'intéresse non pas à un document en particulier (à une page par exemple) mais aux données véhiculées par cette page. Un principe important, notamment, est celui de l'interopérabilité sémantique : « la possibilité d'interpréter automatiquement les informations échangées, de manière juste et en respectant leur sens, dans le but de produire des résultats utiles » (source : [http:// en.wikipedia.org/wiki/Interoperability](http://en.wikipedia.org/wiki/Interoperability)).

Pour faire d'une forge une source de données, plusieurs approches sont possibles : l'approche « ontologies pour les forges », et l'approche « forges pour les ontologies ».

La première approche consiste à créer une ontologie permettant de décrire l'ensemble des ressources manipulées par une forge, d'ajouter les informations sémantiques nécessaires aux ressources, puis de les rendre accessible, soit par moissonnage (accès passif), soit par requêtage (accès actif). Cette approche permet à un autre système (forge, moteur sémantique) de récupérer les informations de la forge et de les utiliser. Mais l'interopérabilité ne se fait alors que dans un sens, la forge ne peut pas utiliser les informations sémantiques d'autres systèmes.

La seconde approche ne se limite pas simplement à rendre accessibles des informations sémantiques, mais aussi à utiliser ces informations. Le but est de permettre aux ontologies d'émerger de l'usage social de la forge. Non seulement la forge utilise en interne ces informations sémantiques, mais elle permet aussi à l'utilisateur de définir et d'utiliser de nouvelles informations sémantiques. Il est alors possible, par exemple, de relier sémantiquement des ressources de la forge à des ressources sémantiques disponibles sur d'autres systèmes.

C'est cette deuxième approche, l'approche « forges pour les ontologies », qui a été choisie. Elle a été présentée en mai 2010 aux autres membres du projet COCLICO, qui ont validé l'exploration de cette approche comme participation de l'équipe SCORE au projet COCLICO.

3.2.3 Prérequis

Pour des raisons à la fois pratiques (compétences) et techniques (possibilité de l'architecture), il a été décidé et validé que les développements se feraient sur la forge Qualipso. En effet, pour réaliser une telle approche, plusieurs critères sont nécessaires :

1. l'existence d'un « noyau » de fonctionnalités communes rendues disponibles à tous les services de la forge. C'est dans ce noyau que doivent être implémentées les fonctionnalités sémantiques, de manière à les rendre disponibles pour tous les services.
2. l'existence d'une abstraction des ressources. Chaque ressource pouvant être reliée à une information sémantique, toute ressource doit pouvoir être manipulée de la même manière, par le moteur sémantique, quel que soit son type réel.
3. l'existence d'un nommage unique de chaque ressource, et d'un annuaire des

¹ http://en.wikipedia.org/wiki/Linked_Data

ressources. En effet, de manière à être utilisée dans un lien sémantique, une ressource doit pouvoir être identifiée et retrouvée à partir d'une URL unique pour chaque ressource.

De plus, il est souhaitable d'avoir un mécanisme d'affichage des ressources qui soit accessibles de manière générique quelle que soit la ressource, mais qui affiche de manière spécifique chaque ressource en fonction de son type.

Tous ces critères sont réalisés par la forge Qualipso.

3.2.4 La forge Qualipso (Qualipso Factory)

La forge Qualipso a été réalisée dans le cadre du projet Européen Qualipso. Le but de l'équipe SCORE dans la participation à ce projet était de répondre au problème de recherche suivant : comment réaliser une forge totalement modulaire afin de faciliter l'émergence d'une communauté de développeur ?

Le constat de base est le suivant : la plupart des forges existantes sur le marché ne bénéficient pas d'une large communauté de développeur. L'hypothèse faite par l'équipe SCORE est que ceci est dû à la difficulté d'ajouter un service à une forge existante. Il faut le plus souvent télécharger l'intégralité du code de la forge, connaître l'ensemble des principes, et modifier la glue existante. De plus, l'intégration des services entre eux est quasiment inexistante, les services ne se parlent pas entre eux, les interfaces graphiques pouvant même être totalement différentes (par choix, par exemple, de ne pas les réimplémenter).

Pour résoudre ce problème, l'approche suivie par l'équipe SCORE dans la réalisation de la forge Qualipso a été de faciliter le plus possible le travail d'un développeur voulant ajouter un nouveau service, tout en garantissant à l'utilisateur final de la forge la possibilité d'une intégration maximale entre les services. Pour cela, les principes suivants furent posés :

- le développeur ne doit PAS avoir à télécharger l'intégralité de la forge pour ajouter un nouveau service
- le développeur ne doit PAS avoir à modifier du code source existant de la forge pour ajouter un nouveau service
- le développeur ne doit PAS avoir à redévelopper les services essentiels de la forge (authentification, sécurité, indexation du contenu, gestion des événements...)
- Tout ceci doit être vrai autant pour la partie « backend » (intelligence du service) que pour la partie « frontend » (interface graphique)
- pour l'utilisateur final, il ne doit pas y avoir de différence autre que spécifique à l'intelligence du service entre les ressources des services. Tous les services doivent apparaître comme étant au même plan et faisant partie du même logiciel.

Tous ces critères sont désormais réalisés dans la forge Qualipso. Celle-ci se base sur une architecture à composant, autant au niveau backend que frontend.

Au niveau backend, chaque service est un module EJB (Enterprise Java Beans) différent. Le packaging et la distribution du module sont assurés par Maven, ce qui permet un assemblage à froid des modules sans avoir à télécharger l'ensemble des sources de la forge. La forge fournit en plus un ensemble de services commun, les services « Core », comme l'indexation, la gestion des droits ou la gestion des événements), eux aussi sous forme d'EJB. L'utilisation de ces services se fait à travers les interfaces EJB.

Les services « Core » permettent l'interaction et l'agrégation des services à travers un

arbre de nommage, le « binding ». Cet arbre contient un lien vers chacune des ressources créées par les différents services. Il joue ainsi un double rôle : un rôle d'annuaire, car il permet d'accéder à toutes les ressources, de manière générique, quelle que soit leur type, et un rôle de nommage, puisque chaque ressource est uniquement identifiée par son chemin d'accès dans l'arbre. Ainsi, l'ajout d'un nouveau service nécessite uniquement l'accès à l'arbre de nommage, et donc aucune modification du code source existant.

Au niveau « frontend », chaque service est lié à un module d'interface graphique, qui contient les vues de ce service. De même que pour le frontend, le packaging et la distribution du module sont assurés par Maven, ce qui permet un assemblage à froid des modules sans avoir à télécharger l'ensemble des sources de la forge. Au niveau technique, les vues sont réalisées avec la technologie GWT, ce qui permet d'avoir des vues et des échanges de données frontend-backend dynamiques. Les vues utilisent une librairie spécifique, OpenParts, pour se lier les unes aux autres dynamiquement. La librairie permet notamment à une vue d'inclure la vue d'une autre ressource sans connaître son type, afin d'éviter toute interdépendance. Il est ainsi possible d'ajouter une nouvelle vue, y compris à l'intérieur de vues existantes, sans avoir à modifier le moindre code source. De manière identique, il est possible de modifier l'interface graphique par l'ajout de « skins », sans avoir à modifier le code source des vues existantes.

3.3 Réalisations

Nous présentons les réalisations entreprises dans le cadre de COCLICO, sur la base de la forge Qualipso.

3.3.1 Affichage sémantique d'une ressource au format RDF

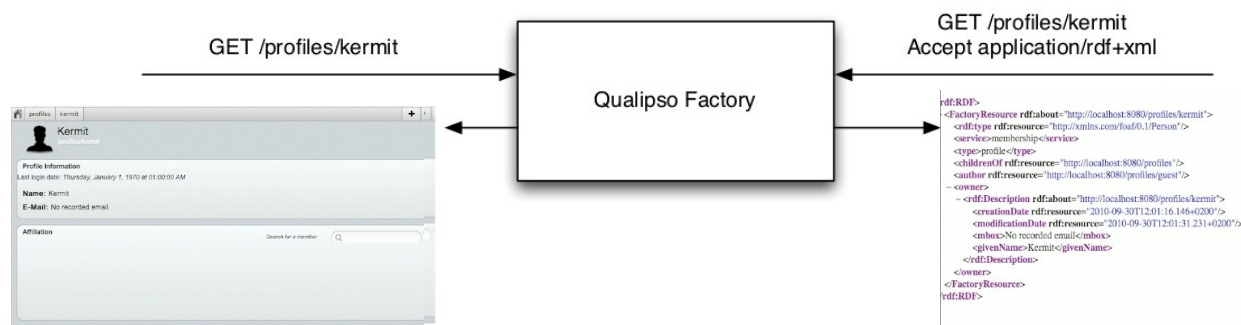
L'approche « forge pour les ontologies » contient l'approche « ontologie pour les forges ». Cela signifie, à minima, que la forge doit pouvoir être moissonnée et fournir une description sémantique de chacune de ses requêtes. C'était donc la première réalisation à effectuer.

Le but de cette réalisation est de permettre l'affichage des données sémantiques d'une ressource. L'objectif est de rendre la ressource moissonnable par des moteurs sémantiques. Pour cela, il faut :

1. que la description sémantique de la ressource se fasse suivant un format standard
2. que l'accès aux informations sémantiques se fasse suivant un protocole standard.

Le format choisi est le format RDF. En ce qui concerne le protocole d'accès, deux choix sont possibles : soit l'ajout d'informations directement dans la page web (choix fait par FusionForge par exemple via RDFa), soit l'accès à une page spécifique en fonction d'un paramètre dans le header de la requête HTTP (par content-negotiation : header HTTP « Accept », qui doit être mis à « application/rdf+xml »).

C'est ce second choix qui a été implémenté dans la forge Qualipso.



Dans la forge Qualipso, chaque ressource possède une URL propre, correspondant à son chemin d'accès dans l'arbre de nommage. Il suffit de préfixer ce chemin d'accès par l'URL du serveur pour obtenir une URL publique permettant d'accéder à une ressource spécifique.

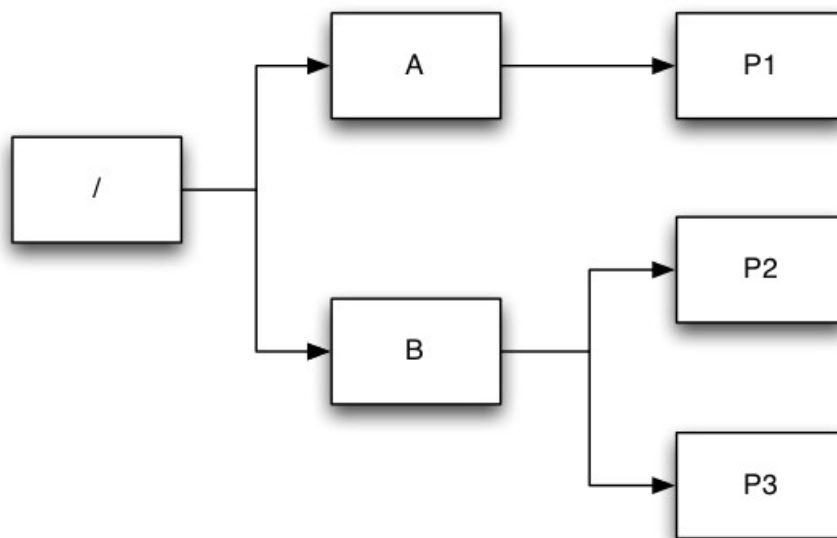
Lorsque la forge Qualipso reçoit une requête vers une de ces URL publiques, elle cherche la présence d'un header « Accept: application/rdf+xml » dans la requête HTTP. Sans ce header, la forge récupère le type du service en charge de la ressource, et lui demande de retourner l'interface graphique standard d'accès à cette ressource en HTML (à gauche). Avec le header, la forge récupère la ressource en question et lui demande de retourner sa propre description sémantique, au format RDF XML (compatible avec le Web des données).

3.3.2 Kernel « sémantique » dans la forge Qualipso

Le but de l'approche « forge pour les ontologies » est de permettre, autant au développeur d'un service qu'à l'utilisateur final de la forge, de définir et d'utiliser des informations sémantiques, notamment entre les ressources proposées par la forge. Ainsi, la forge est capable d'utiliser le modèle sémantique comme modèle de donnée.

En particulier, cela signifie notamment que les relations entre les ressources gérées par la forge ne dépendent pas d'un modèle de donnée propre, mais repose sur la définition de liens sémantiques entre les ressources.

Dans la forge Qualipso, les ressources sont gérées par des services différents et indépendants. Afin de retrouver ces différentes ressources, quels que soient les services qui les gèrent, celles-ci sont référencées dans un arbre de nommage, qui leur assure ainsi une URL propre à chaque ressource.



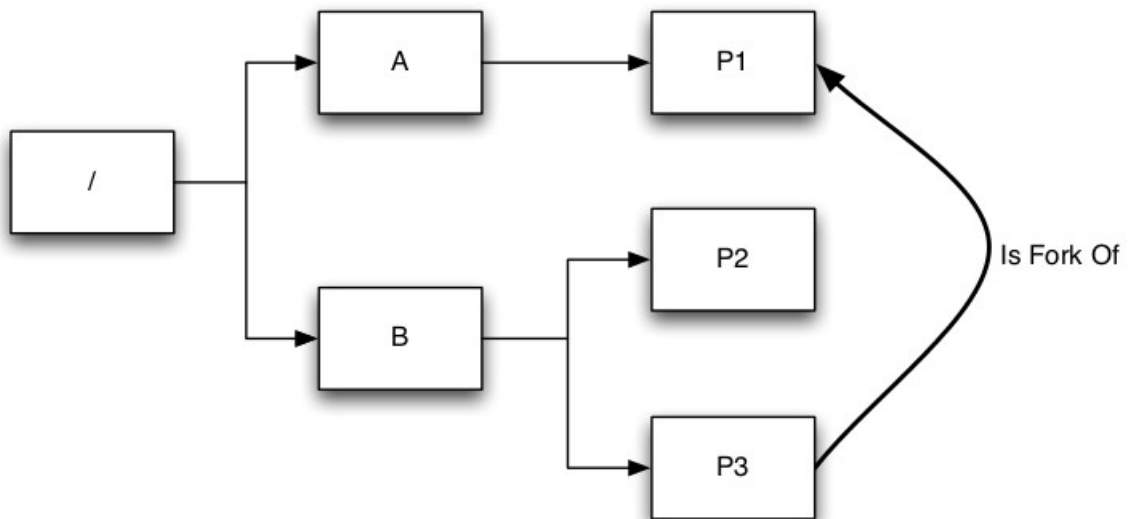
Pour assurer l'intégration des services (par exemple un service qui utilise un autre), des liens existent entre les ressources.

Par exemple, un projet peut être lié à un gestionnaire de version (relation « appartient à »),

ou un gestionnaire de version peut être lié à un autre gestionnaire de version (relation « est un fork de »). Si les deux ressources sont du même type, donc gérées par le même service, le lien fait partie du modèle de donnée du service. Mais si les deux ressources sont de types différents, alors il n'est plus possible d'utiliser un modèle de donnée propre à un service, car cela impliquerait une dépendance entre les deux services (« git » devrait connaître « projet » par exemple, ou l'inverse). Dans ce cas, dans l'approche normale, le lien est fait par l'arborescence de l'arbre de nommage : les ressources « P2 » et « P3 » sont liées, car elles sont toutes les deux filles de « B », par exemple.

3.3.3 Liens sémantiques

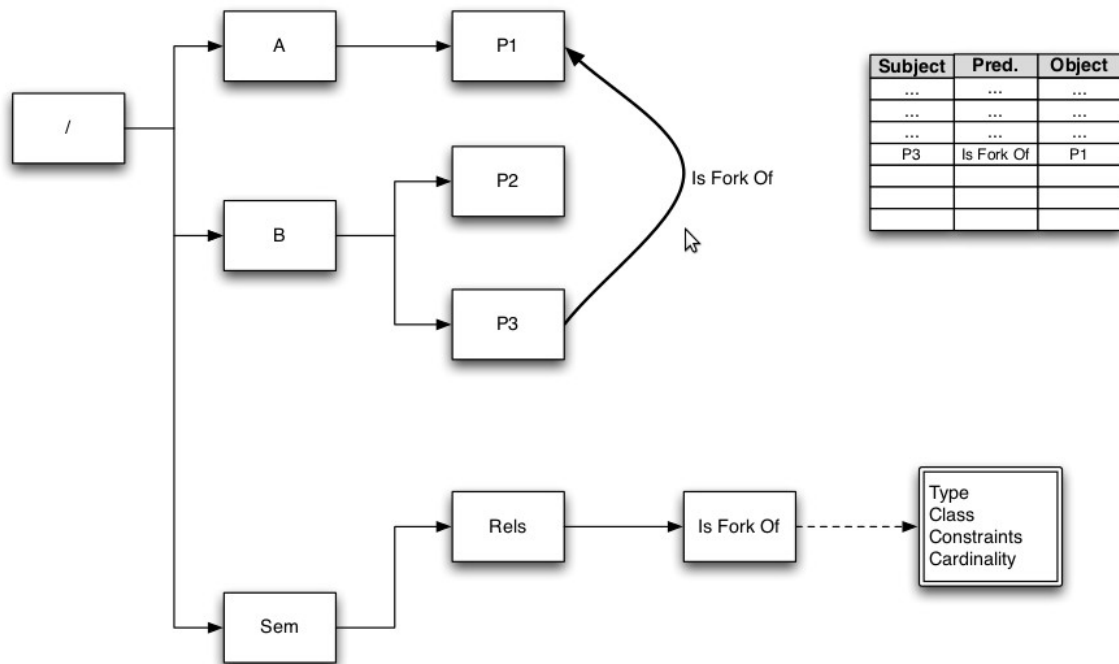
Dans l'approche sémantique, le lien est modélisé directement par un lien sémantique, une relation de la forme classique « sujet - prédicat - objet » (nativement supportée par RDF).



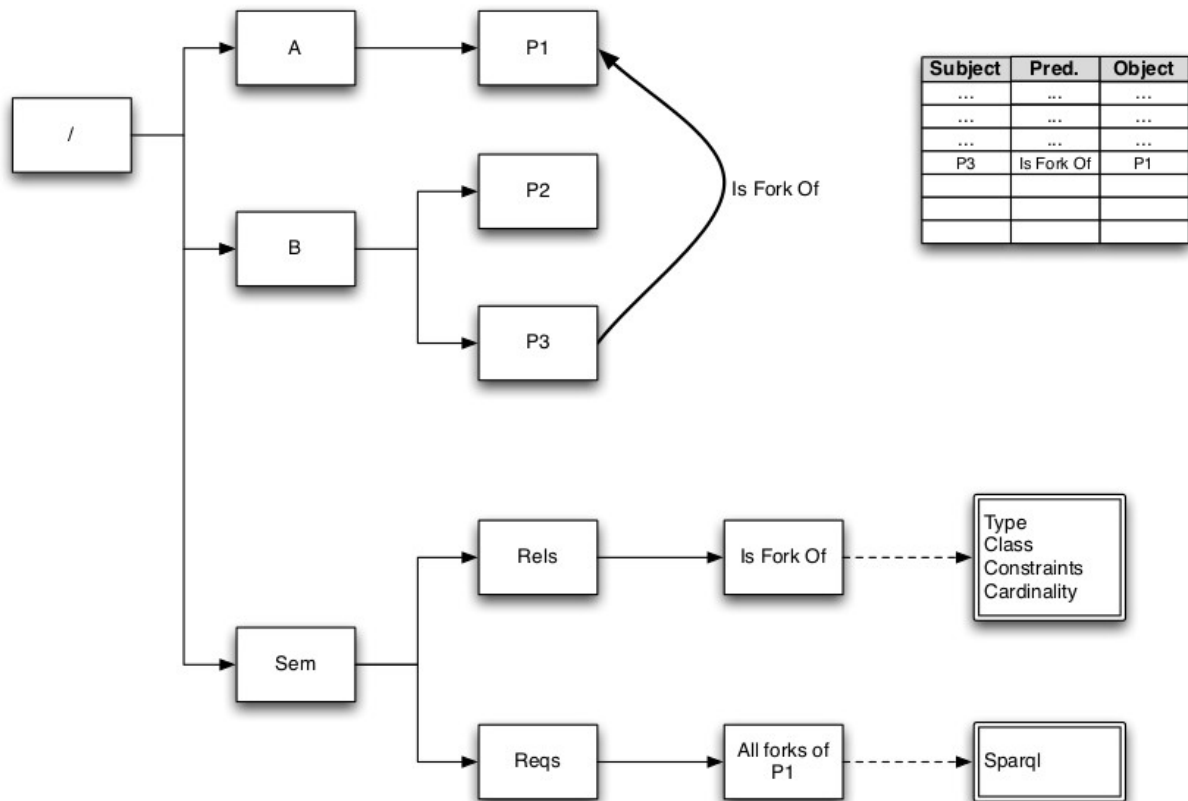
Cela nécessite qu'il existe un « store sémantique » qui contient ces relations, et dans lequel il est possible d'ajouter ces relations.

Subject	Pred.	Object
...
...
...
P3	Is Fork Of	P1

De plus, chacun des 3 membres de la relation doit être accessible par une URL unique. Cela est vrai pour n'importe quelle ressource. Pour que cela soit aussi vrai pour les prédicats, ceux-ci sont gérés comme des ressources, donc référencés dans l'arbre de nommage. Cela a l'avantage de les manipuler de la même manière que les autres ressources (notamment au niveau de la gestion des droits).



Il est possible d'exécuter une requête sémantique, suivant le langage de requête SPARQL, sur le store sémantique. Un développeur de service peut se servir de ces requêtes pour récupérer des données propres à son service, ou pour réaliser sa logique métier (par exemple, connaître les autres forks pour un gestionnaire de version). Dans ce cas, le développeur code, dans son service, la requête, et l'exécute lorsqu'il en a besoin. Mais une requête peut aussi être créée par l'utilisateur final, par exemple pour afficher des données personnalisées sur sa page, ou pour paramétrer et configurer l'affichage d'une ressource en fonction des ressources avec lesquelles elle est liée. Dans ce cas, la requête doit être exécutée à chaque fois que sa page est chargée. Dans ce cas, afin de pouvoir être réutilisée, la requête peut être sauvée comme une ressource. Elle a alors une référence dans l'arbre de nommage.



3.3.4 Service sémantique

L'implémentation de cette architecture sémantique se fait, dans la forge Qualipso, au travers du service « semantic ». Ce service est un service « core », c'est-à-dire faisant partie du kernel de la forge et commun à tous les services. Il a été implémenté spécifiquement pour le projet COCLICO.

Ce service permet :

- la création et le retrait de relations dans le store sémantique
- la gestion CRUD (Create, Read, Update, Delete) de ressources « prédicat »
- la gestion CRUD de ressources « requête »
- l'exécution de requêtes sur le store sémantique.

3.3.4.1 Relations sémantiques

Les relations sémantiques sont créées dans le store en passant au service sémantique un triplet de références vers l'arbre de nommage : le sujet, le prédicat, et l'objet. L'objet peut être soit une référence, soit une String, contenant dans ce cas la valeur d'un « littéral », c'est-à-dire une valeur d'un type simple (par exemple, le nom d'une personne, ou le numéro de téléphone). Le sujet et le prédicat, eux, sont toujours des ressources, le prédicat étant une ressource d'un type spécifique.

Ces méthodes sont d'accès publics, c'est-à-dire accessibles autant au développeur d'un service qu'à l'utilisateur final. Ainsi, un utilisateur peut définir ses propres relations sémantiques entre les ressources, à un détail près (cf. le paragraphe sur la sécurité).

3.3.4.2 Ressources prédicats

Les ressources prédicats sont des ressources comme les autres (donc avec des règles de sécurité, générant des événements, etc.). Elles peuvent être créées de deux manières : soit directement, par appel d'une méthode auprès du service sémantique, soit par import d'une ontologie. Cette dernière possibilité est réservée aux développeurs de services, qui peuvent ainsi ajouter, dans leur service, un ou plusieurs fichiers décrivant les ontologies, et faire charger ces ontologies directement par le service sémantique au bootstrap de la forge.

Les ressources prédicats ont un type particulier qui est testé lors de l'ajout d'une relation sémantique.

3.3.4.3 Ressources requêtes

Les ressources requêtes sont elles aussi des ressources comme les autres, avec les mêmes règles. Elles sont créées par appel direct au service sémantique. Elles possèdent en plus des propriétés spécifiques pour leur affichage dans l'interface graphique (c.f. la partie affichage). Cela permet ainsi à un service ou à un utilisateur de réutiliser la requête, ou de la partager avec d'autres.

3.3.4.4 Exécution de requête

Le service sémantique permet l'exécution de requêtes sémantiques utilisant le langage de requête SPARQL.

Exemple de requête SPARQL :

```
PREFIX qualipso:<http://www.qualipso.org/2010/07/factory#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?person $name ?email ?owner ?service
WHERE {
  ?person a foaf:Person;
  foaf:givenName ?name;
  foaf:mbox ?email
  OPTIONAL {?name a qualipso:FactoryResource;
  qualipso:owner ?owner}
  OPTIONAL {?person a qualipso:FactoryResource;
  qualipso:service ?service}
}
```

Le service sémantique permet l'exécution de deux types de requêtes SPARQL : les requêtes directes, passées sous la forme d'une chaîne de caractère contenant le code SPARQL, et les requêtes ressources. Dans ce cas, c'est le chemin d'accès à la ressource qui est passé au service. Dans les deux cas, le résultat est retourné sous la forme standard en XML.

3.3.4.5 Sécurité

Au cours de l'implémentation de ces fonctionnalités, un problème de sécurité est apparu : si un service utilise des relations sémantiques pour modéliser son intelligence métier, alors le fait de pouvoir créer certaines relations sémantiques doit être sécurisé, et ne doit plus être accessible à n'importe quel utilisateur.

Soit par exemple la relation « est un sous-projet », pouvant relier deux projets. Soit un projet A et un projet B, sous-projet du projet A. Qui a le droit de définir la relation « est un sous-projet » entre A et B ? Certainement pas n'importe qui, car si B n'est pas sous projet de A, il n'a pas envie d'être vu comme étant dépendant de A. De même, A n'a pas forcément envie d'avoir B déclaré comme « sous projet » si B n'est pas un sous-projet de A. Il peut y avoir, dans les deux cas, des problèmes de « vol de notoriété ». Pourtant, n'importe qui, ayant un projet, peut créer un sous-projet et donc mettre la relation entre SES deux sous-projets.

Une première approche a été de fixer des conditions sur les droits des ressources. Il est clair que tout le monde a le droit d'accès au prédicat « est un sous-projet », donc il fallait fixer les droits sur le sujet ou sur l'objet. Par exemple, un utilisateur n'a le droit de fixer une relation entre un sujet et un objet que s'il a les droits sur le sujet. Ou alors sur l'objet. Mais quelle que soit la combinaison de conditions, il a toujours été possible de trouver un contre-exemple.

C'est pourquoi une autre approche a été choisie : celle de la séparation des espaces de nommages, entre les relations établies par les services et celles établies par les utilisateurs finaux. Ainsi, les prédicats créés par les services ne peuvent être utilisés que par les services. Les utilisateurs ne peuvent, eux, utiliser que les prédicats créés par leurs soins.

L'avantage est qu'un service peut alors faire confiance aux relations utilisant ses prédicats : ces relations ont forcément été créées par le service, donc il a pu contrôler qu'elles ont été créées conformément aux règles de sécurité. Par exemple, c'est le service « projet » qui, lors de la création d'un sous-projet, met automatiquement la relation « sous-projet ».

Le désavantage est que l'utilisateur ne peut pas créer de relations en utilisant les prédicats créés par le service, c'est-à-dire notamment les prédicats créés par importation d'ontologies. Plus exactement, il ne peut pas DIRECTEMENT créer de telles relations, mais il peut toujours utiliser les services du service ayant importé l'ontologie. Le service « membership », par exemple, importe l'ontologie FOAF². Cela signifie que si l'utilisateur veut créer une relation avec un prédicat FOAF, il doit obligatoirement passer par le service membership, ou bien créer son propre prédicat qui étend celui de FOAF.

3.3.5 Affichage des résultats d'une requête sémantique

Le fait de pouvoir créer des liens sémantiques, que ce soit automatiquement par les services ou bien directement par l'utilisateur final n'a d'intérêt pour l'utilisateur final que par la possibilité d'afficher le résultat d'une requête sémantique.

La possibilité de créer des liens permet de structurer les informations de manière sémantique. La possibilité de créer et sauvegarder une requête sous forme de ressource permet de pouvoir réutiliser une requête. Mais la principale fonctionnalité attendue par l'utilisateur est la possibilité de visualiser les résultats d'une requête sémantique, qu'elle ait juste été créée ou qu'elle soit réutilisée.

Cet affichage doit respecter un certain nombre de principes. En particulier, il doit s'intégrer parfaitement avec l'interface existante. Cela signifie notamment que cet affichage doit :

- pouvoir être inclus et s'intégrer normalement dans l'affichage d'une autre ressource
- pouvoir prendre en compte le type des résultats afin d'afficher une vue adéquate

² <http://www.foaf-project.org/>

En ce qui concerne la partie « inclusion », celle-ci est réalisée de la même manière que pour les autres vues de la forge Qualipso, en utilisant le framework OpenParts. Une requête sémantique étant une ressource comme une autre, sa vue est, comme pour les autres ressources, un composant qui est intégrable avec d'autres composants.

En ce qui concerne l'affichage spécifique en fonction du type de résultat, là aussi, ceci est rendu réalisable par l'utilisation du framework OpenParts : l'affichage de chaque résultat se fait en utilisant directement la vue composant du résultat en question. Cela permet notamment de déléguer au service en question l'affichage de la ressource, et même d'avoir une vue « preview » pour chacun des résultats.

3.4 Résultats

3.4.1 Une forge sémantique

Le principal résultat obtenu lors de ce projet est la réalisation d'une forge utilisant l'approche « forges pour les ontologies », capable d'utiliser en son sein des liens sémantiques et permettant à un utilisateur de créer et d'afficher des requêtes sémantiques : la forge Qualipso.

Comme prévu, les fonctionnalités décrites plus haut ont été développées et incluses dans la forge Qualipso. Celle-ci est donc la première forge à pouvoir utiliser ainsi, comme modèle de donnée intra et inter services, des données sémantiques.

Elle est aussi la première à permettre à un utilisateur de modifier les informations affichées dans son interface graphique grâce à des requêtes sémantiques.

Plus techniquement, la forge Qualipso permet, autant pour le développeur de service que pour l'utilisateur final :

- de créer ses propres prédicats sémantiques
- de créer ses propres relations sémantiques entre les ressources
- de faire des requêtes sur ces relations sémantiques
- d'afficher, de manière totalement intégrée, les résultats de requêtes sémantiques.

C'est, à notre connaissance, la première forge à proposer cela.

De plus, la forge Qualipso permet, par moissonnage, de récupérer des informations sémantiques sur les ressources.

Un tel résultat correspond bien à la proposition de départ : explorer la piste de recherche concernant l'ajout de fonctionnalités « sémantiques » à une forge.

Tout d'abord, elle est bien une source de données pour le « web des données ». En premier lieu, elle fournit les informations sémantiques des ressources, donc est moissonnable par d'autres moteurs sémantiques. Sur ce point, elle ne se contente pas de fournir des données statique sur la ressource, mais elle peut aussi fournir des informations relationnelles, notamment des relations avec d'autres ressources. Ensuite, elle est prête à pouvoir être interrogée par d'autres moteurs, directement à partir de requêtes SPARQL. L'exécution de requêtes fonctionne, il ne manque que la définition d'un « sparql endpoint », c'est-à-dire l'ouverture de manière standard du service d'exécution de requête à l'extérieur.

Ensuite, elle permet bien de faire émerger les ontologies de l'usage social, ceci en permettant à l'utilisateur non seulement de définir ses propres relations, mais aussi ses propres prédicats. L'utilisateur peut même tester et partager ses propres requêtes.

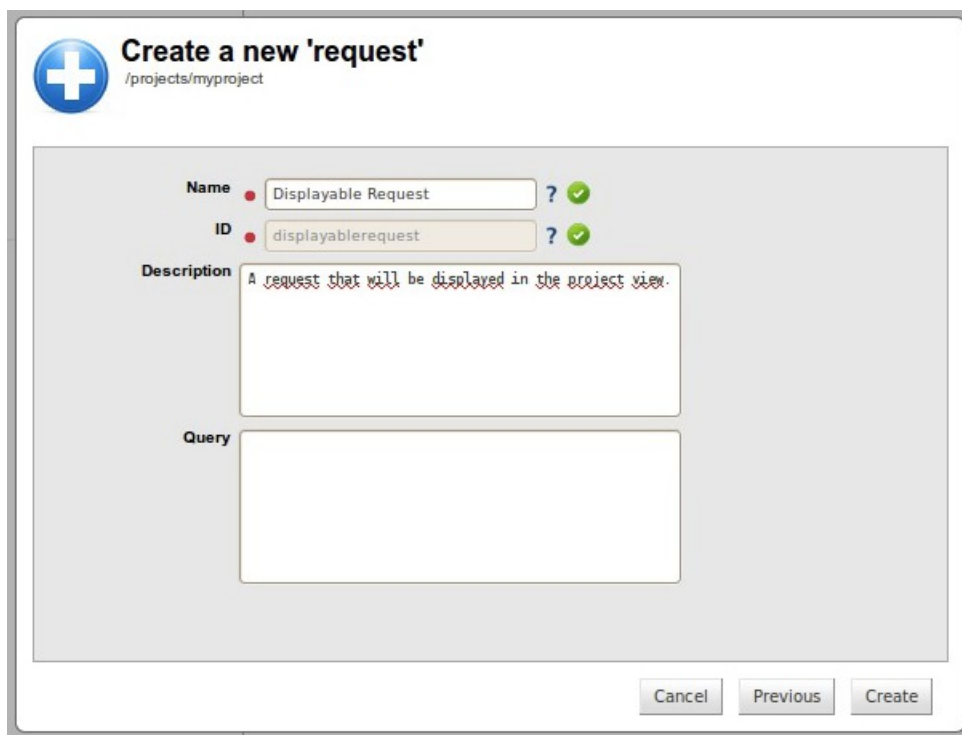
Enfin, la forge est prête pour l'interopérabilité sémantique, soit en tant que source de

données (par moissonnage ou par requêtage), soit en tant que métaforge, par exemple en permettant des liaisons vers des ressources se trouvant en dehors de la forge. Pour l'instant, de telles liaisons ne sont pas possibles, car l'objet d'une relation doit être soit un littéral, soit un chemin dans l'arbre de nommage (donc une ressource locale). Cette expérimentation a notamment permis de soulever le problème de la sécurité qui apparaissait dans l'approche « forge pour les ontologies », du fait de l'interaction entre l'utilisation des relations sémantiques pour la logique métier des services, et la possibilité pour un utilisateur de définir ses propres relations.

3.4.2 Interface graphique

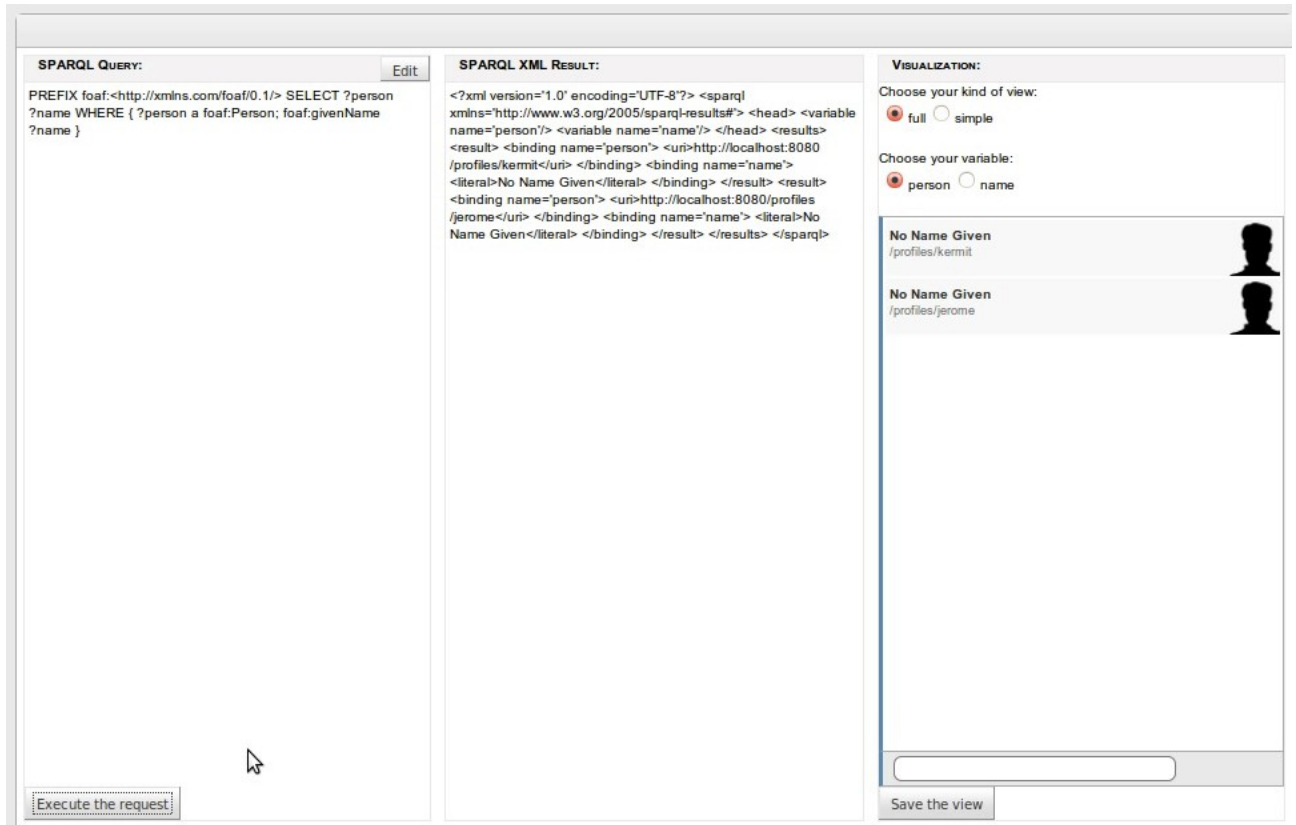
Une interface graphique, en liaison avec l'interface existante, a été réalisée pour la partie « utilisateur final » de la partie sémantique. Il est ainsi possible, pour un utilisateur final, de créer une requête, de la sauvegarder en tant que ressource, de la paramétrer, et de la visualiser comme partie de la vue d'une autre ressource.

3.4.2.1 Création d'une ressource « requête »



La requête est créée de la même manière que n'importe quelle ressource, en passant par le formulaire de création de ressource.

3.4.2.2 Interface de création d'une requête.



The screenshot shows the COCLICO interface for creating a SPARQL query. It is divided into three main sections:

- SPARQL Query:** Contains the query text:


```
PREFIX foaf:<http://xmlns.com/foaf/0.1/> SELECT ?person
?name WHERE { ?person a foaf:Person; foaf:givenName
?name }
```

 There is an "Edit" button and an "Execute the request" button at the bottom.
- SPARQL XML RESULT:** Displays the raw XML output of the query:

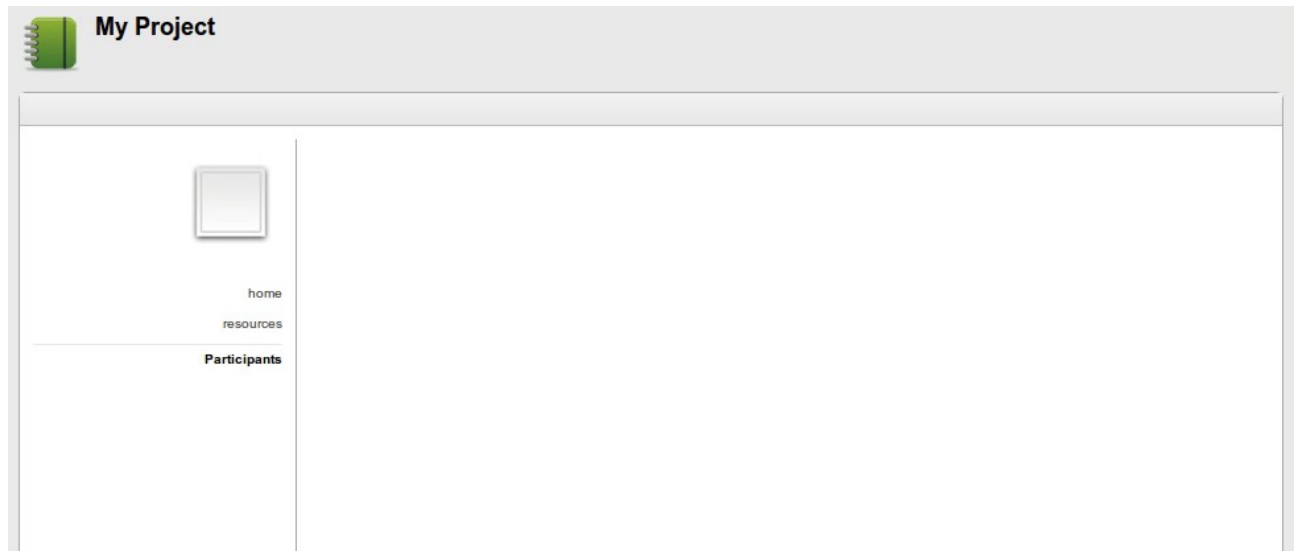

```
<?xml version="1.0" encoding="UTF-8"?> <sparql
xmlns=http://www.w3.org/2005/sparql-results#> <head> <variable
name="person"/> <variable name="name"/> </head> <results>
<result> <binding name="person"> <uri>http://localhost:8080
/profiles/kermit</uri> </binding> <binding name="name">
<literal>No Name Given</literal> </binding> </result> <result>
<binding name="person"> <uri>http://localhost:8080/profiles
/jerome</uri> </binding> <binding name="name"> <literal>No
Name Given</literal> </binding> </result> </results> </sparql>
```
- VISUALIZATION:** Allows for configuring the view. It includes:
 - Buttons for "full" (selected) and "simple".
 - Buttons for "person" (selected) and "name".
 - A preview area showing two entries:
 - "No Name Given /profiles/kermit" with a silhouette icon.
 - "No Name Given /profiles/jerome" with a silhouette icon.
 - A search bar and a "Save the view" button at the bottom.

La requête est entrée à gauche, le résultat brut (en XML) est affiché au milieu. À droite se trouve la configuration de la vue (quelle variable afficher, quel type de vue) ainsi qu'un preview dynamique de la vue obtenue.

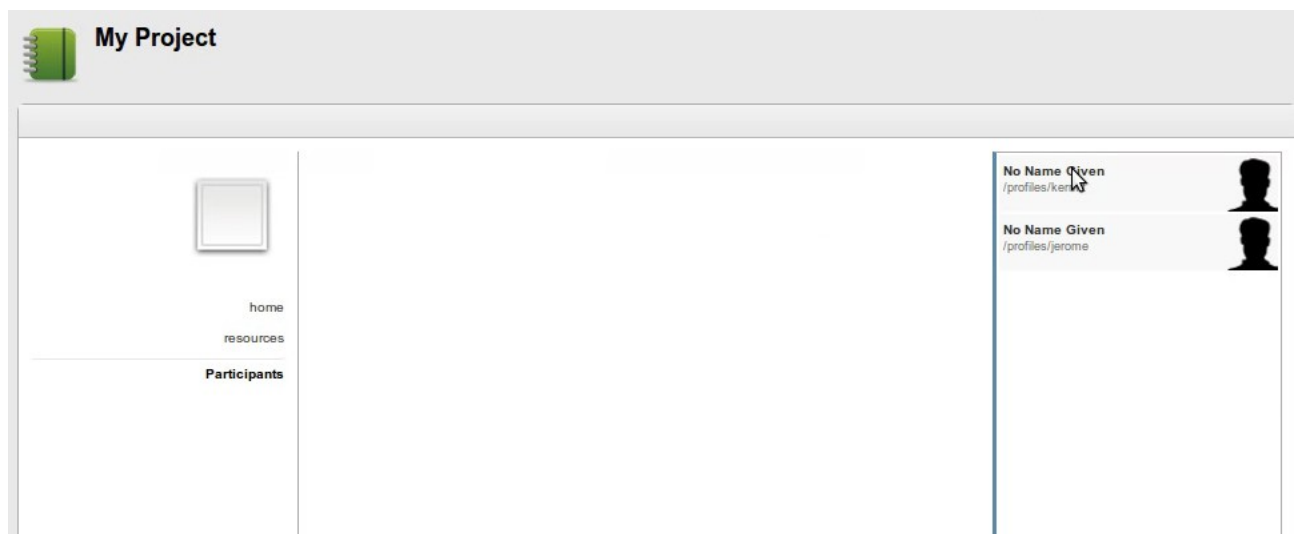
L'utilisateur peut paramétrer, dans le cas d'une requête à plusieurs variables, quelle est la variable qui sera affichée. De même, deux types de vues sont possibles, une vue simple, n'affichant qu'une liste de résultat, et une vue plus riche, comprenant un cadre graphique, une boîte de recherche (propre à la liste), ainsi qu'un algorithme de sélection (permettant de sélectionner un élément de la liste).

Les paramètres de la vue sont stockés dans les propriétés de la ressource, au niveau de l'arbre de nommage. Ainsi, la ressource est indépendante de la paramétrisation graphique.

3.4.2.3 Affichage des résultats d'une requête



Affichage d'une ressource sans vue incluse. Il s'agit de la vue d'un projet.



Affichage d'une ressource avec la vue incluse définie précédemment. On retrouve la vue « requête » identique au preview affiché lors de sa création.

3.4.3 Livrables

Les réalisations faites par l'équipe SCORE dans le cadre du projet COCLICO ont été faites sur la forge Qualipso. Elles se retrouvent donc majoritairement dans le code source de la forge Qualipso. De plus, plusieurs présentations vidéos ont été faites, afin de démontrer les fonctionnalités proprement dites. En particulier, une vidéo de démonstration a été réalisée pour la conférence OWF 2010 (Open World Forum).

Les livrables sont donc, en plus de ce rapport, le code source de la forge Qualipso (qui est sous licence open-source LGPL) ainsi que les vidéos de démonstrations. Ces dernières sont accessibles en ligne.

3.4.4 Code source

Le code source de la forge Qualipso se trouve sur la forge INRIA, dans le projet du même nom. L'accès aux sources se fait soit en utilisant subversion (un outil de gestion de version), soit directement par l'interface web, à l'adresse suivante :

https://gforge.inria.fr/scm/?group_id=1872

Le code est sous licence LGPL.

3.4.5 Vidéos

Trois vidéos de démonstrations ont été réalisées. Afin de faciliter la distribution, ces vidéos ont été mises en ligne.

3.4.5.1 Vidéo de démonstration pour OWF (Open World Forum)

Cette vidéo, faite pour la conférence OWF de fin septembre 2010, contient la démonstration de l'accès à la description RDF des ressources, ainsi qu'un tout premier jet montrant la possibilité d'un affichage sommaire des résultats d'une requête.

Cette vidéo est accessible à l'adresse suivante : <http://vimeo.com/15527478>

3.4.5.2 Vidéo de démonstration du requêtage

Cette vidéo, réalisée en avril 2011, contient la démonstration de toute la partie « requêtage » de la forge Qualipso. On y voit comment créer et sauvegarder une requête, puis comment configurer l'affichage d'une requête. En particulier, on voit la démonstration d'un affichage inclus dans la vue d'une autre ressource.

Cette vidéo est accessible à l'adresse suivante : <http://vimeo.com/21798611>

3.4.5.3 Vidéo de démonstration de l'affichage spécifique

Cette vidéo, réalisée en avril 2011, contient la démonstration de l'affichage spécifique des résultats d'une requête sémantique. On y voit comment l'affichage est spécifique suivant le type de la ressource faisant partie d'un résultat, et ceci même si plusieurs types différents font partie du même résultat. On y voit aussi que l'affichage est délégué au service spécifique, avec notamment la création d'une vue « preview » interactive de la ressource.

Cette vidéo est accessible à l'adresse suivante : <http://vimeo.com/21802933>

3.5 Conclusion

3.5.1 Ce qui est fait

Comme prévu, la forge Qualipso, enrichie des réalisations faites dans COCLICO est un prototype utilisable démontrant l'utilisation de l'approche « forge pour les ontologies ». Elle possède les fonctionnalités suivantes :

- affichage des informations sémantiques des ressources
- gestion des prédicats comme des ressources
- gestion des requêtes comme des ressources
- affichage de manière intégrée des résultats des requêtes.

La participation à ce projet a permis d'avancer dans l'exploration de la piste de recherche

« forge pour les ontologies », en montrant ce qui était faisable et ce qui posait problème, ainsi que les bénéfices que cela apportait.

3.5.2 Ce qui manque

Pour différentes raisons (notamment de sérieux problèmes de santé d'un des principaux membres de l'équipe affecté au projet), toutes les expérimentations et réalisations prévues n'ont pas pu être réalisées.

En premier lieu, il manque principalement les interfaces graphiques permettant à un utilisateur final de définir ses propres prédicats ainsi que ses propres relations sémantiques.

Ensuite, il manque les expérimentations avec d'autres forges ou d'autres moteurs sémantiques, notamment au niveau moissonnage. Pour la partie meta-forge, le problème des liaisons vers des ressources extérieures à la forge n'a pas été résolu. De plus, il fut décidé tardivement qu'un module de compatibilité OAuth-JAAS serait réalisé, afin de faciliter l'interopérabilité au niveau des droits. Hélas, vu le temps imparti, seule l'analyse technique eut le temps d'être faite (avec la conclusion qu'une telle compatibilité était bien techniquement possible).

Finalement, il manque la rédaction d'un article de recherche décrivant l'expérimentation réalisée, autant au niveau de l'architecture et des choix techniques qu'au niveau des résultats obtenus ou des problèmes rencontrés.