Sujet de Master recherche 2010
MARRE ! (Mini AXML Réalisé en REST)
Conception d'une plateforme d'expérimentation minimale pour les Web Services

Encadrants : Loïc Hélouët
Equipe : Distribcom
mail: loic.helouet@irisa.fr
motc clés : Web Services, documents actifs, REST.

1 Résumé

Les systèmes d'information évoluent de plus en plus vers des architectures de type "Web Services" [6], dans lesquelles des applications sont conçues par assemblage (on parle aussi d'orchestrations) de composants logiciels répartis sur le web. Ces composants doivent être coordonnées pour fournir des services de plus haut niveau. Les composants de base sur lesquels s'appuie une solution à base de services ne sont pas nécessairement conçus et maintenus par le créateur de l'application, et peuvent provenir de sites distincts. On peut même imaginer des orchestrations de services dans lesquelles des composants sont mis en compétition pour fournir la meilleurs réponse (la plus rapide ou la plus complète) à un utilisateur. Un concepteur de Web Services qui réutilise les solutions fournies par d'autres sites n'a pas a priori accès aux détails des implémentations et ne les perçoit que par ses interactions avec elles, c'est a dire en tant qu'environnement de son propre système.

Dans ce contexte, un concepteur ne dispose pour les services qu'il invoque que de descriptions sommaires par des *interfaces*, qui indiquent comment appeler un service (adresse sur le web, mais aussi paramètres des appels) et qu'attendre en retour. Bien que les informations disponibles sur l'environnement soient assez parcellaires, il est important de pouvoir raisonner (par exemple vérifier des propriétés de sûreté) sur les comportements d'une orchestration, afin de donner des garanties aux utilisateurs du service que l'on définit de la sorte. Cependant, il existe souvent d'énormes différences entre le modèle théorique d'une application (son modèle formel), et son implémentation réelle. Une première solution pour minimiser ces différences consiste à synthétiser automatiquement du code à partir d'un modèle, en assurant que le code produit préserve les bonnes propriétés du modèle initial (par exemple à l'aide d'un compilateur certifié). Une autre approche consiste à rendre raisonner sur des *modèles exécutables*.

Le concept de *document actif*, initialement propose par Serge Abiteboul [1, 2] est un formalisme permettant d'implémenter et de raisonner sur les comportements de Web Services. Un document actif est un document dont certaines parties ne sont pas explicitées (ce sont simplement des références à d'autres services), mais qui peuvent être évaluées par un appel aux services référencés (on peut se représenter cette situation comme un document html contenant des références à d'autres pages web). Les documents actifs permettent de concevoir des services de haut niveau, mais surtout, rendent possible des évaluation "paresseuses" de données : plutôt que de renvoyer un énorme document dont une large partie n'est pas utilisée dans 90% des cas, un service peut retourner un document contenant les informations les plus utiles, et un lien vers un service à invoquer pour obtenir le reste.

L'équipe Distribcom a proposé récemment une version distribuée de documents actifs [5]. Les principes

généraux de ce langage sont très simples : une architecture Web Service est décomposée en sites. Les sites possèdent des données et des services gardés, invocables localement lorsque leur garde est satisfaite, ou par les autres sites. Les réponses aux invocations de services et les ajouts de données sont faits à l'aide de requêtes (queries). L'évaluation de la valeur à retourner par un service peut necessiter l'appel récursif d'un ou plusieurs sous-services. L'implémentation réelle d'un tel système consiste tout simplement à animer un modèle, c'est à dire lancer localement des requêtes sur une base de données, et assurer la communication entre sites.

De précédentes implémentation d'AXML ont été réalisées par l'équipe de S. Abiteboul au dessus de SOAP [7]. Au moment de la conception d'AXML, SOAP était certainement la solution la plus adaptée pour réaliser une plateforme de démonstration. Cependant, ce choix alourdit les communications entre sites, et d'autres solutions, comme REST [3] ont été popularisées depuis. L'équipe Distribcom souhaite se doter d'une plateforme d'expérimentation AXML distribué, fonctionnant à l'aide de REST. L'idée est de limiter les développements au minimum nécessaire pour animer des modèles. Ainsi, implémenter un prototype d'AXML distribué devrait consister à manipuler des arbres localement sur chaque machine (recherche de pattern [4], ajouts de noeuds,...) , et implémenter les invocations entre sites à l'aide de REST.

2 Compétences requises

La programmation du prototype de démonstration se fera en JAVA. Une connaissance de ce langage est un atout pour commencer. Attention : bien que ce stage puisse apparaître comme un travail d'implémentation, il nécessitera une bonne compréhension du formalisme AXML, des compétences en algorithmique, et l'utilisation de solutions empruntées au domaine des applications distribuées.

References

[1] S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, and R. Weber. Active XML: A data-centric perspective on web services. In *BDA*, 2002.

[2] S. Abiteboul, O. Benjelloun, and T. Milo. Positive active XML. In *PODS*, pages 35–45, 2004.

[3] Roy Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[4] Georg Gottlob, Christoph Koch, and Reinhard Pichler. Efficient algorithms for processing xpath queries. *ACM Trans. Database Syst.*, 30(2):444–491, 2005.

[5] Loïc Hélouët and Albert Benveniste. Document based modeling of web services choreographies using active xml. In *ICWS*, pages 291–298, 2010.

[6] Ed Ort. Service-oriented architecture and web services: Concepts, technologies, and tools. Technical report, SUN, 2005. http://java.sun.com/developer/technicalArticles/WebServices/soa2/soa2.pdf.

[7] W3C. Soap version 1.2. Recommandation W3C, June 2003. http://www.w3.org/2002/07/soap-translation/soap12-part0.html.