Les Architectures Orientées Services (SOA)

Ulrich Duvent

Guillaume Ansel

Master 2 Informatique ISIDIS

2009 -2010

Encadrant : H. Basson

Université du Littoral Côte d'Opale
50, Rue Ferdinand Buisson – BP 699 – 62228 Calais Cedex
Téléphone (33) 03.21.46.36.92
Télécopie (33) 03.21.46.06.83

1. Sommaire

2. Introduction

Les SOA (*Service Oriented Architecture* - ou Architecture Orientée Services) sont souvent assimilés à des technologies mais ce sont en réalité des principes d'architectures. En effet, la notion de SOA renvoie à une nouvelle manière d'intégrer et de manipuler les différentes briques et composants applicatifs d'un système informatique (comptabilité, gestion de la relation client, production, etc.) et de gérer les liens qu'ils entretiennent. Cette approche repose sur la réorganisation des applications en ensembles fonctionnels appelés services. Avant l'arrivé des SOA, les services d'une entreprise étaient développé au sein d'une application monolithique c'est-à-dire déployé sur un serveur central. Petit à petit, le modèle distribué s'est mis en place, permettant plus de souplesse dans la gestion du système d'information. En effet, les systèmes d'informations des entreprises sont constitués d'applications et de données constituants leurs héritages. Avec la fusion des groupes et l'évolution des technologies ces héritages deviennent hétérogènes et ont tendance à se spécialiser par métier au travers des services. On peut alors avoir une vision globale du système d'information d'une entreprise. Ainsi, une nouvelle conception est apparue utilisant les modèles distribués. Celle-ci, conceptualisée par le Gartner Group, le SOA tente de s'imposer en rendant les applications plus souple et réutilisable. Les SOA s'appuient sur des standards et peuvent fonctionner dans des environnements hétérogènes. Le principal but des SOA est d'améliorer l'interopérabilité entre les systèmes sans créer de contraintes fortes. En effet, les différents services n'ont pas besoin de répondre aux mêmes contraintes structurelles tant qu'ils respectent un contrat. Il existe une faible interdépendance entre les services. Les SOA sont utilisés pour le développement des applications à long termes. Ils induisent une bonne conception et donc une maintenance du code facilité ainsi que l'ajout de nouveaux services sans endommager l'application existante.

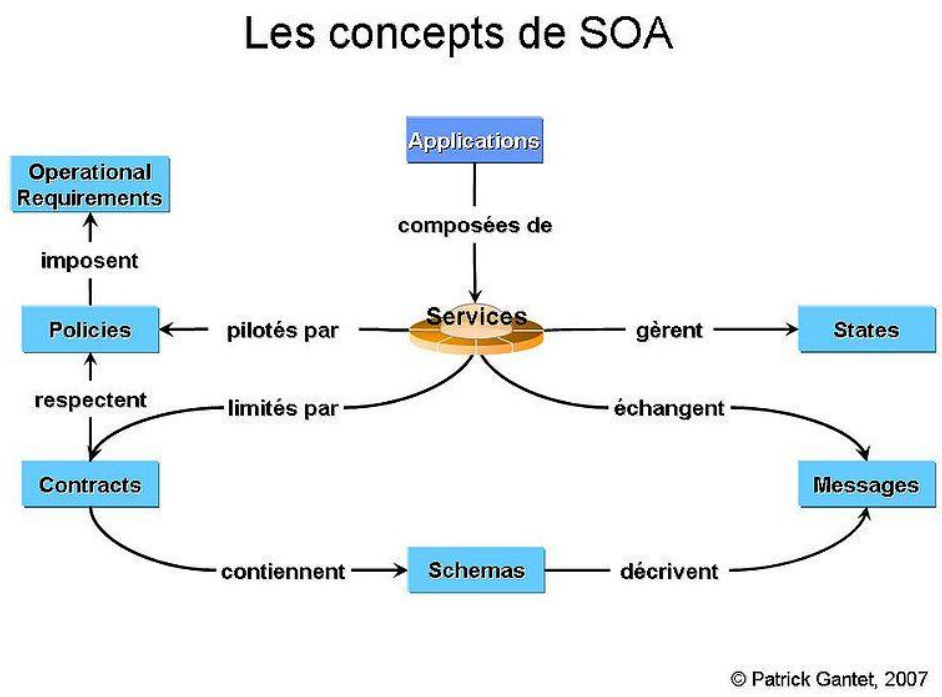
Dans ce document, nous allons essayer de voir les avantages d'utiliser ce modèle de conception. Tout d'abord, il faut donc étudier les différentes définitions relatives à un SOA. Dans une deuxième partie, nous allons nous intéresser aux performances ainsi qu'aux inconvénients de mettre en place ce type de conception. Et enfin, nous allons observer un exemple d'architecture SOA.

3. Fonctionnement des Architectures Orientées Services

3.1. Définition d'un SOA

Une architecture orienté service est une architecture logicielle s'appuyant sur un ensemble de services simples. Ils sont développés en s'inspirant des processus métier de l'entreprise. Un service est un composant fonctionnant de manière autonome et offrant des fonctionnalités métiers à d'autres applications ou d'autres services. Ces services représentent les fonctions basiques des fonctionnalités des entreprises. Ils dialoguent entre eux au travers de bus ou par Internet, on parle alors de *WebService* (WSOA). Les échanges peuvent se faire de manière synchrone ou asynchrone. L'entreprise s'enrichit de services mutualisables permettant de répondre rapidement et avec souplesse aux demandes du marché. En effet, ils correspondent à un processus métier mutualisable au niveau de l'entreprise. Cela permet les changements au niveau informatique des décisions stratégiques et tactiques de l'entreprise.

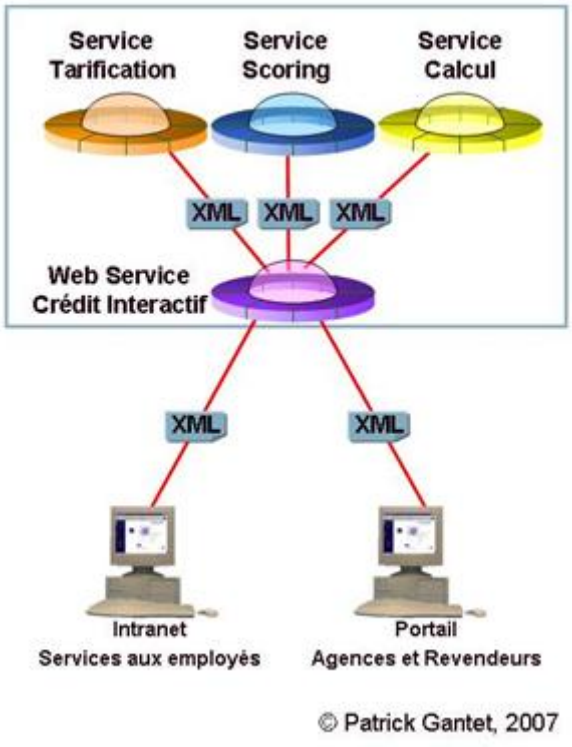
Il n'existe pas spécifications officielles pour l'architecture d'une SOA. Il peut-être décrit par la notion, la description, la publication et l'invocation de services. La notion de service est une fonction encapsulée dans un composant que l'on peut interroger à l'aide d'une requête composée d'un ou plusieurs paramètres et fournissant une ou plusieurs réponses. Idéalement chaque service doit être indépendant des autres afin de garantir sa réutilisabilité et son interopérabilité. La description de service est la manière de décrire les paramètres d'entrée du service, le format et le type des données retournées. Le principal format de description de services est WSDL (*Web Services Description Language*), normalisé par le W3C. Ensuite la publication consiste à publier dans un registre (en anglais *registry* ou *repository*) les services disponibles aux utilisateurs, tandis que la notion de découverte recouvre la possibilité de rechercher un service parmi ceux qui ont été publiés. Le principal standard utilisé est UDDI (*Universal Description Discovery and Integration*), normalisé par l'OASIS. Enfin l'invocation représente la connexion et l'interaction du client avec le service. Le principal protocole utilisé pour l'invocation de services est SOAP (*Simple Object Access Protocol*).



Comme dit précédemment, les SOA sont des concepts créées par le Gartner Group. Sur la page suivante, un schéma montrant le fonctionnement des concepts de SOA.

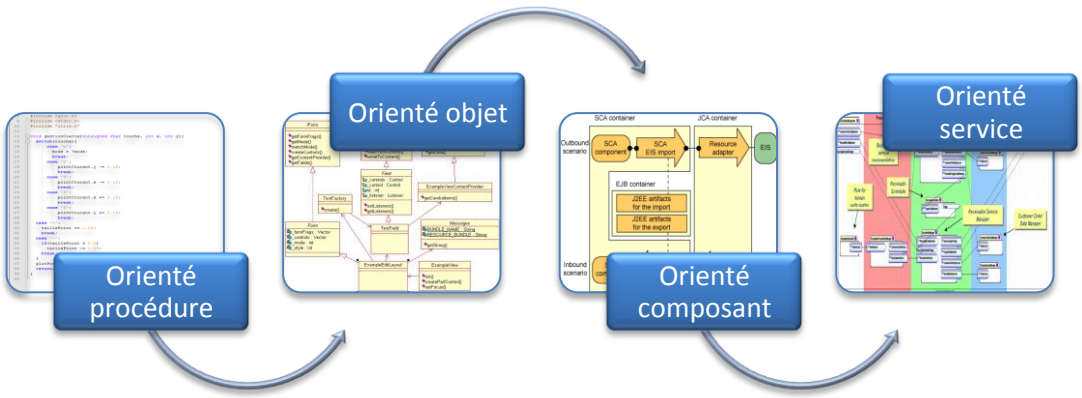
Les services au centre de ce schéma sont les unités atomiques d'une architecture orientée services. Ces services communiquent entre eux par le biais de messages. Ces communications peuvent être synchrones ou asynchrones. Le langage de programmation du service n'a aucune importante, ainsi que la plateforme d'exécution, matérielle et logicielle. Ces services sont limités par un ensemble de contrats, c'est-à-dire qu'un service doit respecter un ensemble de règle de fonctionnement.

Enfin dans les WSOA (*WebService Oriented Architecture*), les *WebServices* sont des services disponibles via Internet par le biais d'URL. Pour les WSOA, les données sont stockées sur le serveur. On peut se connecter aux *WebServices* à l'aide de terminaux mobiles. L'utilisateur dispose donc à partir d'un client léger la puissance de calcul du serveur. La connexion à distance est réalisé par protocole http, on a donc une indépendance de tout langage de programmation et plateformes. La communication entre les différents acteurs du système est réalisée par la technologie XML. Le schéma suivant montre le fonctionnement d'un WSOA entre différents terminaux et les *WebServices*.



3.2. Cycle de vie des services

Une des plus grandes fonctionnalités des SOA est la réutilisation des services pour plusieurs entreprises. En effet, en étudiant l'évolution des différentes architectures, on peut observer une réutilisation croissante. Ceci était impossible dans le modèle des composants.



En passant du procédural à l'orienté objet, on a la notion de réutilisation de classes d'objet. Celle-ci n'a cessé d'augmenter avec la vision orienté composant puis enfin service.

3.3. Déploiement des SOA

Le déploiement des SOA peut-être effectué sur des serveurs d'intégrations, EAI (*Enterprise Application Integration*). Ces serveurs organisent la circulation de l'information entre les applications et les rendent interopérables en développant des connecteurs, middleware, permettant d'interfacer ces applications utilisant des protocoles de communications différents généralement propriétaire. L'EAI permet donc la connexion aux applications, les conversions des informations dans un langage commun et le transport des informations d'une application vers une autre.

Le fonctionnement de l'EAI repose sur les composants suivants. Tour d'abord, un référentiel des objets métier de l'entreprises ou de l'ensemble des processus. Ensuite, un moteur de gestion de règles, des connecteurs applicatifs permettant l'interface avec les applications et les données de l'entreprise, et enfin un système de transport des informations. Il y a plusieurs possibilités pour le transport des informations. On peut utiliser un hub, un middleware (une couche logicielle qui s'intercale entre les applications et un système, un MOM (middleware orienté messages). Dans le dernier cas, les communications sont asynchrones.

L'EAI permet un accès universel et un partage de toutes les données et composants d'un système d'information, qu'ils soient normalisés, propriétaires ou incompatibles. Il nécessite peu ou pas de modifications des applications ou structures de données qu'il intègre. Une architecture EAI n'est pas figée. Elle constitue un socle évolutif, réutilisable et dynamique suivant l'évolution des besoins métier spécifiques comme les évolutions structurelles de l'entreprise.

Cependant, les plateformes J2EE et .Net répondent aussi aux besoins des systèmes informatiques distribuées et services Web.

4. Les apports des Architectures Orientées Services

4.1. Inconvénients

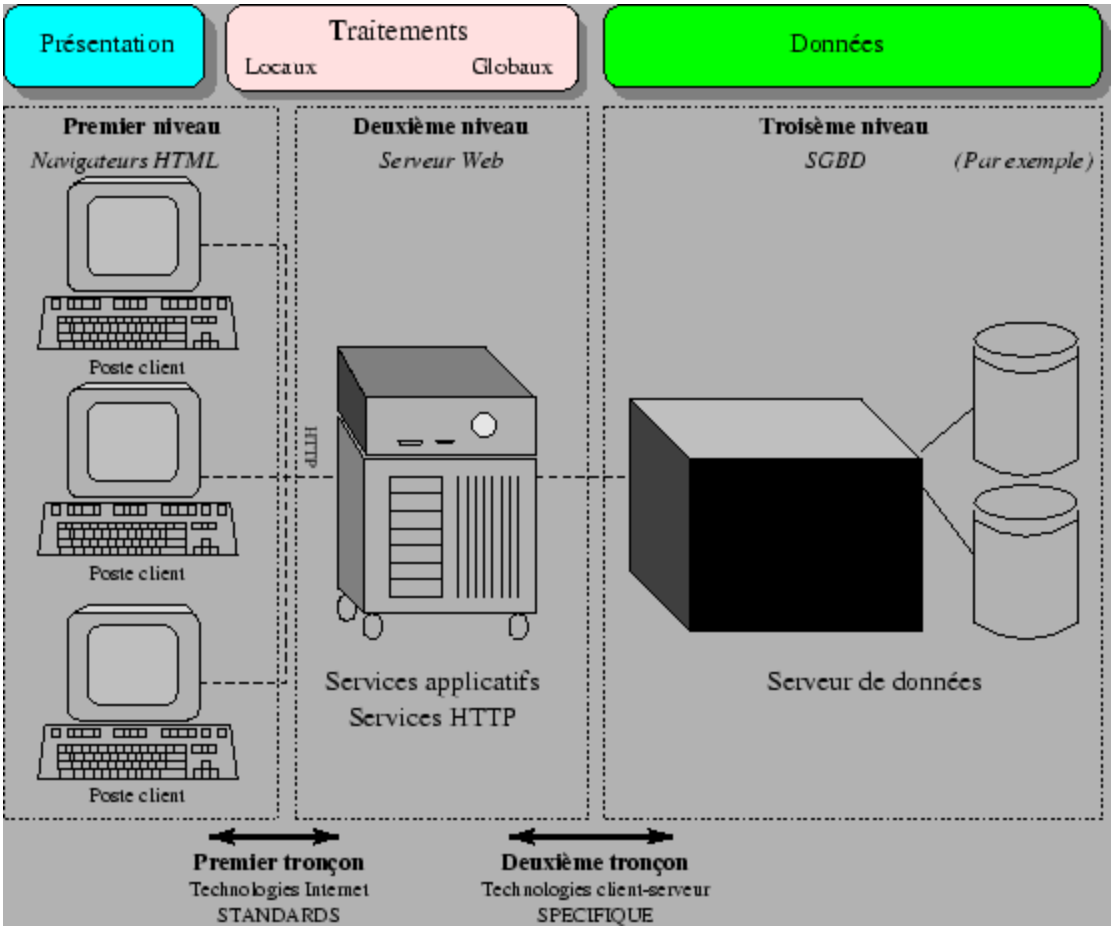
La mise en place d'un SOA a un coût élevé à la fois financier et humain. Il faut former une équipe d'experts en conceptions ainsi que plusieurs équipes pour développer et administrer les différents services. Dans le cas idéal, l'activité de l'entreprise doit être pensée autour des services. La conception du système d'information est donc une étape initiale critique. Si le fonctionnement de l'entreprise n'est pas organisé autour des services alors il est difficile d'utilisé un SOA et donc le coût de fonctionnement sera élevé. En effet, les SOA ont un intérêt limité si l'entreprise ne base pas ses processus sur l'exploitation des services, il faut donc concevoir des *workflows* adaptés. De plus, il est difficile de migrer d'une architecture monolithique vers une architecture SOA sans étude préalable efficace.

4.2. Avantages

Tout d'abord, les SOA disposent de tous les avantages d'une architecture client-serveur. C'est à dire une plus grande modularité, on peut facilement remplacer un composant (service) par un autre. Ils offrent également la réutilisation des composants, des meilleures possibilités d'évolutions, une plus grande tolérance aux pannes ainsi qu'une maintenance plus facile. Il est très facile dans les SOA d'ajouter des nouveaux services sans venir perturber les existants ainsi que de les faire évaluer. Ces service sont réutilisables, il est donc possible de bâtir de nouvelles applications faisant appel à des services tiers. Par exemple, un ERP d'entreprise peut-être créée en faisant appel aux services SAP. Les services sont utilisés selon les besoins des entreprises et la mise à jour de ces services est réalisée de manière totalement transparente pour les consommateurs.

Les SOA ont une architecture n-tiers, c'est-à-dire la séparation de la présentation des données du traitement et de la base de données. Pour une architecture client/serveur, la présentation des données ainsi que le traitement sont réalisés par le client et le serveur utilisé pour la base de données. Par contre, en n-tiers, chaque serveur peut avoir une fonction unique. Par exemple, si on prend une architecture 3-tiers, on a les terminaux pour les requêtes http qui gèrent la partie présentation de l'interface homme-machine. On a un serveur d'application pour gérer les traitements et un dernier serveur de données qui gère les accès aux données. Une architecture n-tiers présente la première infrastructure informatique pour un travail coopératif. Les avantages sont une centralisation des traitements au niveau des serveurs. En effet les services sont stockés sur des serveurs, comme montré sur le schéma ci-dessus. Elle permet également une gestion plus simple de la cohérence et de l'intégrité des données. Sur la page d'après, un schéma illustrant une architecture 3-tiers.

Architecture 3-tiers



Les applications clientes et les services correspondent entre eux à l'aide de middleware, des couches logicielles permettant la communication inter-application.

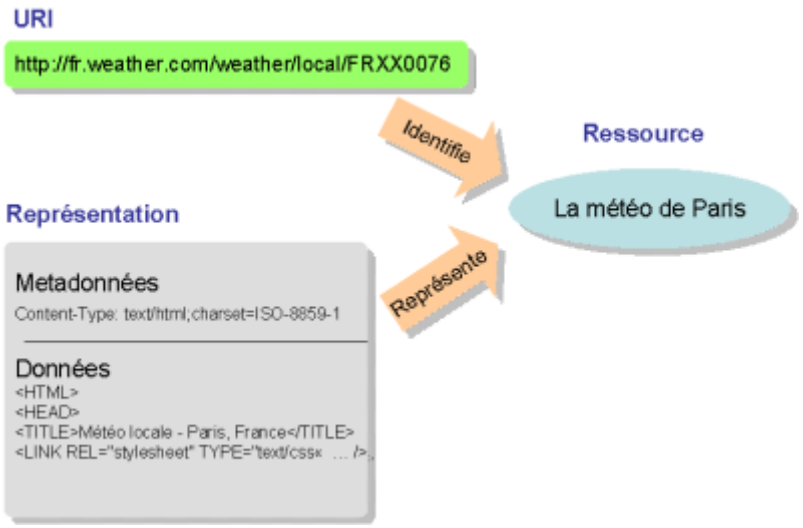
4.3. Un exemple de SOA : REST

Pour montrer les performances des SOA, intéressons nous plus particulièrement à un exemple concret. REST est l'acronyme de "Representational State Transfer" inventé par Roy T. Fielding et dont la définition a été donnée dans sa thèse de doctorat sur les architectures Web en 2000. REST est un style architectural inspiré des SOA, mais basé sur le Web, il ne définit pas une spécification, mais explique un principe de développement à suivre. Il s'appuie sur l'utilisation des standards fondamentaux d'Internet. L'adressage des ressources est réalisé par les URI, *Uniform Ressource Identifier*, une chaine de caractères permettant d'identifier une ressource sur un réseau. Ceci permet donc d'identifier les différents services disponibles sur le réseau ainsi que des images, vidéos... Ensuite la communication avec le service est réalisée par protocole HTTP et la représentation des données est de type MIME

(Multipurpose Internet Mail Extensions). Par exemple, si les données sont de type html, on aura « text/html ». Il existe différents type : text/css, text/html, video/mpeg, image/png …

L'utilisation du protocole HTTP définit également l'interface du service grâce aux principaux verbes définit par celui-ci. Cette interface est standard et implémenté par tous les serveurs Web. Les services proposent des méthodes qui répondent aux requêtes http. On a « GET » qui permet de récupérer des données, « POST » qui envoie une commande, « PUT » pour mettre à jour une donnée et « DELETE » pour la suppression. Cependant, REST ne gère pas l'état du client (session), les données client doivent être stockées ailleurs, par exemple dans des bases de données, les cookies …
REST propose donc une architecture légère donnant de bonnes performances et tirant parti des fonctionnalités du protocole HTTP implémentées dans les serveurs (caching, etc.). La conception de l'architecture d'une application en REST commence par l'identification des ressources disponibles. Ensuite, chaque ressource est définit par une URI permettant son accès de manière unique et sûre. Les scénarios de navigations sont définis grâce à des hyperliens auxquels on accède en « GET ». Et les clients récupèrent les informations des ressources consommées directement, il n'y a pas de sessions. Enfin, le filtrage du site est facilité par le biais des URI.

Exemple d'application en REST



Sur ce schéma, on peut observer un exemple de représentation d'une application en REST. La ressource « La météo de Paris » est identifiée et accessible avec une URI. Les données renvoyées par cette ressource sont représentées sous forme d'une page HTML.

5. Conclusion

L'utilisation des SOA a donc pour but d'avoir une vision globale du système d'information d'une entreprise. Il représente la capitalisation des ressources métier d'une entreprise avec la réutilisation des services et le stockage des données sur une base de données unique. L'utilisation d'une telle architecture facilite l'évolution des applications mais demande beaucoup d'efforts au niveau conception. Cependant, des architectures types existent et se développent de plus en plus. Elles sont également validées par des experts.

Mais les entreprises possédant des applications centralisées développées de manière monolithique n'ont pas de bénéfices à adopter ce type d'architecture. En effet, l'application doit être pensée orientée services dès le début, afin d'orienter les *workflows* vers cette notion de service.

De plus, la plupart des applications dans les entreprises sont abandonnées. L'apparition de ce type d'architecture permet donc diminuer ce point de vue et chaque service peut être réutilisé pour une autre application.

Ainsi, cette architecture doit être mise en place au plus tôt au risque de voir son coût et ses avantages diminués.

Pour conclure, voici quelques chiffres. En 2004, un budget de 950 millions de dollars est consacré pour le marché mondial des Web Services. Selon Radicati Group, il devrait atteindre 6,2 milliards de dollars en 2008. Enfin, d'après le groupe Gartner, 60% des entreprises opéreront leurs applications métiers par le biais d'une architecture SOA d'ici 2008.

6. Annexes

6.1. Quelques définitions

6.1.1. SOA

Service Oriented Architecture, architecture orientée service, préconisant un système d'information bâti à partir de processus aisément configurable reposant sur des services applicatifs autonomes.

6.1.2. Service Web

Composant applicatif présentant une interface standardisée (au format WSDL), accessible au travers d'Internet grâce au protocole SOAP.

6.1.3. WSOA

Web Service Oriented Architecture, SOA mise en œuvre grâce à l'utilisation de services Web et de leur annuaire de référencement UDDI.

6.1.4. ERP

Entreprise Resource Planning, ou PGI pour Progiciel de gestion intégré en français. Logiciel permettant de gérer l'ensemble des processus d'une entreprise. Les données sont standardisées sur une seule base de données commune à toutes les applications de l'entreprise.

6.1.5. Workflow

La modélisation et la gestion informatique de l'ensemble des tâches à accomplir et les communications entre les différents acteurs impliqués pour la réalisation d'un processus métier.

6.1.6. HTTP

Hypertext Transfert Protocol est le standard sur Internet pour les liaisons entre les différentes ressources à partager.