Introduction Aux Architectures Orientées Services

I. Introduction :

A partir des années 80 les entreprises commençaient à s'orienter vers l'informatisation de leurs systèmes d'information en se basant sur les grands systèmes qui existaient à l'époque. Puis, dans les années 1990, se sont développés la microinformatique et les systèmes ouverts (basés sur Unix). Depuis la fin des années 1990 et dans les années 2000, le monde entier connaît un fort développement d'internet, de la dématérialisation et de la production des biens immatériels. Les besoins de l'entreprise et de leurs clients augmentaient d'une manière excessive, ainsi la situation des développeurs d'application dans une entreprise est devenue très critique, ils se retrouvaient en défi face à deux situations nécessitant un grand changement :

Situation interne qui nécessite un changement : caractérisée par des systèmes composés de **dizaines** voire de **centaines** d'applications qui :

- Communiquent entre elles directement ou à travers des messages en utilisant différents protocoles de communication.
- Utilisent des formats de messages (de données) différents.
- Se retrouvent dans des serveurs locaux ou distants.
- Sont écrites dans des langages différents (Java, C#, C++, Python ou pire encore COBOL ou Clipper...)

Q_1) Comment faire communiquer l'application développée avec les autres ? Ou pire encore comment faire communiquer toutes les applications entre elles ?

Q_2) Comment connaitre les produits et fonctionnalités déjà implémentés dans d'autres applications pour pouvoir les réutiliser et éviter de les ré-implémenter à nouveau « éviter de faire réinventer la roue » pour gagner à la fois du temps et éviter la redondance ?

Q_3) Comment localiser les applications (déterminer leurs URL ou leur IP) pour pouvoir communiquer avec elles ? Et pire encore comment faire si une application change d'adresse ?

Q_4) Comment faire tourner les applications même en absence ou défaut de connexion réseau ?

Ce sont des exemples parmi d'autres de problèmes auxquels les développeurs étaient confrontés avant le développement de l'architecture orientée services (SOA).

Situation externe qui procure de nouvelles exigences : caractérisée par :

- Changement du créneau commercial actuel,
- Augmentation des contraintes de concurrence,
- Évolution et bouleversement des besoins du client,
- Evolution rapide de la technologie de l'information.

E_1) Comment développer et mettre en place des systèmes flexibles qui peuvent rapidement s'adapter avec le changement de l'environnement du commerce pour qu'il permet à l'entreprise d'y garder sa place.

E_2) Comment concevoir et développer des produits réutilisables afin d'éviter les remises en cause de l'existant à chaque fois qu'il y a de changements.

E_3) Comment permettre l'interaction avec le système à travers des moyens et outils très simples (surtout avec l'apparition du protocole IPv6).

Devant ces situations critiques, une petite lumière s'est éclairée au début et milieu des années 90, qui a été exploitée plus tard (après l'apparition du langage XML) et qui a abouti à une nouvelle architecture qui apporte solution à l'ensemble des problèmes cités, cette architecture s'appelle « **Architecture Orientée Services** » en anglais « ***Services (Soft) Oriented Architecture*** ».

II. Notions de la SOA :

1) Historique :

En 1992 ***Gio Wiederhold*** a développé une nouvelle vision de l'architecture du traitement de l'information en entreprise en tentant de régler la problématique de l'accès et de l'intégration de l'information en introduisant la notion de ***médiateur*** (***architecture médiane***).

Un ***médiateur*** est un composant logiciel qui exploite des connaissances encodées concernant un ensemble de données et crée les informations nécessaires à une couche de niveau supérieur d'une application.

Ceci constituât l'idée principale qui va être exploitée plus tard et permettre à un groupe de recherche appelé "***Gartner Group***" *(un des plus importants fondateurs de cette architecture)* de fonder les bases de la ***SOA en 1996***. L'exploitation de cette architecture n'a commencé à apparaître qu'à partir de l'année ***2000*** après le succès du langage ***XML*** (apparu en ***1998***).

En fait le terme ***SOA*** n'est popularisé qu'à partir de l'année ***2002*** après l'orientation de beaucoup d'entreprises commerciales vers cette architecture.

2) Définition :

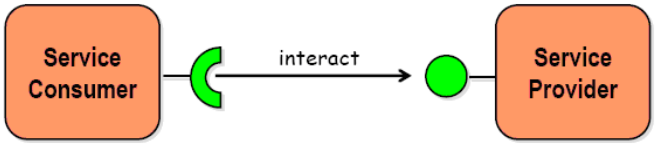
L'Architecture Orientée Services (***SOA : S****ervices "Soft"* ***O****riented* ***A****rchitecture*) est une approche de conception pour les systèmes d'information, qui se base sur la (répartition de l'activité commerciale sur ***3 acteurs*** : ***client, fournisseur de service et annauire***) mise en œuvre des différents services disponibles sur le réseau, et qui permet l'interaction entre eux tout en gardant:

- Une ***forte cohérence interne*** en utilisant un format d'échange ***"pivot" (XML)***.
- Un ***couplage faible "lâche"*** entre services (indépendance entre services) en utilisant une couche d'interface interopérable (service web WS-*).

L'architecture orientée services SOA est une réponse très efficace aux problématiques que rencontrent les entreprises en terme de ***réutilisabilité***, ***d'interopérabilité*** et de ***réduction*** de couplage entre les différents systèmes qui implémentent leurs SI [1].

Selon le groupe ***«Gartner »*** (*définition donnée en Septembre 2005)* :

- *L'architecture orientée service constitue un style d'architecture basée sur le principe de séparation de l'activité métier en une série de services.*
- *Ces services peuvent être assemblés et liés entre eux selon le principe de couplage lâche pour exécuter l'application désirée.*
- *Ces services sont définis à un niveau supérieur à celui de l'approche « Composant » traditionnelle.*



III. Principes fondamentaux :

1) Paradigme SOA :

Le paradigme SOA se base sur 3 actions principales:

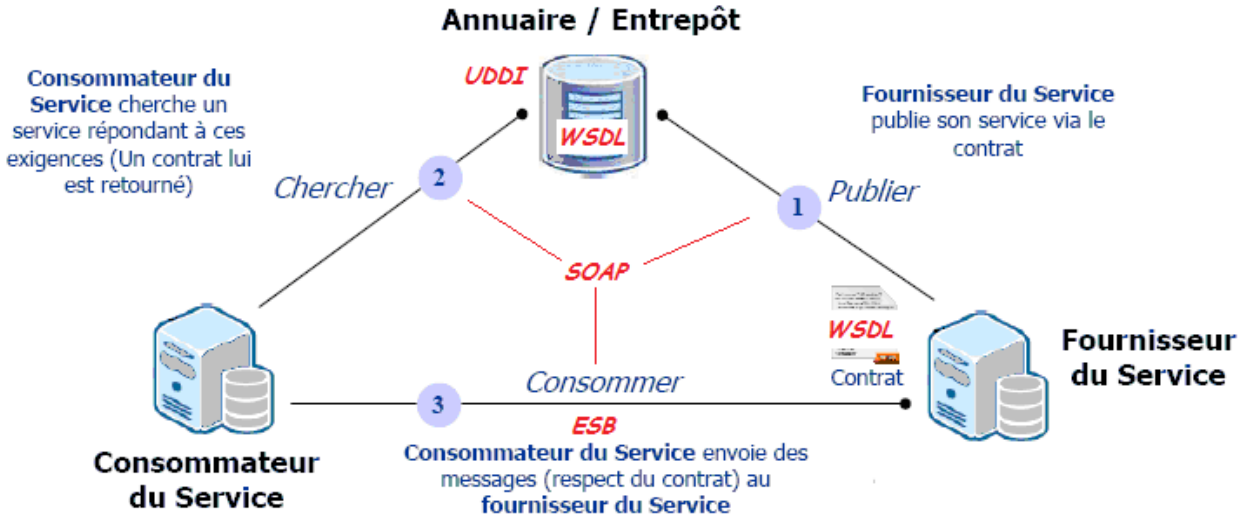
a. **Publier (*Publish*):** le fournisseur publie l'information sur son service.

b. **Chercher (Search):** le client cherche un service approprié.

c. **Consommer (Consume/ Bind):** le client consomme le service.

2) Architecture et éléments de base :

L'architecture SOA se repose sur un ensemble d'acteurs et éléments standardisés [2][3] :



SOA: Architecture de base

- **WSDL:** *Web Service Description Language*.
- **SOAP:** *Simple Object Access Protocol*.
- **UDDI:** *Universal Description Discovery and Integration*.
- **ESB :** *Enterprise Service Bus*

Les acteurs :

a) **Le fournisseur de service (*Service Provider*) :** est chargé de :

1) Définir le service: implémentation, description, contrat d'utilisation.
2) Publier sa description dans l'annuaire
3) Réaliser les opérations

b) **L'annuaire (Entrepôt) (Discovery Agency) :** est chargé de :

1) Recevoir et enregistrer les descriptions de services publiées par les fournisseurs
2) Recevoir et répondre aux recherches de services lancées par les clients

c) **Le client (Service Requestor) :**

1) Lancer la recherche sur un service approprié.
2) Obtenir la description du service grâce à l'annuaire.
3) Utiliser le service.

Les Eléments de base standardisés :

Au sein de l'architecture orientée services, on distingue les notions **d'annuaire**, de **bus**, de **contrat** et de **service**, ce dernier étant le noyau et le point central d'une architecture orientée services [4].

a. **Service:** "Un service est un comportement défini par contrat, qui peut être réalisé et fourni par tout composant pour être utilisé par tout composant sur la base unique du contrat" *[**Bieber and Carpenter 2002**]*.

b. **Annuaire (UDDI) :** liste dynamique de recherche de descriptions de services dans laquelle les fournisseurs de services publient leurs descriptions.

c. **Contrat** (description **WSDL**) **:** contient la description du service et de son utilisation, échangée entre le fournisseur de service et le client.

d. **Bus (*ESB : Enterprise Service Bus*) :** le moyen par lequel s'effectue la communication entre le client et le fournisseur du service. Son but est avant tout de permettre la communication des applications qui à la base ne sont pas pensées pour fonctionner ensemble.

e. **SOAP (Simple Object Access Protocol) :** Protocole de communication a sein de la SOA.

Remarque:

L'implémentation de la SOA qui se repose entièrement sur Internet est appelée la **WOA** (***Web Oriented Architecture***).

IV. Comparaison avec d'autres technologies :

1) SOA vs programmation distribuée et POO [5] :

a) Caractéristiques de la programmation orientée objets :

La programmation orientée objets est basée sur le principe d'encapsulation d'un ensemble d'informations dans un objet doté d'un ensemble de méthodes permettant la manipulation de ses informations. Ce style de programmation à un grand avantage surtout du côté séparation du code ce qui facilite largement les taches suivantes du développement du logiciel. Néanmoins ceci ne nous empêche pas de signaler quelques limites et points faibles de cette architecture :

1. Structure et architecture de l'application peu visible.
2. Interactions entre objets enfouies dans le code.
3. Évolution / modification difficile.
4. Gestion de la consistance d'un changement délicate
5. **Granularité encore trop fine**: ce qui la rend mal adaptée à la programmation à grande échelle.
6. **Couplage fort**: ce qui rend difficile la réutilisation et accroît la complexité des Systèmes OO.

b) Caractéristiques des applications distribuées (orientées composants) :

Une application distribuée est formée d'un ensemble de composants logiciels qui :

1) Collaborent pour l'exécution de tâches communes
2) Distants géographiquement
3) Interconnectés via un réseau de communication
4) Hétérogènes
5) Solutions qui ont fait leur preuve: DCOM, CORBA, EJB, RMI, .Net Remoting, …

Les faiblesses de ces solutions peuvent être résumées dans :

1) Format de représentation de données spécifiques
2) Interopérabilité uniquement si les composants utilisent la même solution
3) Parfois elle utilise des protocoles de transport spécifiques ce qui nécessite une configuration spécifique du réseau.
4) Couplage fort : phénomène du plat de spaghetti.

c) Architecture orientée service:

1) Elle partage l'aspect modulaire (ensemble de fonctionnalités qui font sens) avec la programmation orientée objets.
2) Elle partage les caractéristiques suivantes d'un composant (application distribuée) :
   - Boite noire (séparation interface/implémentation)
   - Indépendant de la localisation
   - Neutralité vis-à-vis des protocoles de transport
3) Correspond à un périmètre fonctionnel que l'on souhaite exposer à des consommateurs.
4) Services faiblement couplés (indépendant des autres services)
5) Expose un petit nombre d'opérations offrant un traitement de bout en bout
6) Utilisation de standards.
7) Pas de remise en cause de l'existant lors d'évolutions technologiques
8) Découplage entre fournisseur et consommateur de services.

2) SOA vs technologie du code mobile :

a) Technologie du code mobile :

La pièce maitresse dans la technologie du code mobile est un code (composant logiciel) capable de se déplacer d'un poste à un autre, par sa propre initiative ou à la demande d'un utilisateur afin de s'exécuter sur un autre poste ou bien pour y continuer son exécution. On distingue ainsi trois catégories principales :

- **Evaluation à distance :** dans ce cas l'utilisateur (le client) envoie son code vers une autre machine pour s'exécuter sur cette dernière et utiliser ses ressources d'une manière locale, à la fin le résultat de l'exécution doit être envoyé à cet utilisateur.
- **Code à la demande :** l'utilisateur ne dispose pas de code, mais il le demande à un serveur de lui envoyer ce code afin de l'exécuter sur sa machine et bien sûr obtenir un résultat.
- **Agent mobile :** un agent mobile est un composant logiciel capable de commencer l'exécution sur un poste, de l'interrompre à un moment donné, de se déplacer vers un autre poste pour y poursuivre son exécution et enfin retourner vers son propriétaire avec son code et son résultat. On parle ainsi de la notion de migration parce que l'agent se déplace avec ses données et son état d'exécution (compteur ordinal, état de la pile et états des registres du processeur).

Ainsi les points attractant remarquant de cette technologie c'est que:

- Elle tourne autour d'un code (composant logiciel ou un service).
- Ce code est mobile (capable de se déplacer et de s'exécuter ailleurs).
- Ce code doit retourner un résultat à l'utilisateur.

b) Architecture orientée service:

L'architecture SOA est une architecture client/serveur, dont sa pièce maitresse est le service qui est un composant logiciel normalisé et adapté à cette architecture. Cette architecture est caractérisée par :

- Le client ne dispose pas de service.
- L'exécution du service est faite à la demande d'un client (utilisateur).

- L'exécution est faite chez le serveur (localement).
- Le résultat est renvoyé au client.

V. Objectifs :

- Doter le SI par des moyens et caractéristiques nécessaires pour s'adapter rapidement aux changements et évolutions technologiques (C'est l'activité qui doit piloter la technologie et non l'inverse).
- Répondre au décalage entre les besoins métiers et leurs réalisations (constituants informatiques)
- Ouverture du SI sur les différents partenaires et matériels ("Libérer le logiciel du PC")
- Minimiser les coûts du développement (pas de remise en cause de l'existant).

VI. Avantages et inconvénients [6] :

1) Avantages :

Au-delà de tous les avantages d'une architecture client/serveur qu'en dispose aussi une architecture SOA, cette dernière est caractérisée aussi par :

1. Accès à distance via un navigateur Web.
2. Pas de logiciel sur le poste client (client léger)
3. Possibilité d'interagir par n'importe quel appareil connecté à Internet.
4. Compatibilité avec différents systèmes d'exploitation et différentes plates-formes (utilisation des standards).
5. Système d'information flexible (anticiper les évolutions technologiques), (il suffit de faire évoluer un service ou d'ajouter un nouveau service sans avoir de perturbations pour les autres services).
6. Réutilisation de code.
7. Une plus grande tolérance aux pannes (maintenance facile).

2) Inconvénients :

1. Recentralisation des systèmes (charge très importante sur le serveur).
2. Coûts de conception et de développement initiaux importants à la fois financier et humain (formation d'une équipe d'experts pour la conception et des équipes pour l'implémentation et l'administration).
3. Lenteur d'exécution (services asynchrones).
4. Manque de maturité de standards (course à la spécification entre W3C et OASIS).

Références :

[1] James P., H. Howell-Barber: « Service-Oriented Architecture – SOA Strategy, Methodology, and Technology» Auerbach Publication 2008.

[2] Le site comment ça marche : « https://www.commentcamarche.net ».

[3] Thomas Erl «SOA Principles of Service Design» 2007 ISBN: 0132344823,9780132344821.

[4] Eric Newcomer, Greg Lomow « Understanding SOA with Web services » . Addison-Wesley Professional (2004).

[5] Occello Audrey et F. Baude « architecture orientée services » SAR O2/SAR 2008.

[6] ZDNet « https://www.zdnet.fr/actualites/soa-comprendre-l-approche-orientee-service-39206712.htm »