Glossaire

Architecture Orientée Services (SOA)

L'**architecture orientée services** (calque de l'anglais **Service Oriented Architecture**, SOA) est une forme d'architecture de médiation qui est un modèle d'interaction applicative qui met en œuvre des **services** (composantslogiciels) :

- Avec une forte cohérence interne (par l'utilisation d'un format d'échange pivot, le plus souvent XML)
- Des couplages externes « lâches » (par l'utilisation d'une couche d'interface interopérable, le plus souvent unservice web WS-*).

Ce terme est apparu au cours de la période 2000-2001 et concernait à l'origine essentiellement les problématiques d'interopérabilité syntaxique en relation avec les technologies d'informatique utilisées en entreprise. Cette conception a évolué pour désigner maintenant le sous-ensemble particulier d'architecture de médiation en fonction de la technologie disponible.

Dans la vie de tous les jours, un fournisseur offre un service à un client le consommant dans une relation de confiance établie entre les deux parties. En général, le client s'intéresse uniquement au résultat produit du service sans avoir le besoin ni le souci de savoir comment ce dernier est obtenu. La SOA suit ce même principe. Le service est une action exécutée par un « fournisseur » (ou « producteur ») à l'attention d'un « client » (ou « consommateur »), cependant l'interaction entre consommateur et producteur est faite par le biais d'un médiateur (qui peut être un bus) responsable de la mise en relation des composants. Le service étant à grandes mailles, il englobe et propose les fonctionnalités des composants du système. Ces systèmes peuvent aussi être définis comme des couches applicatives. L'architecture orientée services est une réponse très efficace aux problématiques que rencontrent les entreprises en termes de réutilisabilité, d'interopérabilité et de réduction de couplage entre les différents systèmes qui implémentent leurs systèmes d'information. Les **SOA** ou **AOS** ont été popularisées avec l'apparition de standards comme les Services Web dans l'e-commerce (commerce électronique) (B2B, inter-entreprise, ou B2C, d'entreprise à consommateur), fondés sur des plates-formes comme Java EE ou .NET. Elles mettent en application une partie des principes d'urbanisation. 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. La déclinaison ou plus précisément la mise en œuvre de la SOA qui repose entièrement sur Internet est appelée la **WOA** (Web Oriented Architecture).

Architecture de médiation

L'**architecture de médiation** est une forme d'architecture en flot de données distribuée qui est souvent nommée architecture orientée services dans le monde commercial (basée sur les services Web WS-* et WS-I) et est, dans sa forme originale, destinée à la synthèse dynamique d'informations utilisées en informatique décisionnelle ainsi qu'à une intégration d'applications d'entreprise intelligente. L'Architecture ARPA I3 est le modèle le plus complet pour ce type d'architecture.

Cette architecture fut conçue en 1992 par Gio Wiederhold dans l'article fondateur «*Mediator in the architecture of future information systems*». Wiederhold y développe une nouvelle vision de l'architecture du traitement de l'information en entreprise, il tente de régler la problématique de l'accès et de l'intégration de l'information en introduisant la notion de médiateur :

«A mediator is a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications.»

Les médiateurs doivent en général se retrouver sur des nœuds différents de ceux qui contiennent les données (BD) et des postes de travail où sont utilisées en dernière instance les données (par les utilisateurs). Il existe plusieurs sortes de médiateurs accomplissant des tâches distinctes, voici les principales :

1. Transformation et filtrage des BD.
2. Accès et intégration de plusieurs BD.
3. Traitement des données supportant un haut niveau d'abstraction et de généralisation (traitement intelligent de l'information).
4. Répertoires intelligents des bases d'information comme les catalogues, les aides à l'indexation, les structures de thesaurus (ontologies).
5. Gestion de l'incertitude reliée aux données absentes ou incomplètes et aux données mal comprises.

Wiederhold distingue par contre clairement les médiateurs de l'ancêtre du concept d'agent, l'acteur et cela de deux façons. Premièrement, les médiateurs sont organisés de manière hiérarchique pour accomplir leurs tâches et deuxièmement, aucune communication intelligente entre les médiateurs n'est nécessaire pour réaliser leurs tâches. Avec l'invention du concept d'agents, nous pourrions dire qu'il s'agit d'agents logiciels intelligents, non communicatifs et statiques.

Wiederhold oppose également dans son article sa vision à celle d'IBM et du « datawarehouse » que cette compagnie promulguait à l'époque et discute de la nécessité de développer un langage d'interface (entre médiateurs et applications) possédant une grande puissance d'expression sémantique.

Service Discovery ou Service Location Protocol

Le **Service Location Protocol** est un protocole de découverte de services qui permet aux ordinateurs et autres dispositifs de trouver des services dans un réseau local sans aucune configuration préalable.

Principe

Chaque service doit avoir une URL qui est utilisée pour localiser le service. L'URL d'une imprimante pourrait ressembler:

service:imprimante:lpr://myprinter/myQueue

Cette URL décrit une file appelée *myQueue* sur une imprimante avec le nom d'hôte *myprinter*. Le protocole utilisé par l'imprimante est LPR. Notez qu'un régime spécial d'URL service est utilisé par l'imprimante. Les URLs "service:" ne sont pas nécessaires: tout régime d'URL peut être utilisé, mais ils vous permettent de rechercher tous les services du même type (par exemple, toutes les imprimantes) quel que soit le protocole qu'ils utilisent. Les trois premières composantes du *service:* URL de type (**service: imprimante:lpr**) sont également appelés type de service. Les deux premiers volets (service:imprimante) sont appelés type de service abstrait. SLP a trois rôles différents pour les périphériques. Un appareil peut également comporter deux ou les trois rôles à la fois :

- Les **User Agent** (UA) sont des dispositifs qui recherchent des services,
- Les **Service Agent** (SA) sont des dispositifs qui annoncent un ou plusieurs services,
- Les **Agents Directory** (AD) sont des dispositifs de cache.

L'existence d'un *Agent Directory* dans un réseau est facultative, mais si un *Agent Directory* est présent, les *User Agent* et les *Service Agent* sont tenus de l'utiliser au lieu de communiquer directement. **SLP** est un protocole orienté paquets. La plupart des paquets sont transmis en utilisant UDP, mais TCP peut aussi être utilisé pour la transmission de paquets de plus de temps. En raison du manque de fiabilité potentiels de l'UDP, **SLP** répète tous les multidiffusions plusieurs reprises dans des intervalles de plus en plus jusqu'à ce qu'une réponse a été reçue. Tous les dispositifs sont nécessaires à l'écoute sur le port 427 pour les paquets UDP, SAS et DAS devrait aussi écouter les connexions TCP sur le même port. Quand un client rejoint d'abord un réseau, il fait une requête en multidiffusions pour l' *Agents Directory* sur le réseau. Si aucune des réponses ne vient d'un *Agents Directory*, elle considérera que c'est dans un réseau sans *Agents Directory*.

CAS AVEC *Agents Directory*

Quand une SA découvre un *Agents Directory*, il est nécessaire d'inscrire tous les services à la *Agents Directory*. Quand un service disparaît, les SA doivent en informer le *Agents Directory* et s'y désinscrire. Un *User Agent* va envoyer le paquet de requête à la *Agents Directory* en utilisant soit UDP ou TCP. Comme chaque SA doit inscrire tous les services avec le *Agents Directory*, le *Agents Directory* est en mesure de répondre à la demande complètement et envoie simplement le résultat à l'*User Agent*.

CAS SANS *Agents Directory*

Afin d'envoyer une requête dans un réseau sans *Agents Directory*, l'*User Agent* envoie un paquet UDP multi-caste qui contient la requête.

Examples

There are many service discovery protocols, including:

- Bluetooth Service Discovery Protocol (SDP)
- DNS Service Discovery (DNS-SD), a component of Zero Configuration NetworkingDynamic
- Host Configuration Protocol (DHCP)
- Internet Storage Name Service (iSNS)
- Jini for Java objects.
- Service Location Protocol (SLP)
- Session Announcement Protocol (SAP) used to discover RTP sessions
- Simple Service Discovery Protocol (SSDP) a component of Universal Plug and Play (UPnP)
- Universal Description Discovery and Integration (UDDI) for web servicesWeb Proxy Autodiscovery Protocol (WPAD)
- WS-Discovery (Web Services Dynamic Discovery)
- XMPP Service Discovery (XEP-0030)
- XRDS (eXtensible Resource Descriptor Sequence) used by XRI, OpenID, OAuth, etc.

Cluster

- En réseau et système, un **cluster** est une grappe de serveurs (ou « ferme de calcul ») constituée de deux serveurs au minimum (appelés aussi nœuds) et partageant la plupart du temps une ou plusieurs baies de disques.
- Un **cluster** est le terme anglais pour désigner un bloc (disque dur), la plus petite unité de stockage d'un système de fichiers (utilisé sur une partition d'un disque dur) d'un système informatique.
- En calcul distribué, le **cluster** est un système informatique composé d'unités de calcul (micro-processeurs, cœurs, unités centrales) autonomes qui sont reliées entre elles à l'aide d'un réseau de communication.
- **MySQL Cluster** est une base de données distribuée destinée aux grappes de serveurs.
- Un **cluster Beowulf** est une grille de calcul.

Runtime ou Environnement d'exécution

Un **environnement d'exécution** ou **runtime** est un logiciel responsable de l'exécution des programmes informatiques écrits dans un langage de programmation donné. Un runtime offre des services d'exécution de programmes tels que les entrées-sorties, l'arrêt des processus, l'utilisation des services du système d'exploitation, le traitement des erreurs de calcul, la génération d'événements, l'utilisation de services offerts dans un autre langage de programmation, le débogage, le profilage et le ramasse-miette.

Un runtime peut être vu comme une machine virtuelle : de la même manière qu'un code natif est exécuté par le processeur, un code objet est exécuté par le runtime. Le runtime sert alors à exécuter du code objet en mettant le code natif *ad hoc* à disposition du processeur pour exécution.

La définition est informelle, par exemple le livre *IWarp anatomy* définit un runtime comme suit : « nous définissons le runtime comme étant le composant logiciel responsable de l'exécution des programmes sur les systèmes iWarp, il facilite l'exécution des programmes en cachant les détails bassement matériels de la machine. ».

Technique

Un runtime met en œuvre un langage de programmation en permettant l'exécution des programmes écrits dans ce langage. Il offre des services tels que les entrées-sorties, l'arrêt des processus, l'utilisation des services du système d'exploitation, le traitement des erreurs de calcul, la génération d'événements, l'utilisation de services offerts dans un autre langage de programmation, le débogage, le profilage et le ramasse-miettes.

Un des premiers langages de programmation fonctionnant avec un runtime a été PL/1 en 1972. Lors de la compilation du programme, les instructions simples en langage PL/1 étaient traduites en les instructions correspondantes en langage machine, tandis que les instructions complexes étaient traduites en des utilisations des fonctions du runtime. Cette construction permettait de diminuer la taille du programme en langage machine.

Langages interprétés

Le runtime est l'interpréteur : il interprète le code source, manipule les variables, réserve de la mémoire et prend en charge les erreurs d'exécution.

Langages compilés

Les limites du runtime sont moins nettes : le code est exécuté directement par le processeur ; cependant ce code peut parfois avoir besoin de certaines fonctionnalités mises à disposition par le runtime, par exemple la création d'objets, le contrôle des types et le ramasse-miettes des langages de programmation orientés objet.

Exemples

Traditionnellement différents langages de programmation utilisent différents runtimes, par exemple la bibliothèque standard de C, l'environnement d'exécution Java, et le Common Language Runtime. Le runtime peut être accompagné d'une *bibliothèque standard* à disposition du programmeur. Le runtime met en œuvre les fonctionnalités élémentaires du langage, tandis que la bibliothèque standard est typiquement écrite dans le langage lui-même.

La bibliothèque standard de C est un runtime normalisé par ANSI et largement inspiré du système d'exploitation Unix. Avant la normalisation chaque compilateur du langage C offrait sa propre bibliothèque de fonctions, et les programmes écrits pour un compilateur donné ne pouvait pas être compilé avec un autre compilateur, bien qu'il soit dans le même langage de programmation.

Le Common Language Runtime contrôle l'exécution des programmes dans différents langages de programmation et fournit des services. Les services sont exposés de manière différente selon le langage de programmation concerné, cependant la palette de fonctions est la même dans tous les langages et ils sont mis en œuvre pas le même logiciel: le Common Language Runtime accompagné d'une bibliothèques de classes de plus de 2500 modules.

Dans les langages de programmation Lisp, Smalltalk, ML et Prolog, l'essentiel des fonctionnalités du langage sont offertes par le runtime accompagné d'une bibliothèque standard à disposition du programmeur.

Apache Maven

Apache Maven est un outil pour la gestion et l'automatisation de production des projets logiciels Java en général et Java EE en particulier. L'objectif recherché est comparable au système Makesous Unix : produire un logiciel à partir de ses sources, en optimisant les tâches réalisées à cette fin et en garantissant le bon ordre de fabrication.

Il est semblable à l'outil Ant, mais fournit des moyens de configuration plus simples, eux aussi basés sur le format XML. *Maven* est géré par l'organisation *Apache Software Foundation*. Précédemment *Maven* était une branche de l'organisation *Jakarta Project*.

Maven utilise un paradigme connu sous le nom de *Project Object Model* (POM) afin de décrire un projet logiciel, ses dépendances avec des modules externes et l'ordre à suivre pour sa production. Il est livré avec un grand nombre de tâches pré-définies, comme la compilation de code Java ou encore sa modularisation.

Un élément clé et relativement spécifique de *Maven* est son aptitude à fonctionner en réseau. Une des motivations historiques de cet outil est de fournir un moyen de synchroniser des projets indépendants : publication standardisée d'information, distribution automatique de modules jar. Ainsi en version de base, *Maven* peut dynamiquement télécharger du matériel sur des *dépôts* logiciels connus. Il propose ainsi la synchronisation transparente de modules nécessaires.

Maven1 et Maven2 ont été développés en parallèle mais les versions futures seront basées sur la structure de la deuxième version. Les parties suivantes de l'article traitent en priorité Maven2.

Langage dédié (DSL)

Un **langage dédié** (*Domain specific language*) est un langage de programmation dont les spécifications sont conçues pour répondre aux contraintes d'un domaine d'application précis. Il s'oppose conceptuellement aux langages de programmation classiques (ou généralistes) comme le Java ou le C, qui tendent à traiter un ensemble de domaines. Néanmoins, aucun consensus ne définit précisément ce qu'est un langage dédié. Ce manque de définition précise sur la nature d'un langage dédié rend délicate la tâche d'établir un historique clair sur l'origine du concept.

En informatique, les langages dédiés traitent divers domaines informatiques (pilotes informatiques, le calcul scientifique, les bases de données[Note 1]) ou traite de différents autres domaines où intervient l'informatique (médecine, aéronautique[Note 1]). L'utilisation de langage dédié n'est pas propre à l'informatique, il existe des langages dédiés aux domaines de la médecine, de la cuisine, etc[Note 2].

La construction des langages dédiés diffère fondamentalement de celle d'un langage classique. Le processus de développement peut s'avérer très complexe. Sa conception nécessite une double compétence sur le domaine à traiter et en développement informatique. Qu'il soit langage dédié interne ou externe, la mise en œuvre de ce type de langage requiert l'utilisation de patron de conception. Il existe des patrons de conceptions décrivant l'implémentation comme le décrit Spinellis

et décrivant plutôt les phases de développement du langage. Ces deux approches, bien qu'étant fondées sur des idées différentes, se complètent l'une et l'autre.

L'utilisation de langages dédiés présente des avantages et des inconvénients par rapport à l'utilisation de langages généralistes. La justification de l'utilisation de langages dédiés peut se faire en fonction de considérations d'ordre technologique, ou d'ordre financier. D'autre part, les avantages et les inconvénients ne sont pas les mêmes qu'il s'agisse d'un langage dédié interne ou externe.

Système d'échange

Un **système d'échange informatique** est une infrastructure transverse de communication agnostique inter et intra systèmes applicatifs et mettant à disposition de ces systèmes des primitives d'interaction et de médiation, reposant sur des protocoles et des formats standards et ouverts.

Les Enterprise Service Bus sont utilisés pour mettre en place un système d'échange.

Répartition de charge

En informatique, la **répartition de charge** (en anglais : *load balancing*) est un ensemble de techniques permettant de distribuer une charge de travail entre différents ordinateurs d'un groupe. Ces techniques permettent à la fois de répondre à une charge trop importante d'un service en la répartissant sur plusieurs serveurs, et de réduire l'indisponibilité potentielle de ce service que pourrait provoquer la panne logicielle ou matérielle d'un unique serveur[1,2].

Ces techniques sont par exemple très utilisées dans le domaine des services HTTP où un site à forte audience doit pouvoir gérer des centaines de milliers de requêtes par seconde.

La *répartition de charge* est issue de la recherche dans le domaine des ordinateurs parallèles. L'architecture la plus courante est constituée de plusieurs répartiteurs de charge (genres de routeurs dédiés à cette tâche), un principal, et un ou plusieurs de secours pouvant prendre le relais, et d'une collection d'ordinateurs similaires effectuant les calculs. On peut appeler cet ensemble de serveurs une *ferme de serveurs* (anglais *server farm*) ou de façon plus générique, une grappe de serveurs (anglais *server cluster*). On parle encore de *server pool* (littéralement, « groupe de serveurs »).

PaaS

Un PaaS, Platform as a service, est un service en cloud computing permettant de pouvoir, tester, développer et mettre en ligne des applications à l'usage des serveurs virtuels.

Étant hébergés dans le Cloud, les utilisateurs y accèdent simplement, par leur navigateur web.

Les avantages du PaaS:

- plus besoin de se soucier du système d'exploitation, du middleware, de l'infogérance, du SGBD ou encore de l'hébergement de vos applications.
- tous vos collaborateurs peuvent travailler sur un projet, même s'ils ne se trouvent pas sur un site ou pays commun.
- Le développement devient accessible à tout le monde
- La flexibilité en ne choisissant que les fonctionnalités nécessaires
- Éviter le gaspillage en ressources dont vous n'avez pas besoin

Pour résumé, le PaaS fournit un environnement d'exploitation pour le développement. Car elle fournit l'architecture et l'infrastructure nécessaires pour supporter le développement d'applications. Par conséquent, le PaaS est idéale pour développer de nouvelles applications destinées au web, aux équipements mobiles et aux PC.