Chapitre 5

Vision Informatique Logique
Architectures Applicative et Logicielle

NFE107

Chapitre 5

Vision Informatique Logique Architectures Applicative et Logicielle

5.4 Architecture Logicielle

Définitions

- Architecture Logicielle
  - Elle se consacre à structurer et à concevoir une application à partir de ses spécifications fonctionnelles
  - Elle structure et décompose de façon logique chaque application en couches
  - Elle introduit les notions et concepts de découpage en couches, modules, composants, design patterns et frameworks

- Approche « boîte blanche »
  - Découpage interne du bloc applicatif en couches motifs de conception (Design Patterns)
  - Framework (« cadre de travail ») et services techniques (gestion des transactions, logs, traces, gestion des fichiers de configuration...)

Construire en assemblant

- A l'origine, les matériaux de base de l'architecte logiciel sont les langages de programmation et les librairies associées

- La capitalisation et la réutilisation ayant toujours été des préoccupations majeures de l'industrie informatique, l'architecte dispose aujourd'hui d'une palette bien plus large :
  - La conception des couches se base fortement sur des pratiques éprouvées (design patterns ou motifs de conception) validés par l'industrie et répondant généralement à des problématiques récurrentes
  - En associant motifs de conception et librairies, les frameworks constituent le « cadre de travail »

- La standardisation aidant, les composants deviennent un ingrédient essentiel de cet assemblage. Certains éditeurs proposent, pour des domaines fonctionnels verticaux, de véritables progiciels supportant les standards

Références
- Design Patterns, par Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Addison-Wesley)
- Core J2EE Patterns : http://java.sun.com/blueprints/corej2eepatterns/index.html
- Microsoft .NET Patterns : http://msdn.microsoft.com/architecture/

Une démarche en 4 étapes

- De manière itérative et incrémentale l'architecte va dans la phase d'architecture logicielle:
  1. Définir le modèle d'architecture en couches et en tiers à mettre en œuvre pour chacun des blocs applicatifs
  2. Préconiser des motifs de conception à mettre en œuvre pour les couches
  3. Préconiser les librairies, composants, frameworks et outils à utiliser pour :
     - L'implémentation des couches logicielles (présentation / coordination / services / domaine / persistance, gestion de logs, …)
     - La fabrication de l'application (conception, développement, tests unitaires, intégration, packaging)
     - La mise en production et le suivi (déploiement, configuration, surveillance, suivi de la qualité de service, suivi des erreurs)
  4. Guider les phases de conception et de développement en s'assurant que les concepteurs et les développeurs ont bien compris l'architecture

Guider la conception et le développement

- L'architecte doit :
  - veiller à ce que l'architecture conçue soit bien appréhendée par les équipes de conception et de développement
  - cadrer la démarche de conception et de développement.
- Sa mission consiste
  - A bien documenter son architecture, afin que les équipes de conception et de développement soient guidées respectivement dans la conception et l'implémentation des couches et l'utilisation des motifs de conception, composants et frameworks
  - Proposer le « framework » du projet, constitué de motifs de conception et de librairies, qui va cadrer :
    - Les concepteurs dans leur conception du fonctionnel et
    - les développeurs dans leur implémentation du fonctionnel.
  - Une couche d'abstraction permettant de rendre l'implémentation indépendante des librairies ou briques choisies trouvera tout naturellement sa place dans un tel socle (notion de méta-framework).
- L'architecte doit préconiser les outils qui seront utilisés :
  - par les concepteurs pour spécifier (conception UML, intranet documentaire…)
  - par les développeurs lors du développement et des tests (environnement de développement, gestionnaires de sources, tests unitaires, automatisation des déploiements…).
- La productivité est une préoccupation grandissante dans les projets. L'architecte, pourra préconiser dans son architecture de développement les outils de productivité (générateurs, L4G personnalisés, …) compatibles avec les contraintes du projet (budgets, culture des personnes).
- L'architecte peut imposer des pratiques (tests unitaires, documentation automatique) qui permettent de mieux assurer la qualité de la réalisation.
- L'architecte **participe à décrire dans le détail le processus de développement de bout en bout** (de la conception à l'intégration : guides de développement, patterns, méthodologie d'intégration continue…) en contraignant l'utilisation des outils, des librairies, composants et frameworks dans le contexte du projet

Framework de développement

- La mise en œuvre d'un framework de développement est un élément fondamental pour la réussite du projet.

- Un framework (cadre de travail) correspond à un ensemble d'outils du marché, de bibliothèques spécifiques et de méthodologies qui visent à faciliter, cadrer et accélérer les développements du projet.

- Le framework de développement, élaboré en phase d'étude et consolidé en phase d'implémentation, est fondamental à la bonne réussite du projet :
  - il définit le cadre de travail et les engagements de chacune des parties, fonctionnelles et techniques, en spécifiant le processus de développement
  - il permet de mieux décorréler les aspects techniques de l'implémentation des fonctionnalités
  - il contraint et standardise les développements
  - il diminue les risques liés au projet

Chapitre 5

Vision Informatique Logique
Architectures Applicative et Logicielle

5.5 Structuration des applications

Structurer

- La structuration du système peut être vue sous différents angles, selon que l'on considère :
  - le découpage « logique » hors de tout contexte d'exécution (machines, OS et réseaux)
  - le découpage « physique » (prenant en compte le contexte d'exécution)

- L'architecte structure le système selon plusieurs « vues » :
  1. Vue en couches (layer view) : vue « logique » montrant le découpage des fonctions de l'application
     - elle est indépendante des considérations physiques
     - la littérature propose des modèles standards de structuration qui couvrent les types classiques d'applications
     - le modèle de référence est le modèle à 5 couches, qui s'applique aux applications munies d'une interface graphique manipulant des données persistantes
  2. Vue en niveaux (tier view) : vue « physique » de la structuration de l'application

La vue en couches

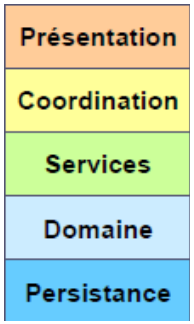
- La structuration des applications en couches permet :
  - de maîtriser la complexité des applications (développement, échanges entre les applications, interactions entre objets)
  - d'optimiser les temps de développement, en factorisant certaines briques applicatives
  - d'isoler les problématiques d'enchaînements de processus en définissant et en délimitant le périmètre du contrôle de l'intégrité transactionnelle (locale et distribuée)
  - de favoriser la communication :
    - à l'intérieur d'une application, en structurant les échanges entre les différentes couches
    - entre les applications en précisant les principes de communication liée aux couches de diverses applications

- Le passage d'une couche vers une autre doit impérativement se faire via des interfaces qui représentent chacune un service d'accès (introduction de la notion de contrats de services)

- La cohérence entre l'urbanisme (qui conçoit les plans des SI dans une perspective de fonctionnement rationnel et d'évolutivité) et l'architecture d'un projet (qui en construit les blocs applicatifs), est d'autant plus aisée à assurer si l'on parvient à découper les blocs applicatifs selon les couches du modèle de l'architecture applicative logique

Modèle en 5 couches

- La structuration des applications se traduit par une décomposition logique de chaque application en 5 couches :
  - Présentation
  - Contrôleur
  - Services
  - Domaine
  - Persistance



- Chaque couche a ses propres responsabilités et utilise la couche située en dessous d'elle

- En fonction du projet, les architectes enrichissent et élaguent le modèle. La structuration est alors guidée par les contraintes exprimées et existantes

Couche présentation

- La couche Présentation gère et assure l'affichage de l'interface graphique utilisateur ou les Interfaces Homme-Machine (IHM : fenêtres, pages, composants graphiques...)

- Cette couche intègre principalement :
  - la gestion du domaine visuel
  - l'interaction avec les utilisateurs
  - l'interception des événements utilisateurs et l'appel à la couche Contrôleur
  - la gestion du multicanal (web, voix, mobile, fax)
  - les services de portail (agrégation d'IHM, bouquets de services)
  - les services d'impression (impressions PDF, gestion de templates…)

- L'interface graphique du système est composée de deux parties :
  1. Les écrans, qui assurent la restitution de l'état du système sur un terminal graphique et permettent la saisie de données et la demande d'exécution de traitements
  2. Les éditions, qui assurent la restitution de l'état du système en vue d'une impression. Un écran de prévisualisation accompagne généralement chaque édition

Couche Présentation (suite)

- On distingue trois catégories d'IHM pour les applications interactives :
  - Client léger :
    - Dans ce modèle, aucun déploiement n'est réalisé sur le poste client à l'exception d'un navigateur Web
    - Les différents écrans de l'application sont générés en temps réel côté serveur et téléchargés par le poste client
  - Client lourd :
    - Dans ce modèle, l'ensemble des écrans de l'application sont stockés ou générés sur le poste client et doivent avoir été déployés sur celui-ci préalablement à l'exécution
    - Ce type de client n'impose à priori pas de restriction sur le contenu et l'ergonomie des écrans
    - En règle générale, une complexité croissante va de pair avec une taille croissante de l'application à télécharger
  - Client riche (Smart Client) :
    - Ce modèle constitue un compromis entre le client léger et le client lourd
    - Il présente une ergonomie comparable à celle d'un client lourd tout en limitant les problématiques de déploiement inhérentes à ce dernier
    - Plusieurs options sont disponibles pour le développement d'IHM riches, à savoir Adobe Flex, Microsoft Silverlight, Google Web Toolkit qui permettent d'exécuter directement le code dans le navigateur

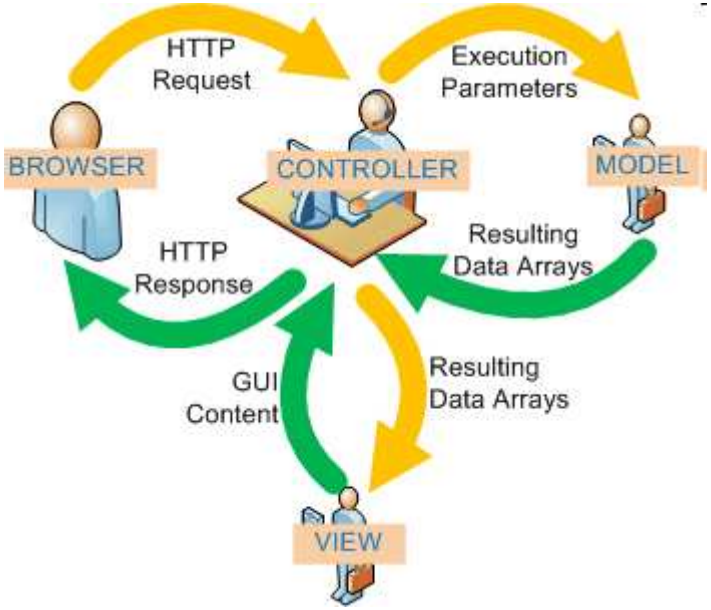
Couche Contrôleur

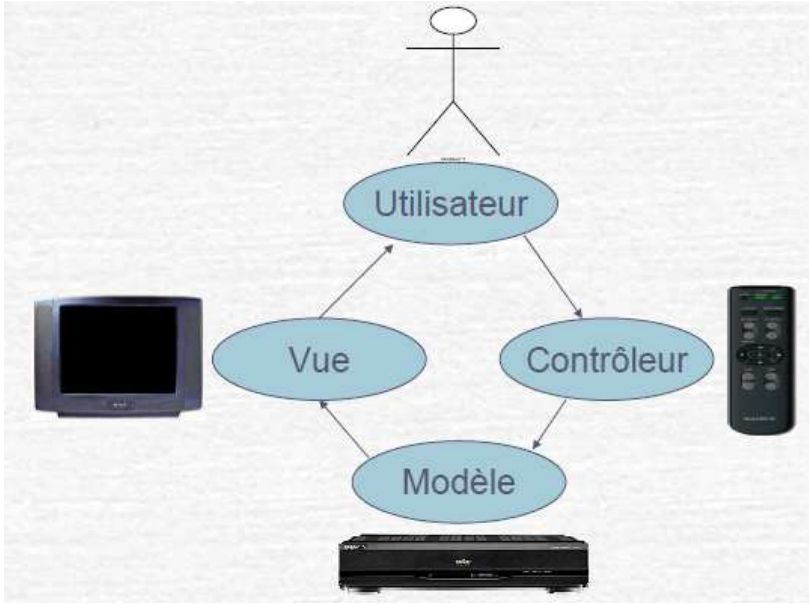
- La couche Contrôleur gère :
  - le contrôle de la cinématique des écrans
  - l'invocation des appels de services
  - les erreurs et les exceptions qui peuvent être levées
  - les sessions / espace de travail utilisateur
  - les habilitations et les droits d'accès

- Dans certains frameworks, la couche Contrôleur peut prendre en compte la langue et le type de terminal de l'utilisateur

- L'architecture applicative de gestion des interactions utilisateur est généralement mise en œuvre autour du motif de conception MVC (Modèle-Vue-Contrôleur):
  - le Modèle représente l'ensemble des composants qui sont chargés de réaliser des appels à la couche Services et de mettre les résultats de l'appel à la disposition de la Vue
  - la Vue représente l'interface utilisateur.
  - le Contrôleur gère la synchronisation entre la Vue et le Modèle. Le contrôleur réagit aux actions de l'utilisateur en effectuant les actions nécessaires sur le Modèle. Le Contrôleur surveille les modifications du modèle et informe la Vue des mises à jour nécessaires.

Le modèle MVC





Couche Services

- La couche Services correspond aux traitements qu'effectue l'application
- Elle représente l'implémentation de la logique des cas d'utilisation (use-case fonctionnels) à proprement parler
- Cette couche doit :
  - implémenter la logique métier
  - gérer la sécurité applicative
  - gérer les transactions étendues (processus, compensation)
  - gérer l'intégrité transactionnelle (transactions locales et distribuées)
  - gérer les appels aux objets métiers de la couche Domaine
- Elle gère les services métiers qui enchaînent des règles métiers (processus métier) et des appels à la couche Domaine
  - Exemple : virement de compte à compte

Couche Domaine

- La couche Domaine gère l'intégrité du modèle « métiers ». Cette couche intègre principalement:
  - la gestion des règles métiers « élémentaires » (sans état, sans processus)
  - la fourniture des moyens d'accès à l'information (SGBDR, Mainframe...)
  - le respect des propriétés transactionnelles de la couche persistance

- La couche Domaine recense les objets métiers manipulées par l'application

- La couche Domaine est concentrée sur le métier de l'entreprise, commun à toutes les applications
  - Elle contient les Objets Métier qui implémentent le modèle métier. Ils offrent à la couche Services une abstraction pour la manipulation unitaire ou multiple des occurrences de données, ainsi que la mise en œuvre des règles de gestion associées
  - Exemple bancaire : l'opération de virement de compte à compte
    - l'opération de virement de compte à compte est un élément de la couche Services
    - le compte bancaire et le client et leurs règles de gestion respectives, se situent dans la couche Domaine

Couche Persistance

- La couche Persistance intègre principalement :
  - la persistance complète du Système d'Informations (données structurées ou non structurées, gérées entre autres via un SGBDR, annuaire LDAP, transaction CICS, ...)
  - La fourniture des services de stockage des données, moteurs relationnels, bases objets, bases XML…
  - la création, la modification, la suppression d'occurrences des objets métiers

- Elle contient un niveau d'abstraction de données les DAO (Data Access Object) qui prennent en charge l'accès à la source de données (SGBDR, fichiers XML, CICS, …). Ils offrent une vision objet des occurrences d'entités du modèle physique de données

- La couche Persistance offre les fonctionnalités de base qui permettent :
  - de créer, rechercher, modifier et supprimer des composants objets métiers dans le respect des propriétés transactionnelles classiques
  - d'utiliser le mécanisme de projection objet vers relationnel (mapping Objet / Relationnel) qui consiste en la transformation de la représentation des données en une représentation objet
  - d'offrir le support, des contextes transactionnels issus de la couche domaine

Couches Transverses

- Couche Sécurité
  - Services de sécurité : SSO, authentification, gestion des habilitations, intégrité, non-répudiation… La sécurité n'est pas une couche isolée, mais transverse aux autres couches:
    - authentification des utilisateurs et contrôle des habilitations au niveau des services IHM, sécurisation des traitements (authentification, habilitations grosse maille et habilitations fines…),
    - sécurisation des échanges, sécurisation des données…

- Couche Services Techniques (Core Services)
  - Indépendamment des fonctionnalités des applications et de leur découpage en couches logicielles, on retrouve des composants et services de base communs (Core Services) et transverses à l'ensemble des couches :
    - gestion des traces
    - statistiques et logs
    - gestion des erreurs
    - gestion des propriétés de configuration
    - gestion des fichiers de messages (internationalisation, messages d'erreurs)
    - monitoring…

Architecture en couches pour une application complexe

- Les architectes peuvent être amenés à effectuer des découpages plus fins lorsque les contraintes deviennent plus industrielles

- Un tel découpage s'explique par :
  - La séparation des traitements dans une couche Service a pour objectif de permettre leur réutilisation entre des processus « automatiques » (arrivée de messages en provenance de systèmes externes) et des opérations manuelles effectuées via les IHMs
  - Une couche Domaine est pertinente dans le cas où les traitements à effectuer sont nombreux, portent sur des entités métiers identifiées, récurrentes et ont une importante durée de vie
  - Le recours à une couche **Echanges** (comprenant les couches **Connectivité**, **Transformation** et **Routage**) permet d'intégrer des sources d'informations multiples et hétérogènes, en les transformant en un ensemble plus réduit de formats pivots pour les router vers les traitements adéquats. Elle propose des services d'échanges entre traitements (échanges synchrones, asynchrones), entre système de persistance (synchronisation de référentiels, ETL, ...), services de garantie de livraison de message, Message Broker (Transformation, Routage, DataFlow), services de gestion de transactions étendues (processus, compensation)

La vue en niveaux

- La vue en niveaux (la tier view) donne une vision plus « physique » de la structuration de l'application. Les niveaux (ou tiers) peuvent être répartis physiquement sur différents composants matériels.

- On identifie un changement de « niveau » dès qu'un module logiciel doit passer par un intermédiaire de communication (middleware) pour en invoquer un autre. Si l'utilisation du middleware est en général transparente pour les développeurs, elle n'est pas sans impact sur l'architecture. L'architecte doit donc maîtriser les caractéristiques (client/serveur, publication/abonnement, sécurité, support du transactionnel, …) et en justifier l'usage.

- Des modèles standards de répartition de niveaux ont été définis dans les projets par l'industrie au fur et à mesure de l'évolution des capacités matérielles et des besoins

Modèle à 1 tiers

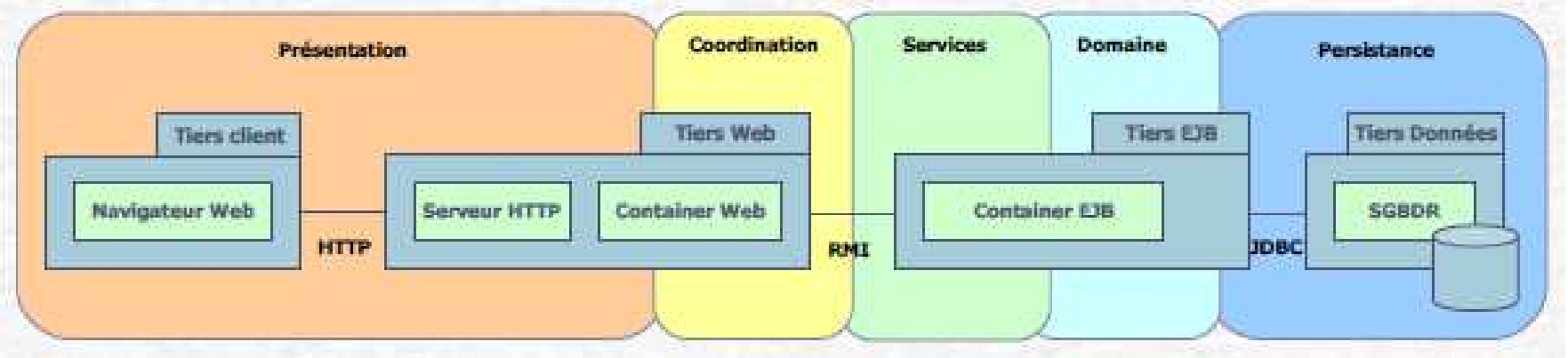
- Le modèle à 1 niveau (ou tiers) correspond à un binaire dans lequel s'exécutent toutes les couches, de la présentation à la persistance.
- C'est l'exemple de l'application utilisée en monoposte ou sur un réseau de serveurs de fichiers, ainsi que de l'application sur système central.
- Les données sont stockées sur un fichier local ou partagées sur un serveur de fichiers

Le modèle à 2 tiers

- Le modèle à 2 niveaux (ou tiers), encore appelé « client/serveur première génération », repose sur l'utilisation de moteurs de bases de données relationnelles. Ces moteurs permettent de distribuer la gestion de la persistance sur un serveur
- Ce modèle a typiquement permis de mieux répondre au besoin d'accès concurrents et de supporter d'importants volumes
- C'est avec ce type d'architectures que l'on a pu gagner en flexibilité et se passer des onéreux systèmes centraux
- L'application d'entreprise peut ainsi être accédée depuis un ordinateur personnel avec des standards de présentation moderne

Modèle n-tiers

- La littérature parle du modèle générique « N-tiers » (ou N-niveaux)
- Le modèle N-tiers est celui mis en œuvre dans le cadre des projets web
- Exemple : tiers impliqués dans le modèle d'architecture J2EE



Chapitre 5

Vision Informatique Logique
Architectures Applicative et Logicielle

5.6 Qualité de service

Qualité de service

- Les points concernant la qualité de service doivent être abordés dès les phases d'architecture applicative logique et logicielle, notamment sur les problématiques suivantes :
  - Sécurité
    - Faire apparaître les blocs applicatifs impactés par les aspects sécurité :
      - identification,
      - authentification,
      - gestion des habilitations,
      - intégrité, confidentialité et non répudiation (données et échanges)
  - Performances
    - Mentionner les points liés aux temps de réponse.
    - En ce qui concerne d'éventuels mécanismes de répartition de charge, faire apparaître les composants progiciels impactés par ces mécanismes, et les composants applicatifs également impactés par ces mécanismes (faire apparaître notamment les éventuels besoin de gestion d'affinité de session)…
  - Continuité de Service
    - Faire apparaître les composants produits impactés par les mécanismes de basculement sur incident (transparent ou non), et les composants applicatifs également impactés par ces mécanismes (faire apparaître l'éventuel besoin de partage/réplication de contexte de session, …
    - Identifier les besoins en termes de Haute disponibilité (mécanismes de loadbalancing, gestion du fail-over)

Chapitre 5

Vision Informatique Logique Architectures Applicative et Logicielle

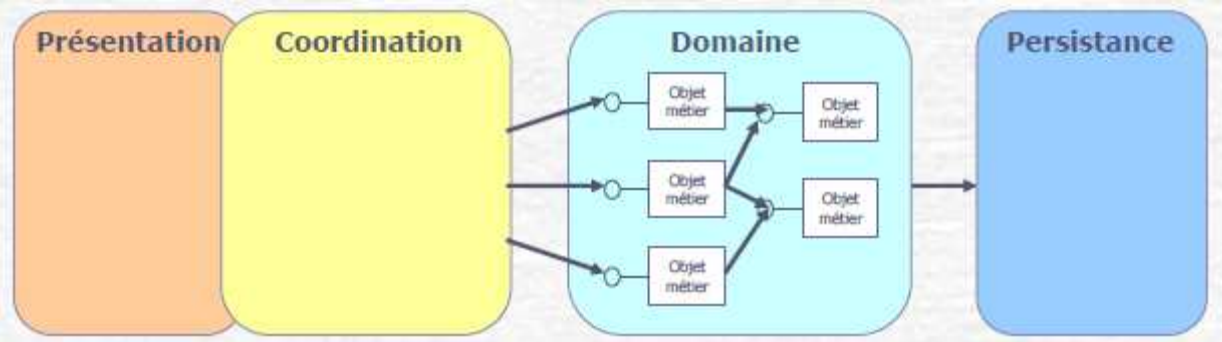
5.7 Typologies d’Architecture

Architecture Orientée Objets

- Dans une architecture orientée manipulation d'objets, on remarque tout de suite le nombre de liens entre la couche Coordination et les objets métiers de la couche Domaine.

- Le code client doit traiter directement avec le modèle objet de la couche Domaine, ce qui a pour conséquence de lier celle-ci très fortement à un modèle spécifique et requiert un nombre d'appels important entre les deux couches.

- La multiplication des appels entre couches pose problème lors de la mise à disposition à distance des objets métiers. De plus le nombre d'objets à manipuler réduit l'indépendance entre couches et complexifie la prise en main de la couche métier



L’Architecture Orientée Services (SOA)

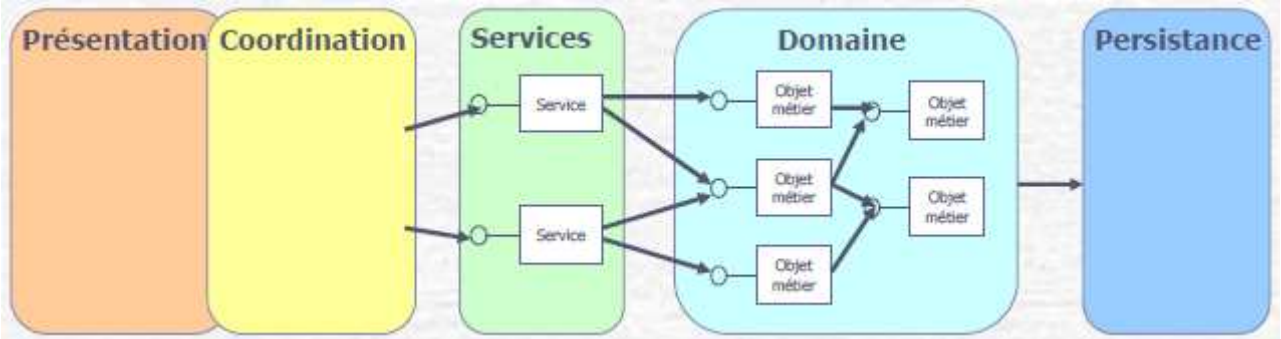
- L’architecture SOA consiste à traiter toute application du système d’information comme un fournisseur de services
- Le principal enjeu de la SOA étant la réutilisation des services, ceux-ci doivent être pensés non seulement en fonction d’un projet immédiat, mais aussi sur le long terme pour servir à d’autres applications
- Les applications d’aujourd’hui ne sont plus monolithiques et doivent s’intégrer harmonieusement dans le système d’information de l’entreprise
- Cela implique l’interaction avec l’existant (systèmes, plates-formes et applications), et une ouverture vers une réutilisation future des nouveaux modules fonctionnels ou techniques
- Les axes majeurs de la SOA sont :
  - La **réutilisation** et la **composition** : partage de modules entre applications
  - La **pérennité** : implique le support des technologies existantes et à venir
  - L'**évolutivité** : la majeure partie des applications sont amenées à évoluer dans le temps afin de pouvoir répondre aux nouveaux besoins fonctionnels
  - L'**ouverture** et l'**interopérabilité** : partager des modules applicatifs entre plates-formes et environnements
  - La **distribution** : pouvoir utiliser ces modules à distance et les centraliser au sein de l'entreprise par exemple
  - La **performance**

SOA (suite)

- Dans une SOA un niveau d'indirection supplémentaire est introduit sous la forme de la couche Services.

- La couche Coordination ne manipule plus directement les objets métiers, mais passe par des appels de services. Les objets métiers se trouvent dans des bibliothèques de classes directement chargées dans le même processus que les services, le coût des appels aux objets métiers est alors très faible.

- Les services agissent comme des « boites noires » faisant abstraction de la complexité du modèle objet, présentant un ensemble de fonctionnalités restreints et permettant de réduire les échanges entre les couches



SOA (suite)

- Un service (local ou distant) est un module :
  - Pouvant être invoqué en mode synchrone ou asynchrone
  - Chargé d'une fonction particulière
  - Présentant une interface bien définie
- Un service doit assurer un rôle bien défini et non redondant (les services doivent être factorisés)
- Un contrat de service précisant le contrat d'interface et la qualité de service doit être défini entre le(s) fournisseur(s) de services et le(s) consommateur(s)
- Un service doit pouvoir être utilisé par plusieurs types de consommateurs :
  - Internes ou
  - Externes (clients, filiales, partenaires, …)
- Un service doit pouvoir être utilisé par exemple pour un traitement batch (asynchrone multi-valué) et pour un traitement TP (synchrone mono-valué)
- Le terme service a un sens plus large que son utilisation dans le cadre de Services Web. On peut faire de la SOA sans Service Web. L'orientation service renvoie à une façon de concevoir le SI avant de renvoyer à une implémentation technique de cette conception

SOA (suite)

- Un service est créé dans le contexte d'une application, mais il doit être conçu de façon à pouvoir être utilisé dans d'autres contextes
- Les services doivent être conçus pour être utilisés aussi bien en local qu'à distance
- Découplage entre les couches et optimisation des échanges afin d'assurer de plus grandes facilités d'évolutivité et de réutilisation, il convient de découpler les couches les unes des autres
  - La communication entre les couches dépend de l'architecture physique de l'application. Lorsque les couches se trouvent sur des machines physiquement distinctes, des mécanismes tels que le remoting ou les Services Web peuvent être mis en œuvre
  - Lorsque les couches d'une application se trouvent toutes sur la même machine, il convient d'optimiser la performance en privilégiant des appels directs entre les couches