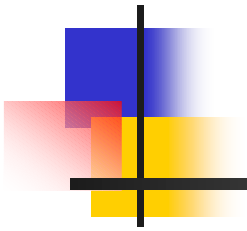


NSY107 - Intégration des systèmes client-serveur

Cours du 17/06/2006, 4 heures,

Thème : Architectures

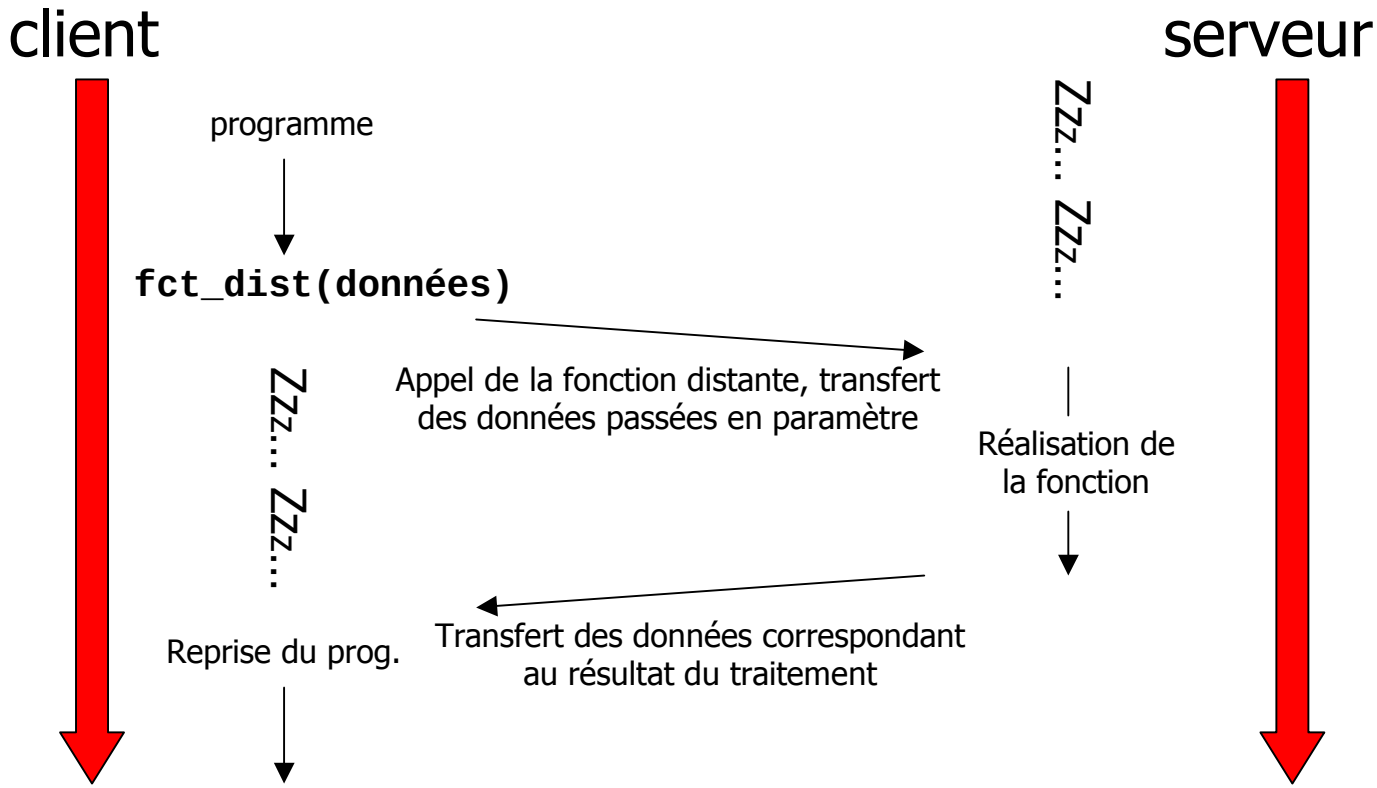
© Emmanuel DESVIGNE
<emmanuel@desvigne.org>

Plan du cours « Architectures »

- RPC : rappels et compléments (SUN RPC, DCE RPC, web services, XML-RPC)
- Client/serveur et programmation objet (CORBA, SOAP, OLE/COM/.NET, JAVA)
- Exemples de services client-serveur célèbres
- Architecture (3-tiers, connecteurs – SAG / ODBC / JDBC –, N-tiers, offre commerciale : JAVA vs .NET)
- « Business Intelligence » et langages L4G

RPC : rappels et compléments [1/9]

- Rappel : RPC = Remote Procedure Call (appel de procédure à distance)



RPC : rappels et compléments [2/9]

- Dans les RPC, la communication client-serveur :
  - peut se faire par **datagramme** (par paquets),
  - ou par **connexion** (flux de données dans un canal).
- Elle peut être :
  - **Synchrone** (Cf. schéma précédent) : le serveur attend la requête du client ; et pendant que le serveur fait le traitement, le client attend ;
  - **Asynchrone** : pendant qu'un des acteurs traite les informations, l'autre acteur, au lieu d'attendre, continue de « vivre sa vie ». Il est interrompu (par une interruption système) quand l'autre acteur lui envoie de nouveau de l'information, afin qu'il aille traiter ce flot entrant.

RPC : rappels et compléments [3/9]

- Nous avons vus les « SUN RPC » (qui utilisent XDR pour la représentation de données, et un système de « *port mapper* » comme annuaire des services proposé par un serveur).
- Il existe d'autres modèles de RPC :
  - Les « DCE RPC » : DCE = Distributed Computing Environment. C'est un modèle de RPC auquel a été ajouté :
    - Un service de sécurité (login et authentification) ;
    - Un service de répertoires des ressources (annuaire) ;
    - Un protocole de gestion d'un temps global synchronisé ;
    - Un système de gestion de fichiers distribués.

  Ex d'implémentation de DCE RPC : MSRPC (Microsoft)

RPC : rappels et compléments [4/9]

- DCE s'appuie sur le concept de cellules : « une cellule = ensemble d'utilisateurs, de machines ou autres systèmes qui ont un but en commun et partagent des services DCE communs ». Pour former des cellules :
  - Par but commun : les personnes travaillant à un même but gagnent à être regroupées dans la même cellule.
  - Par intérêt administratif : il est plus facile d'administrer des utilisateurs si ceux-ci sont regroupés dans une seule cellule.
  - Par souci de sécurité : on préfèrera mettre dans une même cellule les machines d'utilisateurs qui ont le même degré de fiabilité.
  - Par coût d'utilisation : les usagers qui interagissent fortement entre eux seront placés dans une même cellule.

RPC : rappels et compléments [5/9]

- DCE comprends 6 composants :
  - Threads package : la gestion des threads (processus light)
  - Remote Procedure Call facility : la gestion des RPCs (forme ce qui est appelé des ***stubs*** – souches –)
  - Distributed Time Service : la notion de temps global
  - Name services : la gestion des noms, avec :
    - Cell Directory Service,
    - Global Directory Service,
    - Global Directory Agent ;
  - Security Service : la gestion de la sécurité (authentification et autorisations) ;
  - Distributed File Service : le système distribué de gestion de fichiers.

RPC : rappels et compléments [6/9]

- Les web services :
  - ensemble de protocoles et de normes utilisés pour échanger des données entre les applications
  - sont interopérables sur diverses plateformes grâce au respect de normes ouvertes regroupées au sein du terme générique de **SOA** (Service Oriented Architecture, architecture orientée services), définie par l'OSI et le W3C
  - basés sur HTTP (traverse les firewall) et tant que possible, échanges de données sous forme de texte
  - ex : flux RSS (XML), WebDAV (gest. fichiers / http)
  - inconvénients des web services :
    - peu sécurisés (pas de norme *sécurité des transactions*),
    - peu performants.

RPC : rappels et compléments [7/9]

- XML-RPC : protocole RPC qui utilise :
  - le protocole HTTP pour le transport des données,
  - et la norme XML pour le codage des données.
- C'est l'ancêtre de SOAP (Simple Object Access Protocol = protocole de RPC orienté objet bâti sur XML, Cf. prochain chapitre)
- L'appel à une procédure RPC se fait avec du XML, les données transitent sur le réseau en XML, et le résultat obtenu est… du XML.

RPC : rappels et compléments [8/9]

- Exemple de requête XML-RPC :

```
<?xml version='1.0' encoding='UTF-8'?>
<methodCall>
  <methodName>projet.fct_exemple</methodName>
  <params>
    <param><value>
      <struct><member>
        <name>champ1</name>
        <value><string>test</string></value>
      </member><member>
        <name>champ2</name>
        <value><string>val2</string></value>
      </member></struct>
    </value></param>
  </params>
</methodCall>
```

RPC : rappels et compléments [9/9]

- Réponse possible :

```
<?xml version='1.0' encoding='UTF-8' ?>
<methodResponse>
  <params>
    <param>
      <value>
        <string>
          <method> projet.fct_exemple</method>
          <format>xmlrpc</format>
          <foo>bar</foo>
          <api_key>c712879489563</api_key>
        </string>
      </value>
    </param>
  </params>
</methodResponse>
```

Client/serv. & programmation objet [1/12]

- Nous avons vu que le modèle client-serveur permettait :
  - Le transport de données,
  - L'appel de procédure à distance.
- ➔Nous avons les deux éléments qui permettent d'accéder à des « *objets* » à travers un réseau. Rappel : **Objet** =
  - Attributs (données) ;
  - Méthodes (actions applicables à un objet, permettant d'en modifier son état).

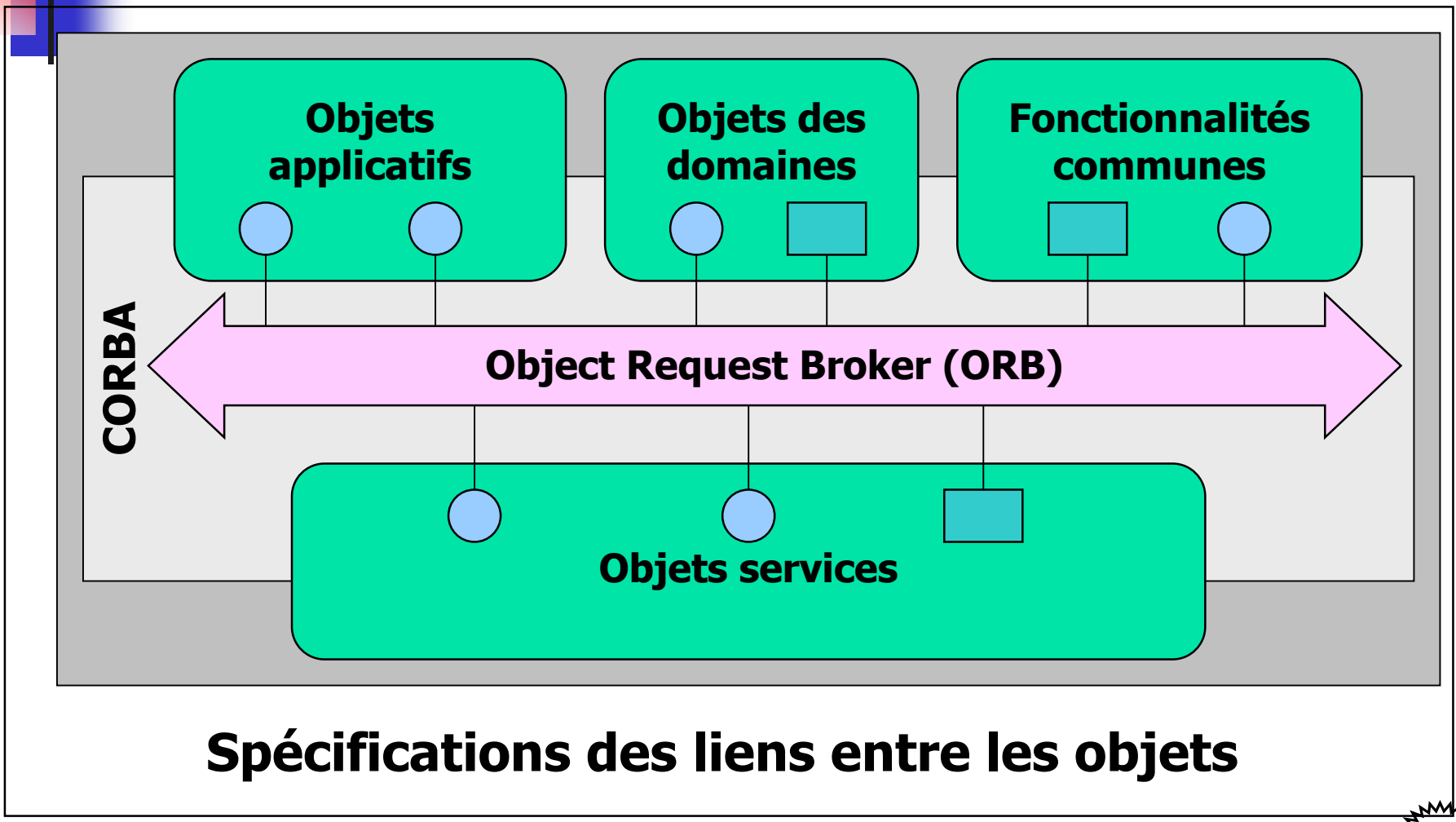
Client/serv. & programmation objet - CORBA [2/12]

- Les deux notions programmation objet et client-serveur ont donné naissance à multitude de modèle de « *distribution d'objets via de réseau* » :
  - CORBA (Common Object Request Broker Architecture) :
    - Créé par l'OMG (consortium international de 800 membres, créé en 1989)
    - CORBA vers. 1.1 : 1992 ; CORBA vers. 2.0 : 1995
    - Utilise la notion d'ORB : Object Request Broker, qui gère les relations client/serveur entre objets

Client/serv. & programmation objet - CORBA [3/12]

- Dans le modèle CORBA, l'ORB gère :
  - La location d'objet,
  - La désignation des objets,
  - L'empaquetage des paramètres (marshalling),
  - Le dépaquetage des paramètres (unmarshalling),
  - L'invocation des méthodes,
  - La gestion des exceptions.
- CORBA définit 4 types d'objets :
  - Les objets applicatifs,
  - Les objets des domaines,
  - Les objets services,
  - Les fonctionnalités communes.

Client/serv. & programmation objet - CORBA [4/12]



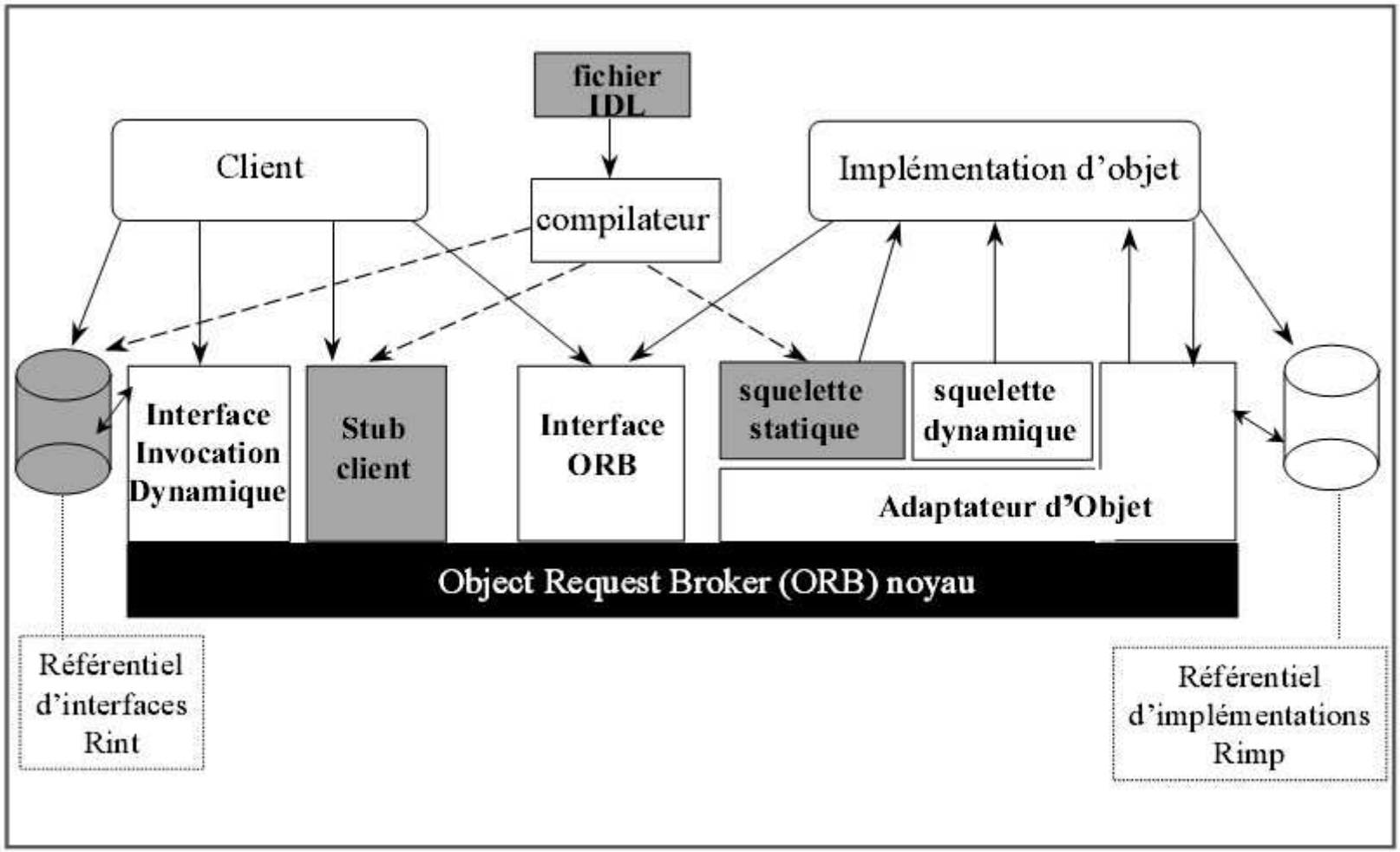
Client/serv. & programmation objet - CORBA [5/12]

- Les objets applicatifs :
  - Spécification des interfaces, à l'aide d'un langage de définition des interfaces (**IDL** = Interface Definition Language, langage permettant l'interaction entre des composants logiciels au sein d'une architecture ou application distribuée ; orienté objet (héritage multiple), permet de définir les types, constantes, exceptions, interfaces, modules…),
  - Définis par l'utilisateur selon les besoins de l'application à programmer (ainsi, par définition, non standardisés par l'OMG)
- Les objets de domaines :
  - Idem aux objets applicatifs, mais standardisés selon les domaines d'application (médical, financier, etc.) ;
  - Peuvent être étendus ou spécialisés par l'héritage

Client/serv. & programmation objet - CORBA [6/12]

- Les fonctionnalités communes :
  - ensemble de services de haut niveau fournissant des fonctionnalités utiles dans de nombreuses applications
  - elles sont indépendantes des domaines d'application
  - exemple : impression, envoi de messages d'erreur…
- Les objets services (CORBA services) :
  - permet d'étendre les fonctions de l'ORB
- Le modèle CORBA est souvent représenté par le schéma de la diapositive suivante *(source : Virginie AMAR, CSTB)*

Client/serv. & programmation objet - CORBA [7/12]



Architecture générale de CORBA

Client/serv. & programmation objet - CORBA [8/12]

- Il est à noter qu'en modélisant les différents objets et leurs interfaces avec le langage IDL, un compilateur IDL génère automatiquement :
  - Coté client (stub – souche –) :
    - Code utilisé par le client pour les invocations statiques,
    - Le lien entre le client et l'ORB,
    - Les opérations d'empaquetage et de dépaquetage des paramètres (marshalling et unmarshalling) ;
  - Coté implémentation de l'objet :
    - Code utilisé par l'adaptateur d'objets pour les invocations statiques,
    - Lien entre l'ORB et l'objet d'implémentation,
    - marshalling et unmarshalling.

Client/serv. & programmation objet - SOAP [9/12]

- Nous avons vu qu'il existait une « extension » de XML-RPC à la programmation orientée objet : **SOAP** (Simple Object Access Protocol)
- Défini par le W3C :
  - http://www.w3.org/2002/07/soap-translation/soap12-part0.html
  - http://www.w3.org/2002/07/soap-translation/soap12-part1.html
  - http://www.w3.org/2002/07/soap-translation/soap12-part2.html

Client/serv. & programmation objet - SOAP [10/12]

- SOAP (est une architecture SOA) :
  - permet la transmission de messages entre objets distants (invocation des méthodes d'objets physiquement situés sur une autre machine),
  - le transfert se fait le plus souvent à l'aide du protocole HTTP, mais peut également se faire par un autre protocole, comme SMTP,
  - Le protocole SOAP est composé de 2 parties :
    - une enveloppe, contenant des informations sur le message lui-même afin de permettre son acheminement et son traitement,
    - un modèle de données, définissant le format du message, c'est-à-dire les informations à transmettre.

Client/serv. & programmation objet – OLE/COM/.NET [11/12]

- Microsoft a proposé sa propre vision d'objets répartis sur plusieurs machines :
  - 1991 : OLE 1.0 (Object Linking and Embedding). Utilisé par exemple dans Word/Excel pour le copier/coller ou la sauvegarde d'objets dans un fichier
  - 1993 : OLE 2 (appelé par la suite simplement OLE), et définition de COM (Component Object Model), plus connu sous le nom de « ActiveX »
  - A terme, remplacé par « .NET » : concept « programmation orientée objets + client-serveur + machine virtuelle avec objets standards – framework –, et accepte divers langages et compilateurs vers le pseudo-code interprété par ce framework (Cf § architecture).

Client/serv. & programmation objet – JAVA [12/12]

- Le langage JAVA propose aussi des outils permettant de faire de « l'objet réparti » :
  - RMI (Remote method invocation) : API qui permet d'appeler des objets distants
  - L'utilisation de cette API nécessite l'emploi d'un registre RMI sur la machine distante hébergeant les objets que l'on désire appeler, au niveau duquel ils ont été enregistrés (annuaire de service, équivalent au portmapper des RPC)
  - Si JAVA propose RMI comme mécanisme d'objets répartis, il propose aussi un mécanisme « d'applications réparties »: EJB (Enterprise JavaBeans). Le dialogue entre le client et un applicatif EJB se fait via cette API EJB (Cf § architecture).

Exemples de services client-serveur célèbres [1/8]

- Pour la consultation de documents :
  - Historiquement, « Wais », « gopher »
  - Le plus célèbre : le Web (protocole HTTP, documents HTML), et récemment pour les portables GSM : Wap (protocole WML) :
    - Client = navigateur (FireFox, Internet Explorer, Opera, etc.)
    - Serveur = Apache, IIS (Microsoft), etc.

Exemples de services client-serveur célèbres [2/8]

- Pour les annuaires :
  - Historiquement : X.500 (annuaire défini par le CCITT, ex ISO),
  - Plus récent : LDAP (Lightweight Directory Access Protocol) :
    - Clients très variés (logiciels d'e-mails, progiciels, serveurs de téléphonie, etc.),
    - Serveur libres (OpenLDAP) ou propriétaires (inclus dans Microsoft ActiveDirectory, etc.)

Exemples de services client-serveur célèbres [3/8]

- Pour l'authentification/sécurité :
  - Serveur RADIUS (Remote Authentication Dial-In User Service), normalisé par l'IETF
  - Utilise le protocole AAA (Authentication Authorization Accounting), et EAP (Extensible Authentication Protocol) qui permet l'extention des fonctions de base
  - Clients : serveur POP, serveur IMAP, SGBD SQL, Bornes Wifi, progiciels, etc.
  - Serveurs : libres (open radius) et commerciaux

Exemples de services client-serveur célèbres [4/8]

- Pour l'acheminement des e-mails (MX : Mail eXchanger, ou MTA : Mail Transfer Agent)
  - Avec l'arrivée d'Internet : SMTP (Simple Mail Transfert Protocole)
  - Clients connus : tous (ou presque) les logiciels d'e-mails, les applicatifs, les serveurs web, etc.
  - Serveurs : sendmail, postfix, qmail, exim, Microsoft Exchange Server, etc.

Exemples de services client-serveur célèbres [5/8]

- Pour le téléchargement des e-mails :
  - Depuis l'arrivée d'Internet, deux protocoles sont principalement utilisés :
    - POP3 (téléchargement des messages sur le poste client),
    - IMAP (plus évolué que POP3, permet la gestion des messages en les laissant sur le serveur).
    - Clients connus : tous (ou presque) les logiciels d'e-mails, etc.
    - Serveurs : qpopper, solid-pop3d, uw-imapd, Microsoft Exchange Server, etc.

Exemples de services client-serveur célèbres [6/8]

- Pour la visioconférence et la téléphonie sur IP :
  - Historiquement : normes ISO H320 (réseaux RNIS), et H323 (sur réseau IP)
  - De plus en plus : SIP (Session Initiation Protocol) défini par l'IEEE
  - Serveurs : libre = Asterisk (H323 et SIP), Open H323 ; commerciaux : CUSeeMe, Radvision, CISCO, etc.
  - Clients : X-Lite, Akiga, etc.

Exemples de services client-serveur célèbres [7/8]

- La gestion d'agendas/calendriers :
  - Format propriétaire Microsoft Exchange Server,
  - Ou iCalendar (RFC 2445), basé sur le protocole WebDAV (échange de fichiers via HTTP) :
    - Clients : iCal d'Apple, Chandler, Lotus Notes, ScheduleWorld, KOrganizer, Mozilla Calendar/Mozilla Sunbird, Ximian Evolution, Windows Calendar, etc.
    - Serveurs : Hula (Novell), Open-Xchange (Novell), OpenGroupware, etc.

Exemples de services client-serveur célèbres [8/8]

- Et la liste pourrait s'allonger à l'infini (ou presque). Ce qu'il faut retenir :
  - Pour chaque besoin, s'il ne s'agit pas d'application métier (et encore…), mais s'il s'agit d'un besoin qui peut être commun à plusieurs applicatifs : plutôt que de réinventer la roue, il y a 99,9% de chances que la solution existe déjà ;
  - Seul problème : souvent, il n'existe pas une solution évidente, mais plusieurs. Il faut alors les évaluer, et faire votre choix (➔ de l'importance des critères d'évaluation, ➔ important de bien définir vos besoins)

Architecture [1/15]

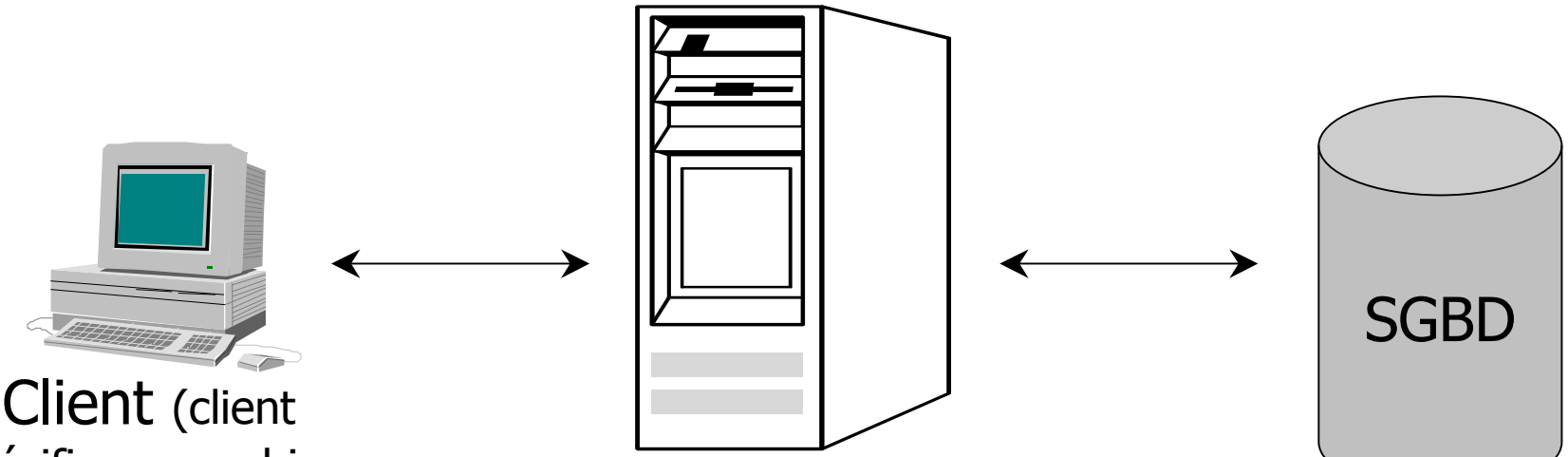
- Définition : c'est l'art de choisir, imbriquer, faire communiquer plusieurs solutions client-serveur afin de répondre à un besoin. Le choix des composantes est fonction des priorités données à :
  - la performance,
  - la sécurité,
  - l'évolutivité (suivant les besoins),
  - la facilité de gérer la montée en charge (*scalabilité*),
  - etc.

Architecture [2/15]

- Ex classique (rappel) : architecture 3-tiers :
  - **Couche présentation (ou cliente)** : correspondant à l'affichage, la restitution sur le poste de travail, le dialogue avec l'utilisateur, etc. (interface homme-machine)
  - **Couche métier (ou applicative)** : traitements propres à l'application (exemple : pour un logiciel de compta, gestion de stock, de la paie, gestion des états, de la balance…)
  - **Couche accès aux données** : correspondant aux données qui sont destinées à être conservées sur la durée, voire de manière définitive. Pour faire simple: mécanismes de cache + SGBD

Architecture [3/15]

- Architecture 3-tiers :

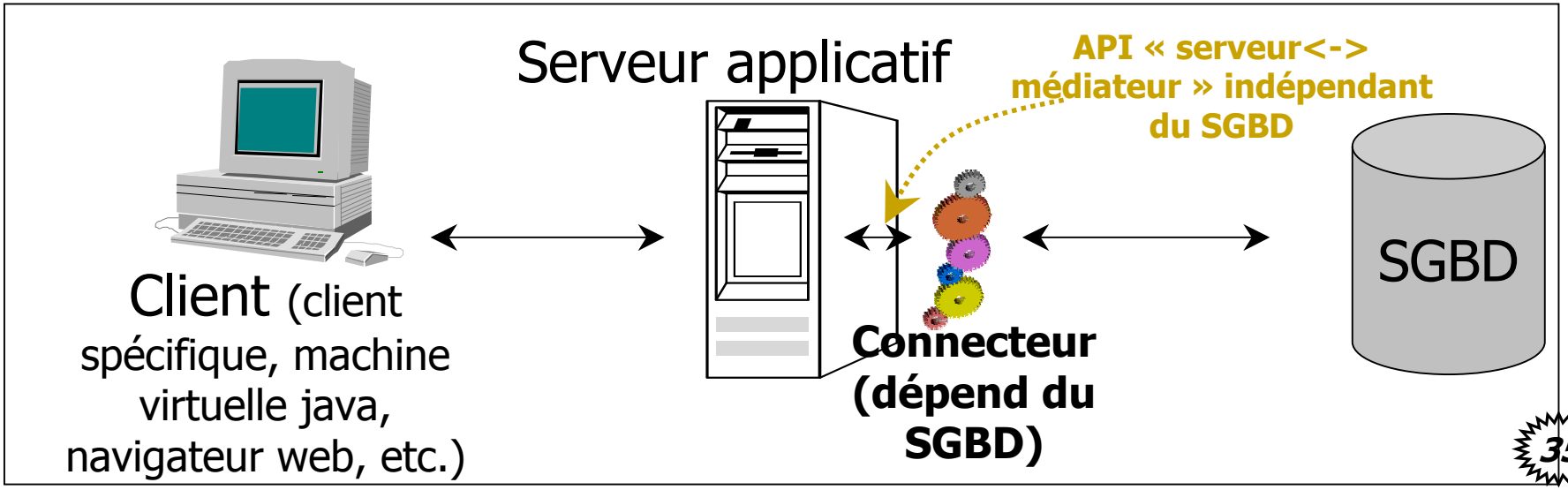


Client (client spécifique, machine virtuelle java, navigateur web, etc.)

Serveur applicatif

Architecture [4/15]

- Pour ne pas rendre le serveur applicatif dépendant d'un seul gestionnaire de base de données (MS-SQL, Oracle, etc.), l'idée est d'utiliser des ***médiateurs*** ou ***connecteurs***



Architecture [5/15]

- Exemple d'API et protocoles de transport « logiciel applicatif <-> connecteur » :
  - CLI (Call Level Interface, API de programmation proposée par le consortium X/Open comme API). Ex. d'implémentation de l'API CLI :
    - SAG (proposé par X/Open),
    - ODBC (Open DataBase Connectivity : Microsoft, etc.),
    - JDBC (Java DataBase Connectivity : Java/Sun).
  - RDA (Remote Database Access, protocole de transport proposé par l'ISO en 1993).

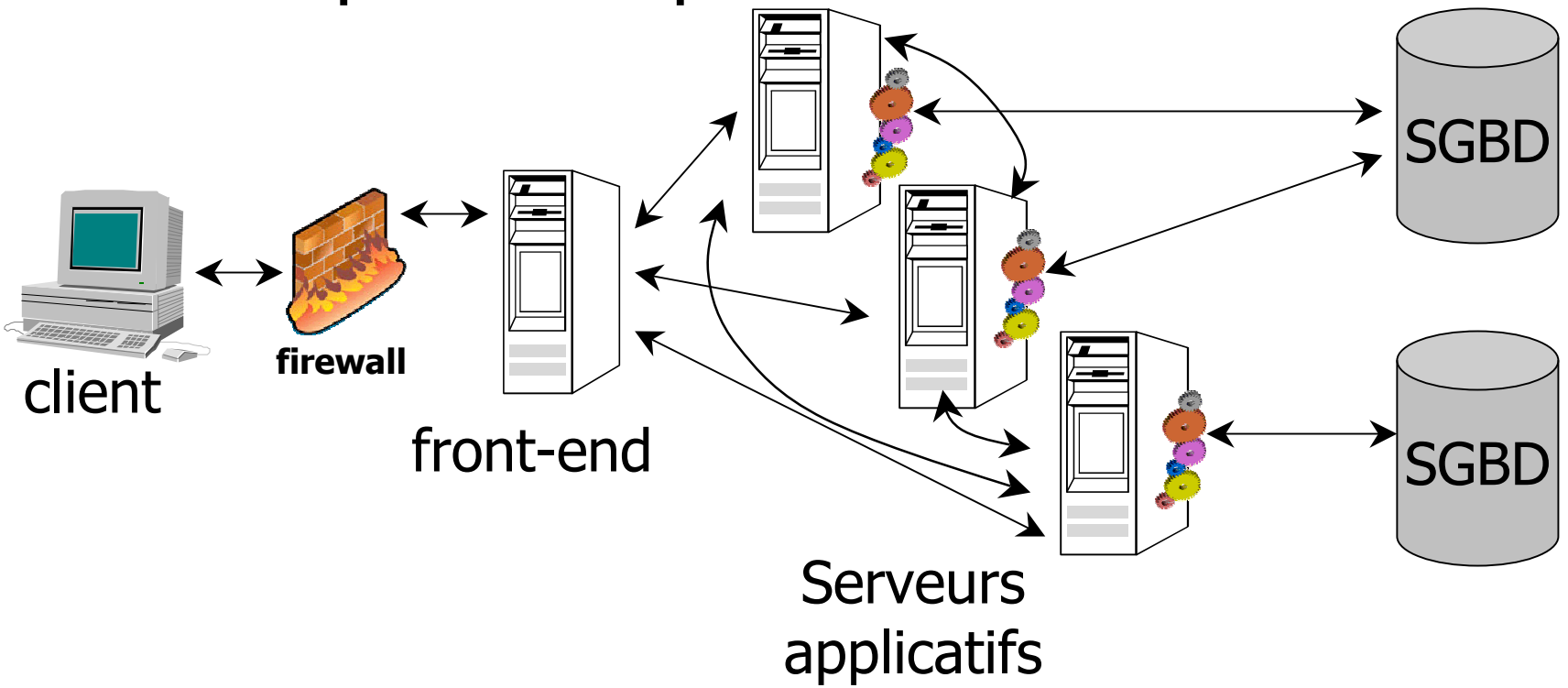
Architecture [6/15]

- Exemple de connecteur : ODBC

Architecture [7/15]

Architecture [8/15]

- Architecture N-tiers (multi-tiers) : la couche applicative est elle-même composée de plusieurs niveaux



Architecture [9/15]

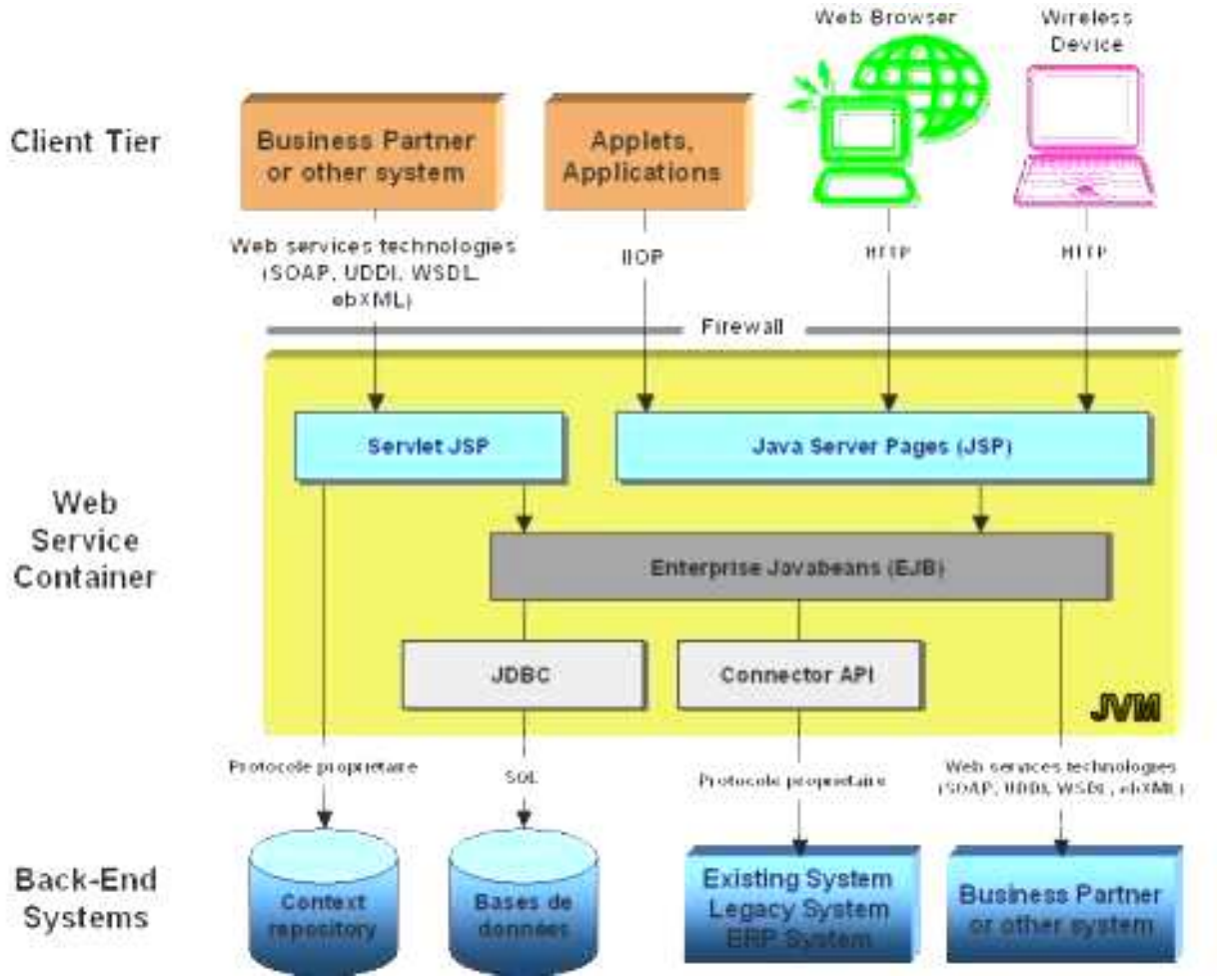
- Architectures N-tiers = couches :
  - La partie cliente est appelée « couche de présentation »,
  - La couche applicative est souvent découpée de façon logique en deux sous-couches :
    - Une couche de services (ex : créer un compte, rechercher un client, calculer un amortissement, …),
    - Une couche d'objets métier (ex : facturation, génération d'un non de commande, gestion des clients, …) ;
  - et enfin, la gestion du stockage/récupération des données/cache est effectué par une « couche d'accès aux données ».
- Intérêt : changer un élément ne remet pas en cause tous les composants

Architecture [10/15]

- Ex de deux grandes offres commerciales :
  - JAVA (Sun),
  - .NET (Microsoft).
- De nombreux points communs :
  - N-tiers,
  - Programmation objet,
  - Connecteurs comme couche d'abstraction avec les SGBD (JAVA : JDBC, .NET : ODBC),
  - Un environnement d'exécution :
    - Runtime pour JAVA, qui contient une Java Virtual Machine,
    - Framework pour .NET, avec un Common Langage Runtime.

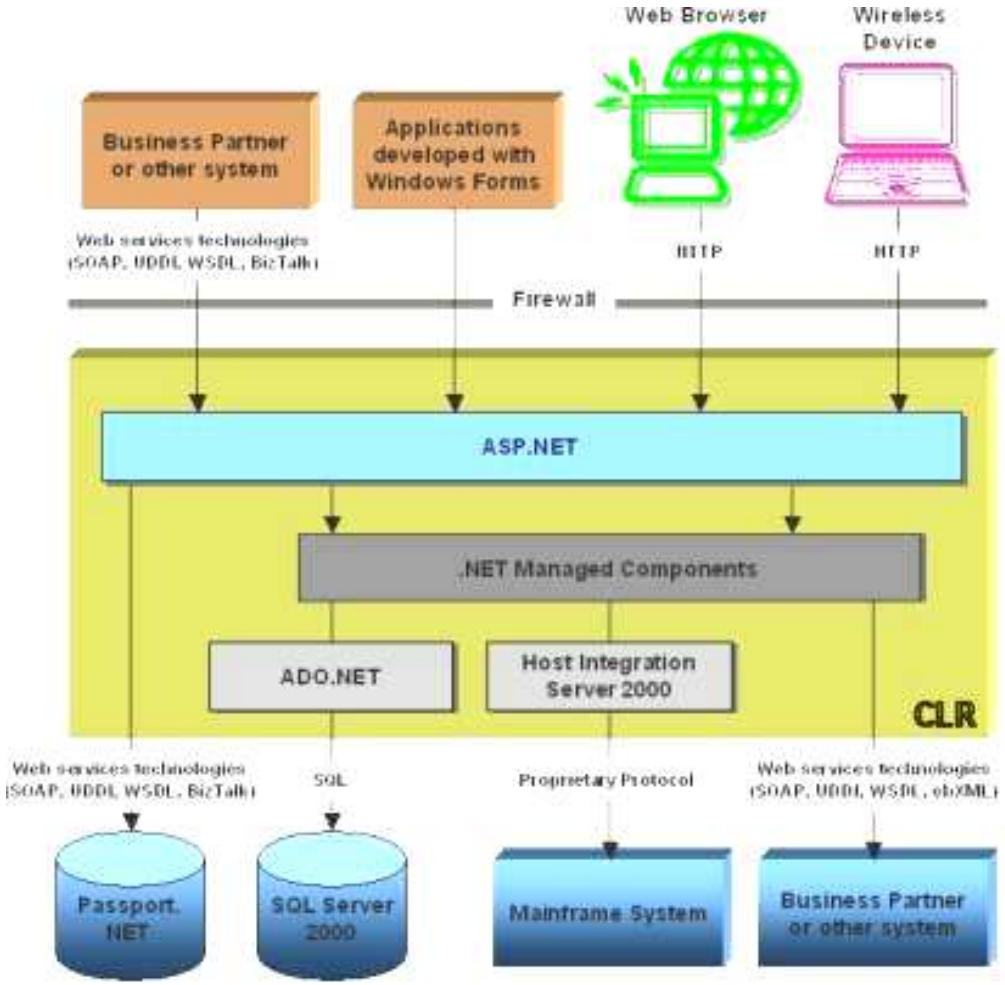
Architecture [11/15]

- Architecture JAVA :



Architecture [12/15]

- Architecture .NET :

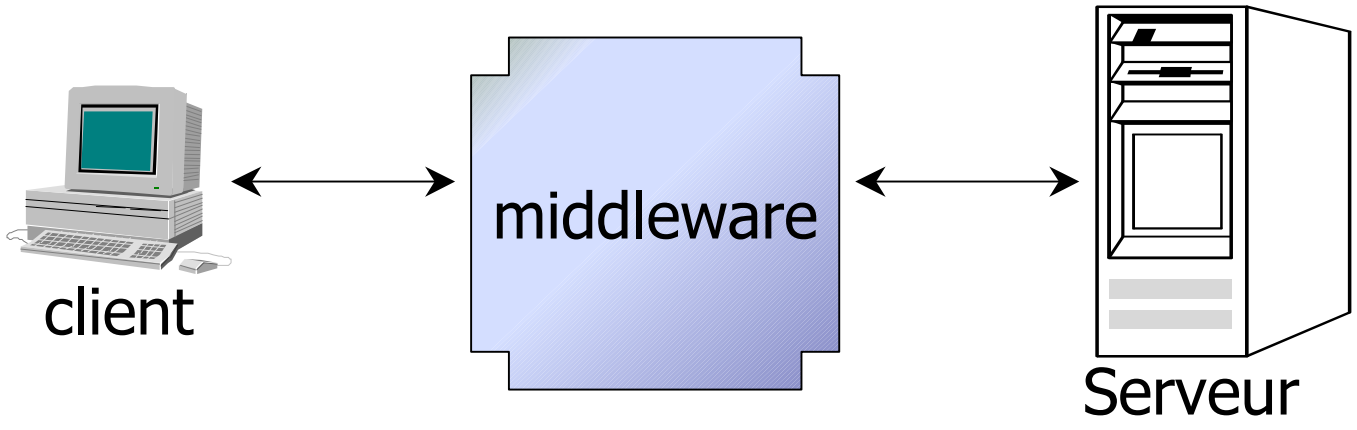


Architecture [13/15]

- Finalement, les deux architectures sont très proches. Quelques différences :
  - JAVA est mono-langage (JAVA), alors que .NET est multi-langages (C#, Visual Basic, voire… JAVA !) ;
  - JAVA est multiplateformes (le JAVA Runtime Environnement existe sous Windows, Linux, Unix, MacOS, etc.), alors que le Framework .NET n'est disponible que sous Windows.

Architecture [14/15]

- Le **middleware** :
  - logiciel qui assure les dialogues entre clients et serveurs hétérogènes, ou entre 2 applicatifs n'ayant pas les même API de RPC. Fait de l'« adaptation de protocole » des couches 5, 6, 7 du modèle OSI



Architecture [15/15]

- Rôles des middlewares :
  - négociation des connexions,
  - conversion des types de données échangées,
  - fiabilisation et sécurisation des échanges.
- Rq : les connecteurs peuvent être vu comme des middleware
- Exemples de produits commerciaux :
  - SequeLink de Techgnosis, DAL de Apple, DRDA d'IBM, IDAPI de Borland, EDA/SQL de Information Builders, etc.

« Business Intelligence » et langages L4G [1/12]

- Avec l'informatisation de la plupart des services de production des entreprises (gestion du personnel, compta, finances, gestion de stock, gestion des process, gestion documentaire, gestion de prospect, communication client, applications métier, …), qui stockent leur données dans un ou plusieurs SGBD, il est apparu intéressant de croiser toutes ces données pour créer des **tableaux de bord**.

  ➔ création de d'« entrepôts de données » (*datawarehouse*) pour faire de l'**informatique décisionnelle** (*business intelligence*)

« Business Intelligence » et langages L4G [2/12]

- Définitions :
  - L'**informatique décisionnelle** (ou Business Intelligence) désigne les moyens, outils et méthodes qui permettent de collecter, consolider, modéliser et restituer les données d'une entreprise (création de tableaux de bords/reporting) en vue d'offrir une aide à la décision et de permettre aux responsables de la stratégie d'avoir une vue d'ensemble de l'activité traitée.
  - Ce type d'application (appelé parfois **infocentre**) utilise en règle générale un **datawarehouse** (ou entrepôt de données) pour stocker des données provenant de plusieurs sources hétérogènes, grâce à des traitements lourds type batch (par ex via des **ETL** - Extract-Transform-Load - ou **datapumping**)

« Business Intelligence » et langages L4G [3/12]

- L'informatique décisionnelle vise à évaluer :
  - un certain nombre d'**indicateurs** ou de mesures (que l'on appelle aussi les **faits** ou les **métriques**),
  - restitués selon les **axes d'analyse** (que l'on appelle aussi les **dimensions**).
- Pour ce, on utilise plusieurs concepts :
  - **Le tableau** (à double entrées) : 2 axes d'analyse (ex : ventes des différents produits dans le temps)

	Produit1	**… produit M**
Année 0	Vente produit 1 année 0	Vente produit M année 0
… Année N	Vente produit 1 année N	Vente produit M année N

« Business Intelligence » et langages L4G [4/12]

- **Le cube** (3 dimensions). Ex : évolution des ventes dans le temps des produits, selon les pays.
- **L'hypercube** (N axes)

- Les systèmes de gestion de base de données sachant gérer N dimensions sont classées sous le sigle **OLAP** (OnLine Analytical Processing)
- La recherche de données pertinentes dans toutes ces données et l'élaboration de tableaux de bords ad hoc s'appelle le **datamining** (fouille de données).

« Business Intelligence » et langages L4G [5/12]

- outils du monde décisionnel pour « naviguer » dans les différentes dimensions d'1 hypercube :
  - le **drill down** ou le forage avant : c'est la possibilité de « zoomer » sur une dimension (par exemple d'éclater les années en 4 trimestres pour avoir une vision plus fine, ou de passer du pays aux différentes régions) ;
  - le **drill up** ou le forage arrière : c'est l'opération inverse qui permet d'« agréger » les composantes de l'un des axes (par exemple de regrouper les mois en trimestre, ou de totaliser les différentes régions pour avoir le total par pays) ;

« Business Intelligence » et langages L4G [6/12]

- le **slice and dice** (« hacher menu », c'est-à-dire couper en lamelles puis en dés) : c'est une opération plus complexe qui entraîne une permutation des axes d'analyse (par exemple, on peut vouloir remplacer une vue par pays/régions par une nouvelle vue par familles de produits) ;
- le **drill through** : lorsqu'on ne dispose que de données agrégées (indicateurs totalisés), le drill through permet d'accéder au détail élémentaire des informations (chaque vente de chaque produit à chaque client dans chaque magasin).

« Business Intelligence » et langages L4G [7/12]

- Ex. anecdotique (et célèbre) de conséquence du datamining :
  - si on baisse le prix du soda de 5%, on va par exemple en augmenter les ventes de 15% ; ce que l'on savait sans datamining (une analyse statistique suffit) ;
  - mais le datamining recherchant des corrélations entre évènements, on pourra révéler l'élément inattendu (bien qu'évident a posteriori) : l'augmentation des ventes de sodas font augmenter d'une proportion voisine celle des cacahuètes (certainement par l'association d'idées : "Puisque j'achète du soda, il me faut aussi des cacahuètes");
  - aussi, même si la marge sur le soda est faible, si celle sur les cacahuètes est importantes, la conclusion se tire d'elle-même : baisser les marges d'un produit d'appel (le soda) permet de monter les ventes d'autres produits (les cacahuètes) sur lequel le bénéfice est important.

« Business Intelligence » et langages L4G [8/12]

- Les langages permettant ce genre de traitement de données, indépendant de tout langage natif les ayant généré (C, Cobol, Java, C#…) sont appelés langages de 4ème génération, ou L4G. Il s'agit souvent de langages :
  - Événementiels,
  - Orientés objet.

« Business Intelligence » et langages L4G [9/12]

- Les L4G supportent :
  - la déclaration de variables,
  - les calculs et assignations,
  - les tests du type IF et CASE,
  - les boucles LOOP et FOR EACH,
  - le traitement des erreurs et des exceptions par des ordres du type SIGNAL,
  - la construction d'objets par réutilisation et spécialisation d'autres objets ;

« Business Intelligence » et langages L4G [10/12]

- Et les L4G intègrent :
  - des bibliothèques de classes réutilisables, en particulier des interfaces graphiques ;
  - des fonctions de service pour effectuer l'accès aux bases de données et aux fichiers, en assurant la transparence des sources de données ;
  - une programmation événementielle permettant de déclencher des fonctions suite à des événements, tels un clic souris ou la mise à jour d'une variable.

« Business Intelligence » et langages L4G [11/12]

- Les outils que nous devons trouver dans tout bon L4G sont :
  - Un éditeur de fenêtre,
  - Un éditeur d'état ou de rapport,
  - Un système d'accès aux fichiers et aux différents SGBD,
  - Un compilateur ou un interpréteur.

« Business Intelligence » et langages L4G [12/12]

- Ex d'outils commerciaux (liste non exhaustive) :
  - Business Objects (ou BO), de la société éponyme,
  - Business Information Warehouse (société SAP) orienté vers l'ERP (Enterprise Ressources Planning, i.e. « progiciel de gestion intégré ») SAP
  - Enterprise Miner (société SAS)
  - 4Thougth & PowerPlay (société Cognos)
  - Latitudes (société Synaxe)
  - Advantage Data Transformer (Computer Associates)
  - Quelques logiciels libres : Weka, Open BI