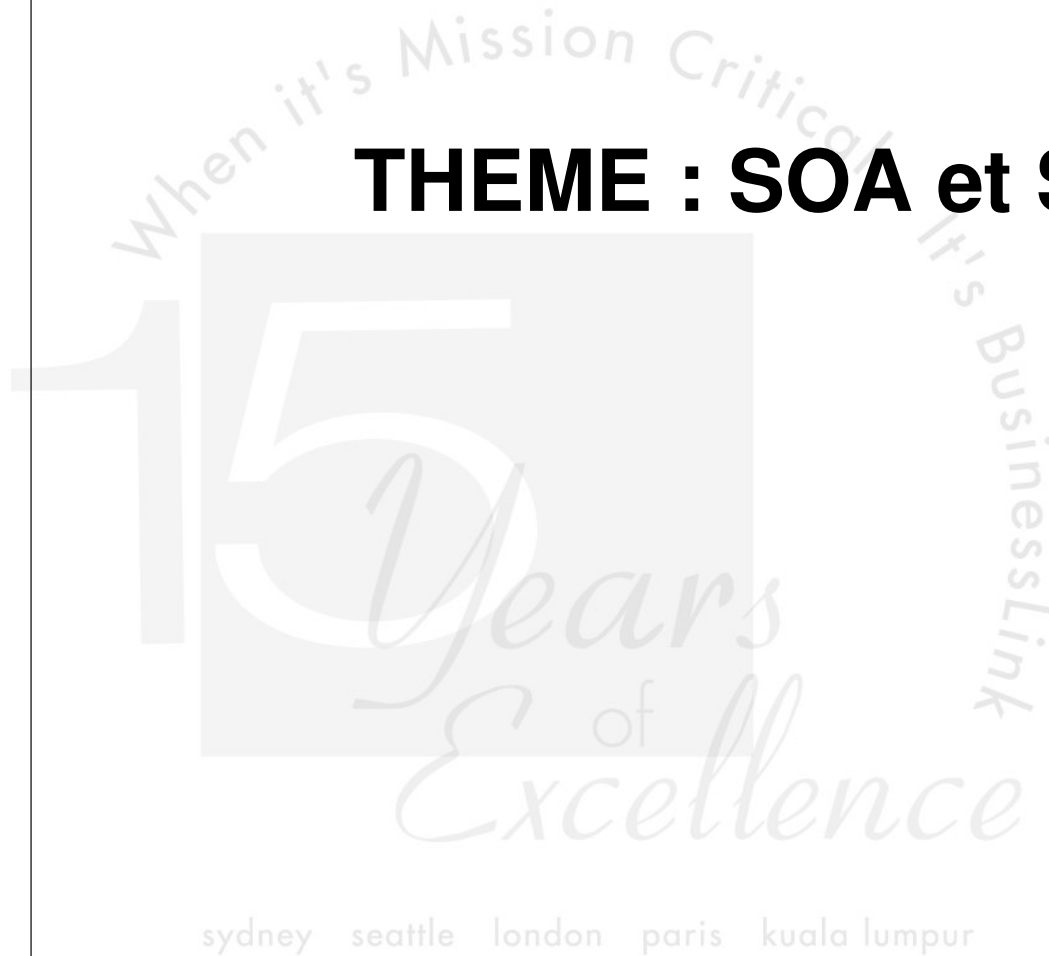


THEME : SOA et System I

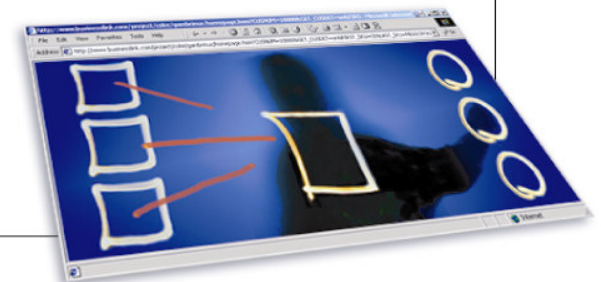


Qui sommes-nous ?

ADVANCED
BusinessLink

- Création en Australie en 1987
- Présent dans 25 pays, en France depuis 2002
- Membre du Partner World for Developers IBM
- IBM all Star
- Apporteur de solutions logicielles de valeur ajoutée sur System i
- CA Monde 2008 : 15.7 M \$
- Effectif Monde : 110 personnes
- + de 1500 clients (106 en France au 30/06/09)

sydney seattle london paris kuala lumpur



REVALORISATION des System i

Notre métier

ADVANCED
BusinessLink

DONNER UNE AUTRE DIMENSION AUX APPLICATIONS System i

REVAMPING

Webisation des interfaces 5250

SOA

Architecture 3 tiers : Séparer la logique métier de l'interface utilisateur

Interfaces utilisateurs nouvelles

Ouverture des applications AS/400 sur le web

Communication avec l'extérieur : webservices

MOBILITE

Accès en mobilité aux utilisateurs itinérants

sydney seattle london paris kuala lumpur



MODERNISATION D'APPLICATIONS

Nos PRODUITS : une offre complète de modernisation d'applications System i

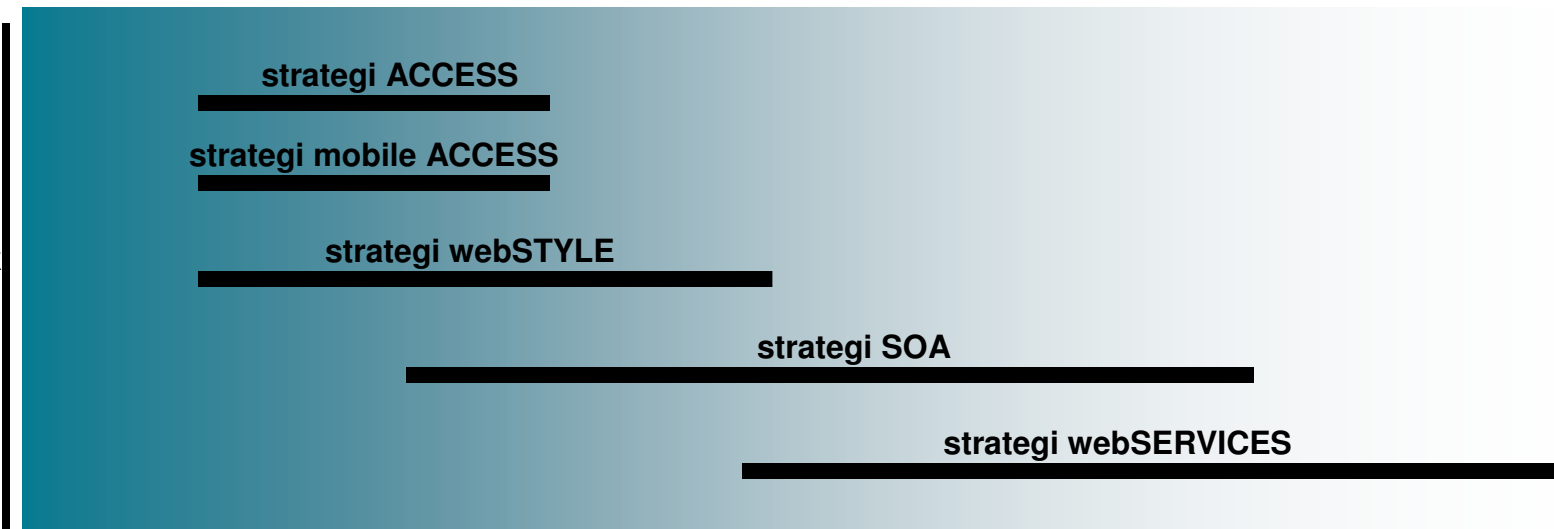
Les Etapes
d'évolution
eBusiness



Modernisation
selon Gartner



Offre
Produits
BusinessLink



1.

Problématique:

Ecosystème en perpétuel changement : clients, actionnaires,
partenaires, usagers

évolution du marché : suivre et anticiper les tendances

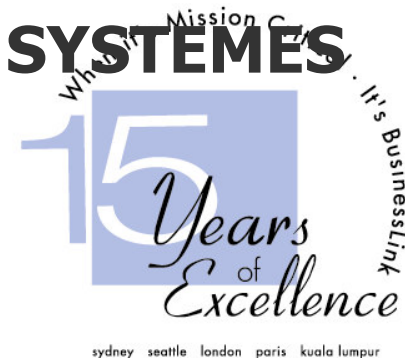
évolution des réglementations

lancement de nouveaux produits et services

répondre aux nouvelles exigences des clients : innovation et qualité

**CONSEQUENCE : REACTIVITE DES SYSTEMES
D'INFORMATION**

sydney seattle london paris kuala lumpur



MODERNISATION D'APPLICATIONS

1.

Problématique:

OR :

Les systèmes d'information n'ont pas été conçus pour de tels changements

**Architectures hybrides
technologies hétérogènes
silos applicatifs
applications monolithiques : non séparation logique métier –
présentation**

sydney seattle london paris kuala lumpur



MODERNISATION D'APPLICATIONS

1.

Problématique:

Solution jusqu'à aujourd'hui :

en interne :

Intégration au coup par coup, point à point

difficulté et coût à maintenir et à faire évoluer dans le temps

une fonction d'un ERP avec telle application spécifique

Module de prise de commande avec telle application de gestion de stocks

En externe :

mais aussi des projets e-business déconnectés des systèmes d'information

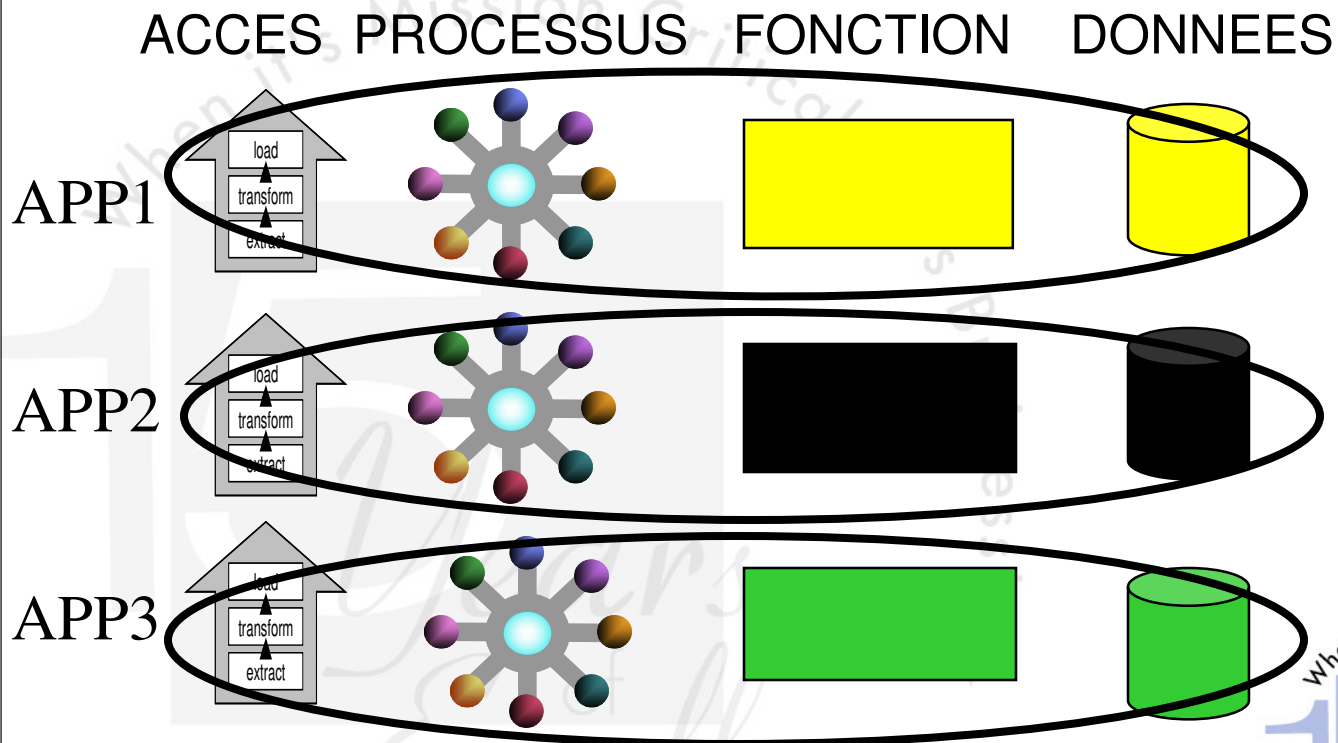
sydney seattle london paris kuala lumpur



MODERNISATION D'APPLICATIONS

SOA et System i

1.



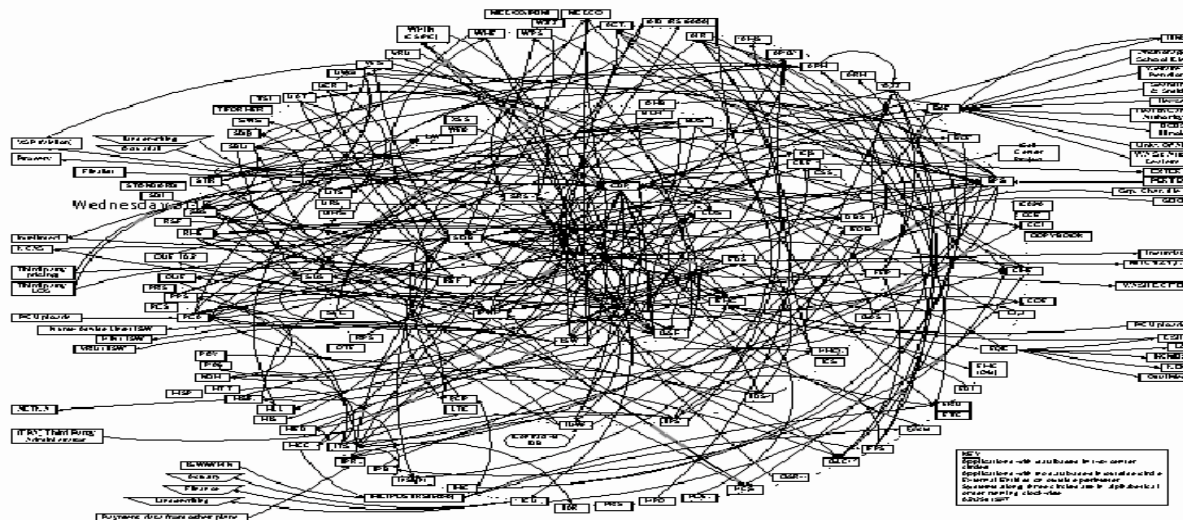
sydney seattle london paris kuala lumpur



MODERNISATION D'APPLICATIONS

1.

RESULTAT : LE SYNDROME DU SPAGHETTIWARE



Processus, fonctions, infrastructures spécifiques à chaque silo d'application ou canal de distribution : manque de flexibilité

Fonctions redondantes par silo

Données manquantes ou redondantes

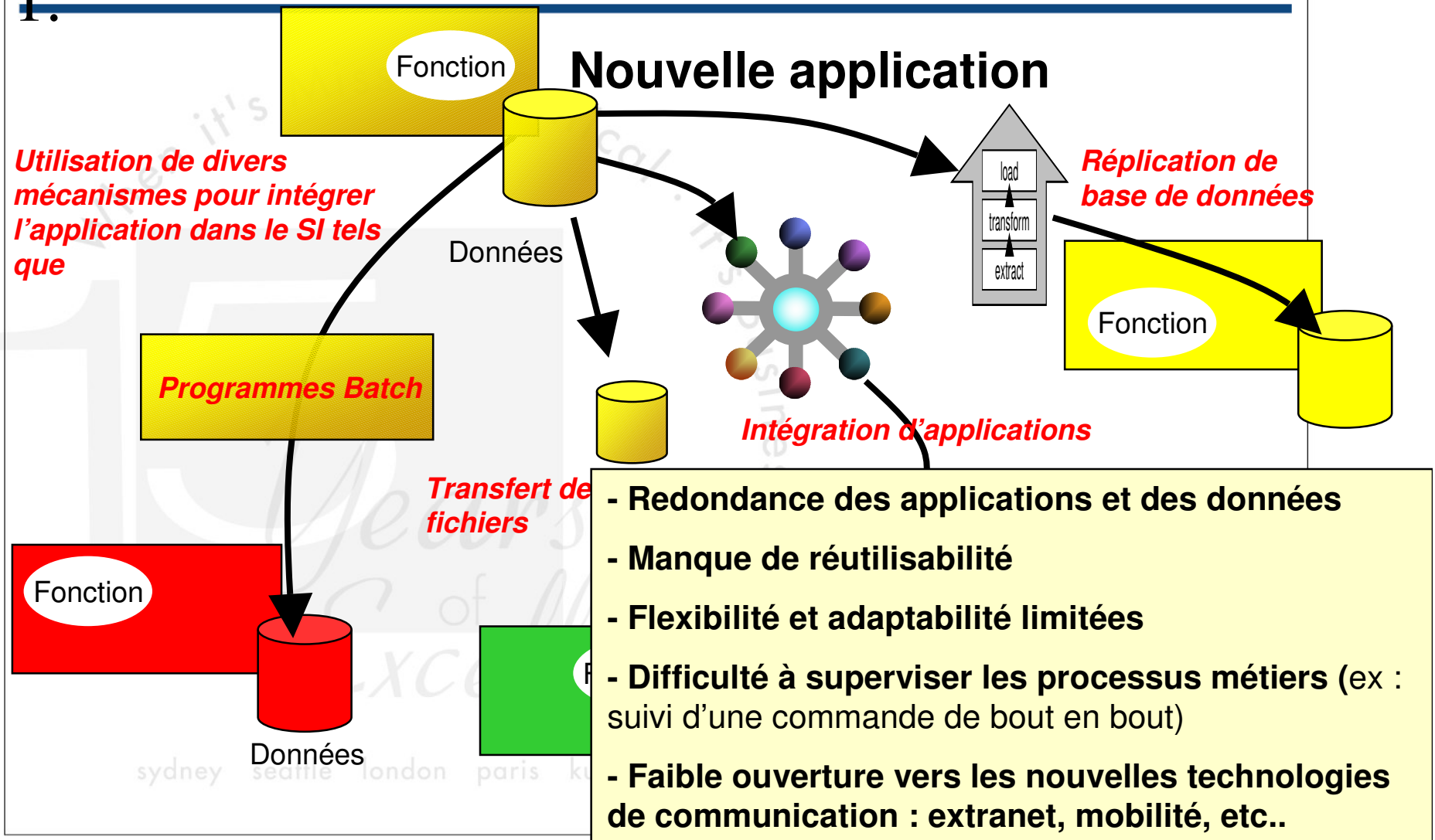
Données couplées aux applications



Exemple : la mise en œuvre d'une nouvelle application

ADVANCED
BusinessLink

1.



Application existantes

1.

Problématique:

D'où, il faut trouver autre chose :

SOA, où comment aligner le système d'information à la stratégie d'entreprise

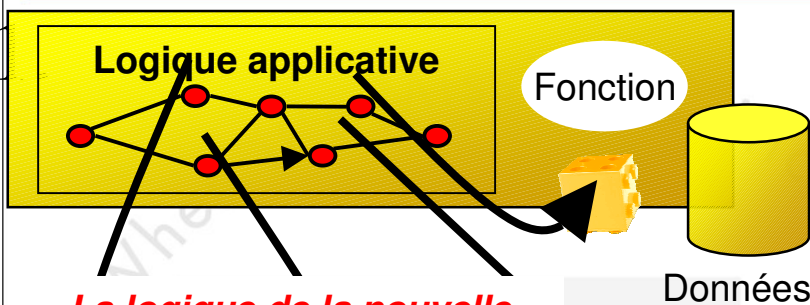
Comment ?

En créant des services métier fonctionnels et techniques pouvant être réutilisés et recomposés en fonction de l'évolution du métier

sydney seattle london paris kuala lumpur



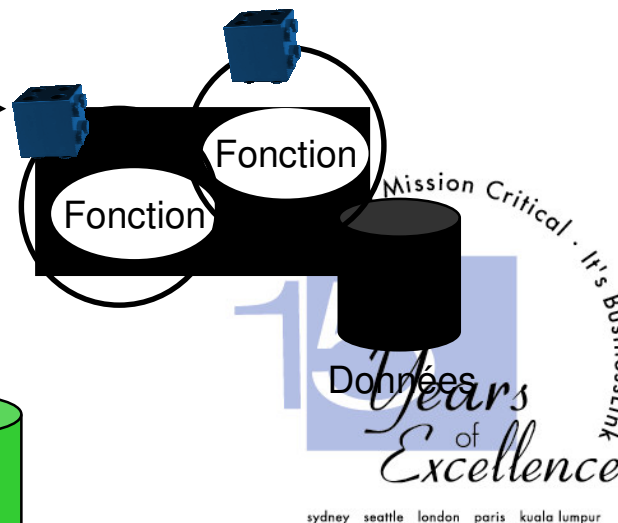
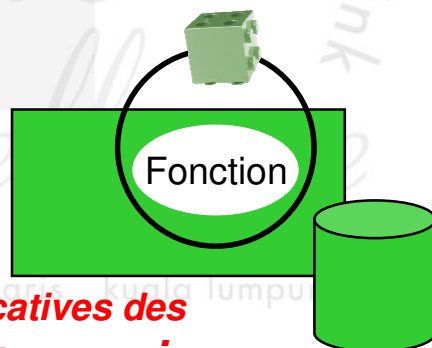
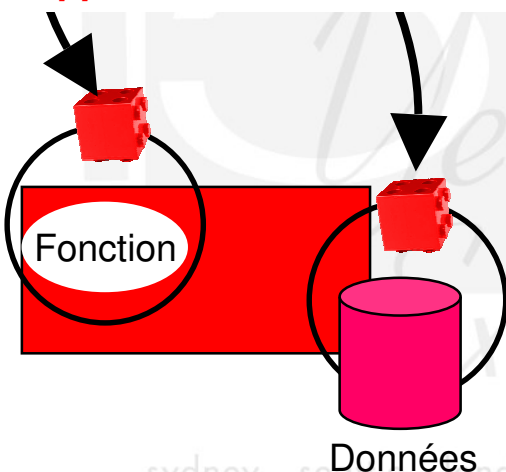
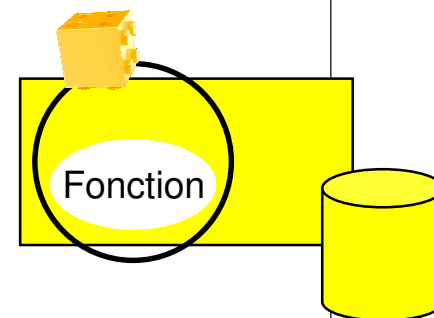
APPROCHE SOA :



Nouvelle application - développement SOA

La logique de la nouvelle application s'appuie sur des appels aux services proposés par les autres applications ...

*De nouvelles fonctions peuvent être développées ... et exposées elles même comme des **services***



*Les fonctions ou les données significatives des applications sont exposées comme des **services** ...*

Application existantes

- **SOA : Services Oriented Architecture – Architecture orientée services**

- **Service**

- Une fonction métier unitaire, répétitive, réutilisable (nombre de sinistres d'un client, vérification du crédit d'un client, tarification d'un contrat, état d'un colis, taux de conversion de devise...)
- Fonction autonome qui ne dépend pas d'un contexte ou de l'état d'un autre service

- **Service Orientation**

- Décomposer les fonctions en services
- Lier et exposer les processus métier aux commerciaux, partenaires et clients
- Créer des processus flexibles et intégrés sous forme d'un assemblage ou orchestration de services

- **Service Oriented Architecture**

- Un style architectural pour construire les applications
- Création des services
- Création des applications à partir des services



1.

Exemple avant SOA: une compagnie d'assurance

	Appli 1 - saisie de commande	Appli 2 - saisie de commande	Appli 3 - devis en ligne
Canal de vente	Call center	agents commerciaux	Web
Fonction commune aux 3 applis	Devis	Devis	Devis
Serveur	AS/400		
Langage de dev	RPG		
SGBD	DB2 400		

- Redondance des applications et des données
- Asynchrone : manque de cohérence des données
- Flexibilité et adaptabilité limitées (quand modification de la fonction « Devis »,
- Equipes informatiques hétérogènes
- insatisfaction client – offre qui s'adapte peu à la demande du marché

1.

Exemple après SOA: une compagnie d'assurance

	Appli 1 - saisie de commande	Appli 2 - saisie de commande	Appli 3 - devis en ligne
Canal de vente	Call center	agents commerciaux	Web
Fonction commune aux 3 applis	DEVIS		
Serveur	AS/400		
Langage de dev			
SGBD			

- Une fonction/service unique : création de devis
- synchrone : fiabilité de l'information
- 1 seul référentiel : cohérence de l'information
- réactivé en cas de modification de la fonction
- hausse de la satisfaction client, augmentation du CA, développement du % de commandes en lignes, etc...

MODERNISAT

SOA et System i :

BENEFICES D'UNE APPROCHE SOA

Bénéfices business

- Accroître la flexibilité, l'agilité
- Par la granularité des processus
- Améliorer la réactivité
- Par le développement rapide de services
- Par une réponse rapide aux changements du marché
- Améliorer le service client
- Par une indépendance de l'infrastructure IT

Bénéfices IT

- Développement rapide – Maintenance réduite
- Utilisation de services prédéfinis, réutilisables
- Investissements préservés
- Nouvelle utilisation de l'existant

SOA et System i :

Un nouveau contrat entre opérationnels et service informatique

Business & IT



Compétitivité
Réactivité
Flexibilité

Réutilisation – Agilité - Innovation

SOA :

Une **approche**, et un **style** d'architecture.

Les **services** sont les **éléments de base** qui **peuvent être réutilisés** pour le développement d'autres services ou applications

Met l'accent sur l'assemblage d'application au lieu des détails d'implémentation.

*“Une architecture de système dans laquelle les fonctions applicatives sont construites sous forme de composants (ou services) qui sont **loosely-coupled** et bien définis pour supporter l'**interopérabilité** et améliorer la **flexibilité** et la réutilisation”.*

SOA et System i :

SOA :

Une **approche**, et un **style** d'architecture.

Les **services** sont les **éléments de base** qui **peuvent être réutilisés** pour le développement d'autres services ou applications

Met l'accent sur l'assemblage d'applications au lieu des détails d'implémentation.

*“Une architecture de système dans laquelle les applications (ou les services) sont construites sous forme de composants (ou de services) qui sont **loosely-coupled** et bien définis, afin d'améliorer l'**interopérabilité** et améliorer la **flexibilité** et la réactivité”*

METHODOLOGIE :

Développer des services reposant sur les standards ouverts

Gérer les échanges d'informations entre ces services

Associer ces services au sein de processus ou d'application

Mettre en oeuvre de façon incrémentale

Le service est l'unité atomique d'une architecture SOA.

Une application est un ensemble de services qui dialoguent entre eux par des messages.

Le service peut :

- être codé dans n'importe quel langage (également en RPG/Cobol ?)
- s'exécuter sur n'importe quelle plate-forme (matérielle et logicielle : System i ?).

Le service doit :

- offrir un ensemble d'opérations dont les interfaces sont publiées : interopérabilité (importance de respecter les standards)
- être autonome (disposer de toutes les informations nécessaires à son exécution : « stateless ») ou encore couplage lâche (s'affranchir des contraintes de temps, d'adresse, de version, etc...)
- Correspondre à des fonctions mutualisables au niveau de l'entreprise : réutilisation (donc publiés, invocables, et activables à distance)

Une fois créé, le service peut :

Etre assemblé au sein d'un processus métier afin de créer une application.
On dit alors qu'il est « fournisseur » d'un client (ou « consommateur »)

Consommer lui-même un service : faire appel à un service

Cette interaction entre service consommateur et fournisseur se fait à l'aide d'un médiateur (middleware) qui peut-être :

Un ESB : Entreprise Service Bus

Le WEB : on parle alors de WEA, Web oriented architecture qui est la grande tendance actuelle car reposant sur des normes d'échanges, de format d'échanges communs à tous et disposant de toutes les qualités nécessaires à une SOA (routage, sécurité, etc...), et l'ouverture vers une multitude de services (réelle interopérabilité).

Le Service devient alors WEBSERVICE

PROTOCOLES ET NORMES

Pour communiquer avec d'autres services, il est nécessaire que ceux-ci indiquent le type de données nécessaires à l'exécution du service et celui qu'il fournit en retour : WSDL

Web Services Description Language

Il faut que le webservice soit ensuite localisable, savoir donc quels sont les services mis à disposition et par qui : annuaire UDDI

Universal Description Discovery and Integration

Il faut ensuite qu'un webservice puisse être appelé (ou « invoqué »), donc que l'on puisse établir une requête au service (encapsulage de la requête) : SOAP.

Simple Object Access Protocol

Il faut que le transfert de données puisse être effectué : protocoles internet

HTTP et TCP/IP

Enfin que le format des données échangées soit commun :

XML

SOA et System i :

Et le System i ?

Les problématiques rencontrées :

- System i : système propriétaire (OS, SGBD, Langage, etc...)
- Problématique d'ouverture du System i
- Problématique de réutilisation des programmes existants (legacy) en composants métiers réutilisables en architecture SOA
- Problématique de découpage de programmes structurés, monolithiques en composants métiers isolés « stateless » de type Objet réutilisables (couplage tâche : les services n'agissent pas directement entre eux)
- Problématique d'évolution des programmes existants aux standards SOA (WSDL, XML, SOAP) pour assurer l'interopérabilité
- Problématique d'invocation/appel des composants métiers isolés à partir de nouvelles interfaces utilisateurs (pour en assurer leur distribution)

ETAPE 1 : Isoler les fonctions métier : architecture 2 tiers ou 3 tiers

ETAPE 2 : Exposer les éléments métier comme des services : production et consommation de Services WEB

ETAPE 3 : Connecter les applications et les services : ESB ou au travers du WEB

ETAPE 4 : Orchestrer les services dans un processus métier au travers d'une application nouvelle : BPM (Business Process Management). Exemple : création d'une application web

ETAPE 1 : Isoler les fonctions métier :

Cela nécessite une modernisation du code
RPG/Cobol existant

Pourquoi ?

Nécessaire pour une ouverture vers l'extérieur et
communication avec d'autres applications (intégration
d'applications)

Pour répondre rapidement aux besoins d'évolution
(flexibilité, agilité)

Pour éviter la redondance et simplifier la maintenance (
assurer l'unicité d'une fonction et sa réutilisation)

Pour organiser ses programmes en architecture 3 tiers
(plusieurs interfaces pour une même fonction métier)

ETAPE 1 : Isoler les fonctions métier :

Cela necessite une modernisation du code RPG/Cobol existant

Comment?

Faire évoluer ses programmes en RPG IV, ILE (support du XML, procédures : plus proche d'une méthode Java que des sous routines)

Séparer l'interface utilisateur de la business Logic (logique de programmation objet)

Externaliser les fonctions et faire des appels de programmes externes (meilleure lisibilité, facilite la maintenance, unicité des la fonction)

Aucune obligation de redévelopper dans des langages objets type java (RPG : plus rapide et facile, compétences déjà présentes, débogage facile, intégration native DB2, fiabilité et robustesse, langage orienté application de gestion)

ETAPE 2 : Exposer les éléments métier comme des services : production et consommation de Services WEB

- Interface de type appel de programmes avec paramètres en entrée/sortie à définir
- Définition des conditions d'erreur
- Le reste est affaire de STRATEGI :

Pour la production de webservices

Création du WSDL

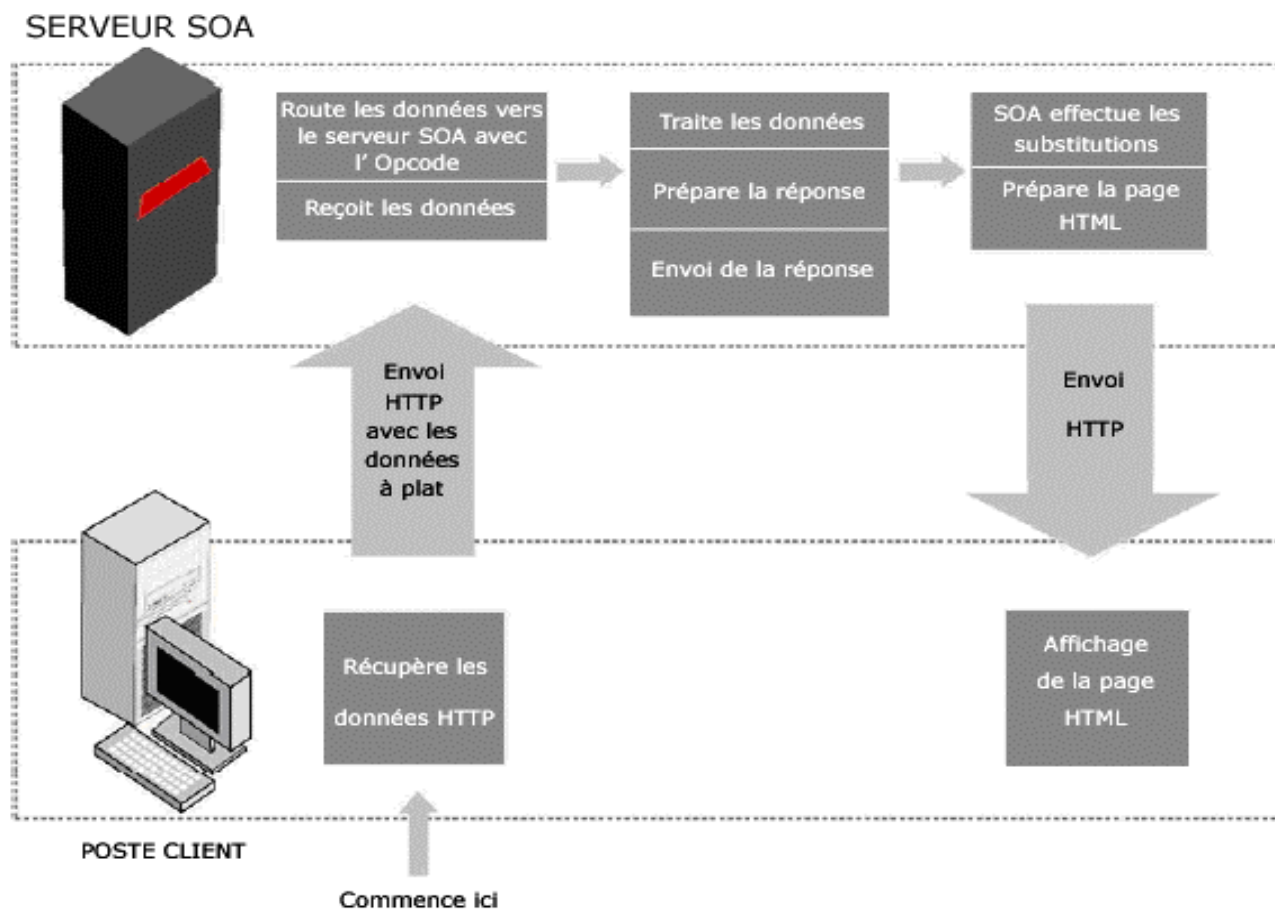
Encapsulage SOAP

Acheminement HTTP

Stockage dans un repository

Format de données : XML

DEROULEMENT DES ECHANGES DANS STRATEGI SOA



ETAPE 3 : Connecter les applications et les services : ESB ou au travers du Web

Solution :

acheter un ESB : services de transport, de transformation des formats de données, de routage intelligent, d'équilibrage de charges et de gestion de la sécurité. Remise en cause de leur utilité réelle

Ou déployer sur le Web : si les webservice sont bien interopérables et invocables – permet d'éviter la communication point à point

ETAPE 4 : Orchestrer les services dans un processus métier au travers d'une application nouvelle : BPM (Business Process Management).

Solution :

acheter une solution de BPM. Remise en cause de leur utilité réelle.

Ou développer une nouvelle application avec une interface web

Who are we?

- Award winning Middleware developer
- Several iSeries world technology firsts, awards, etc
- IBM all star
- Founded 1987
- Offices in Seattle, UK, France, Malaysia, Australia
- 1500+ clients in 22 countries

