

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES (SOA)

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

PLAN :

1- INTRODUCTION

2- NOTIONS DE LA SOA

3- OBJECTIF DE LA SOA

4- PRINCIPE DE BASE

5- COMPARAISON AVEC D'AUTRES ARCHITECTURES

6- Démarche De Conception De Service :

7- EVALUATION

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

1- INTRODUCTION

Années 80 : Début de l'informatisation des SI (grands systèmes).

Années 90 : Début de la microinformatique et les systèmes ouverts (basés sur Unix).

Fin des 90 et début 2000 : fort développement d'internet, dématérialisation et production des biens immatériels.



***Besoins des entreprises et des clients
augmentaient d'une manière excessive***

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

1- INTRODUCTION

*Situation très critique des développeurs
d'application*

Situation interne qui nécessite un changement :

Systèmes = {dizaines d'applications} :

- Protocoles de communication
- Formats de données
- Serveurs locaux ou distants
- Langages différents (Java, C, C++, COBOL, Clipper...)

Situation externe qui procure de nouvelles exigences :

Nouveaux défis apparaissent :

- Changement du créneau commercial
- + des contraintes de concurrence
- Évolution et bouleversement des besoins du client
- Evolution rapide de la IT

1- INTRODUCTION

Problèmes auxquels les développeurs sont confrontés :

- Q1) *Comment faire communiquer l'application développée avec les autres ? Ou pire encore comment faire communiquer toutes les applications entre elles ?*
- Q2) *Comment connaître 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 ?*
- Q3) *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 ?*
- Q4) *Comment faire tourner les applications même en absence ou à défaut de connexion réseau ?*

1- INTRODUCTION

Problèmes auxquels les développeurs sont confrontés :

- *Q1) 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 ?*
- *Q2) 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 ?*
- *Q3) Comment permettre l'interaction avec le système à travers des moyens et outils très simples (surtout avec l'apparition du protocole IPv6) ?*

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

1- INTRODUCTION

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 ».

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

2- NOTIONS DE LA SOA

a- HISTORIQUE :

1992 : Gio Wiederhold introduisait la notion de médiateur (*architecture médiane*), afin de régler la problématique de *l'accès* et de *l'intégration* de l'information.

Médiateur : 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.

1996 : Le groupe de recherche « Gartner Group » exploitait cette idée, pour fonder les bases de la SOA .

2000 : 1ère exploitation de cette architecture suite au succès du langage XML (apparu en 1998).

2002 : Popularisation du terme SOA après l'orientation de beaucoup d'entreprises commerciales vers cette architecture.

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

2- NOTIONS DE LA SOA

b- DÉFINITIONS :

*La SOA est une approche de conception pour les SI, qui se base sur la 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.

2- NOTIONS DE LA SOA

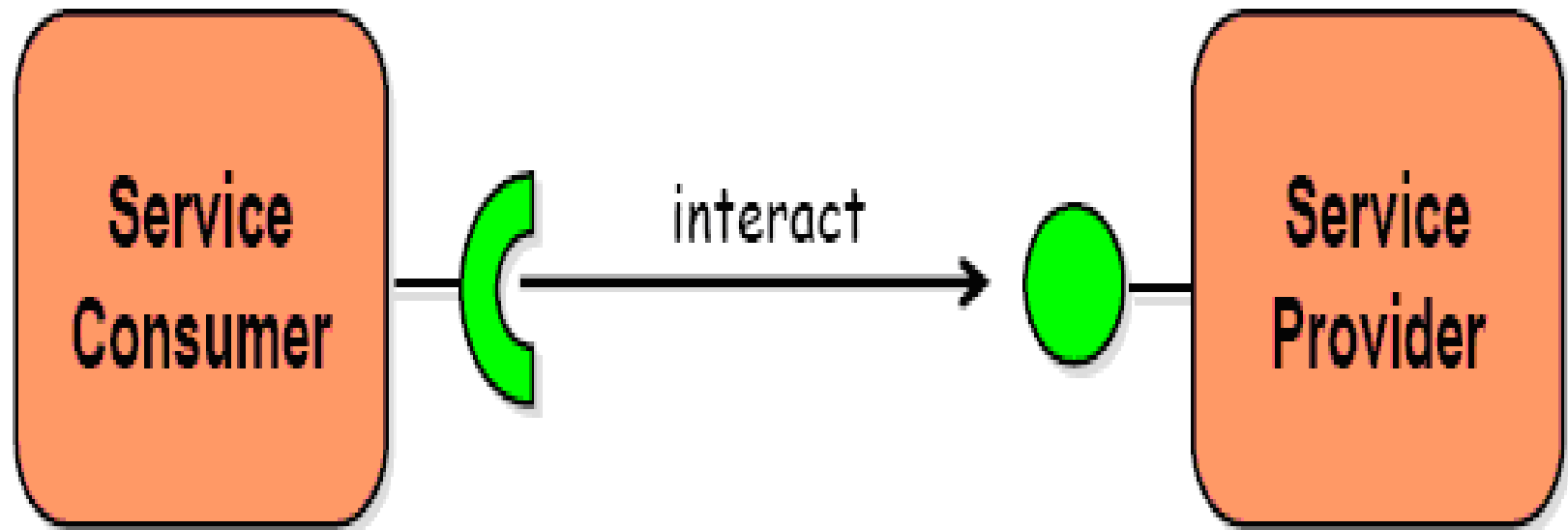
b- DÉFINITIONS :

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.*

2- NOTIONS DE LA SOA

b- DÉFINITIONS :



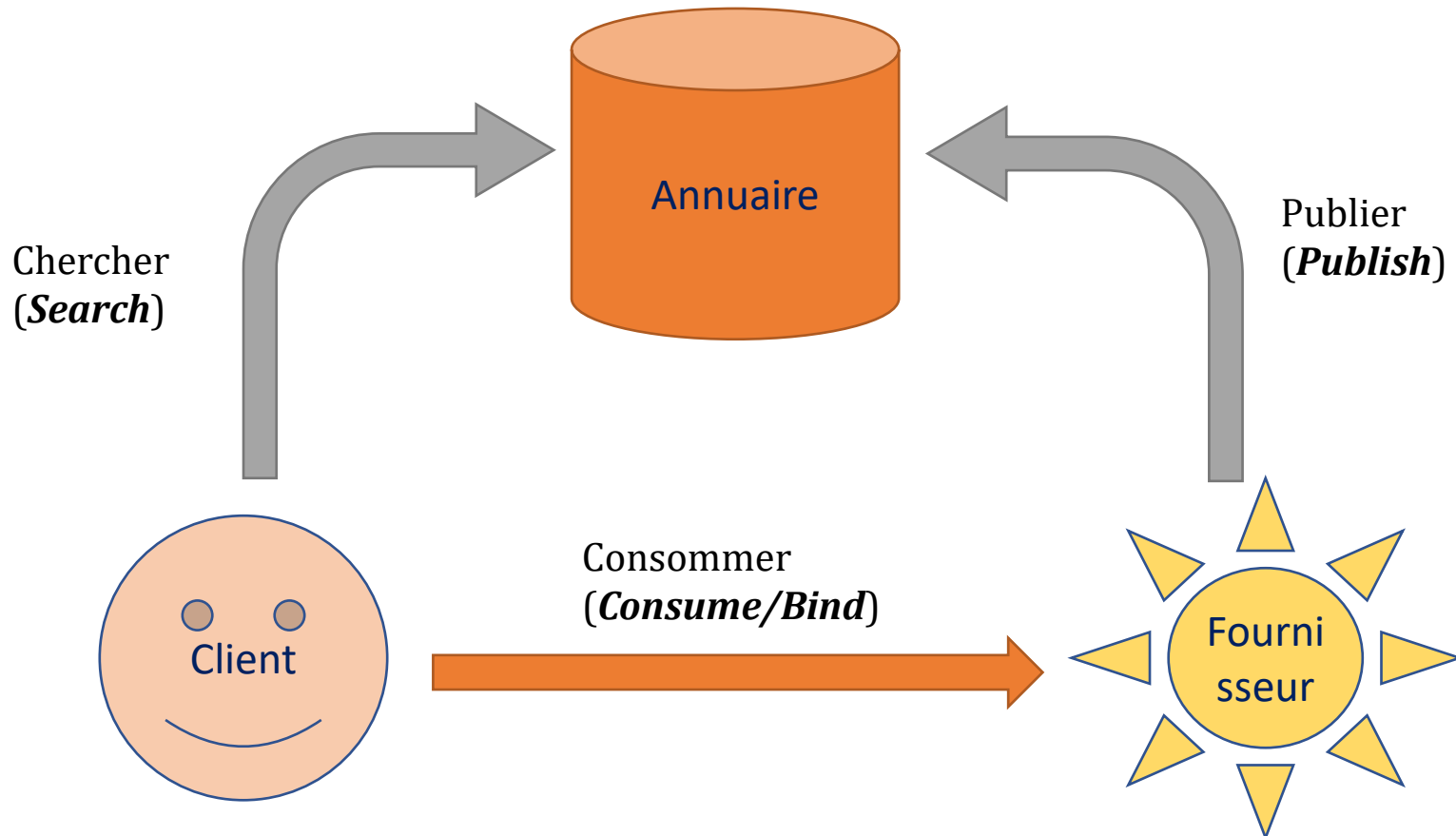
3- OBJECTIF DE LA SOA

- *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).*

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

4- PRINCIPE DE BASE :

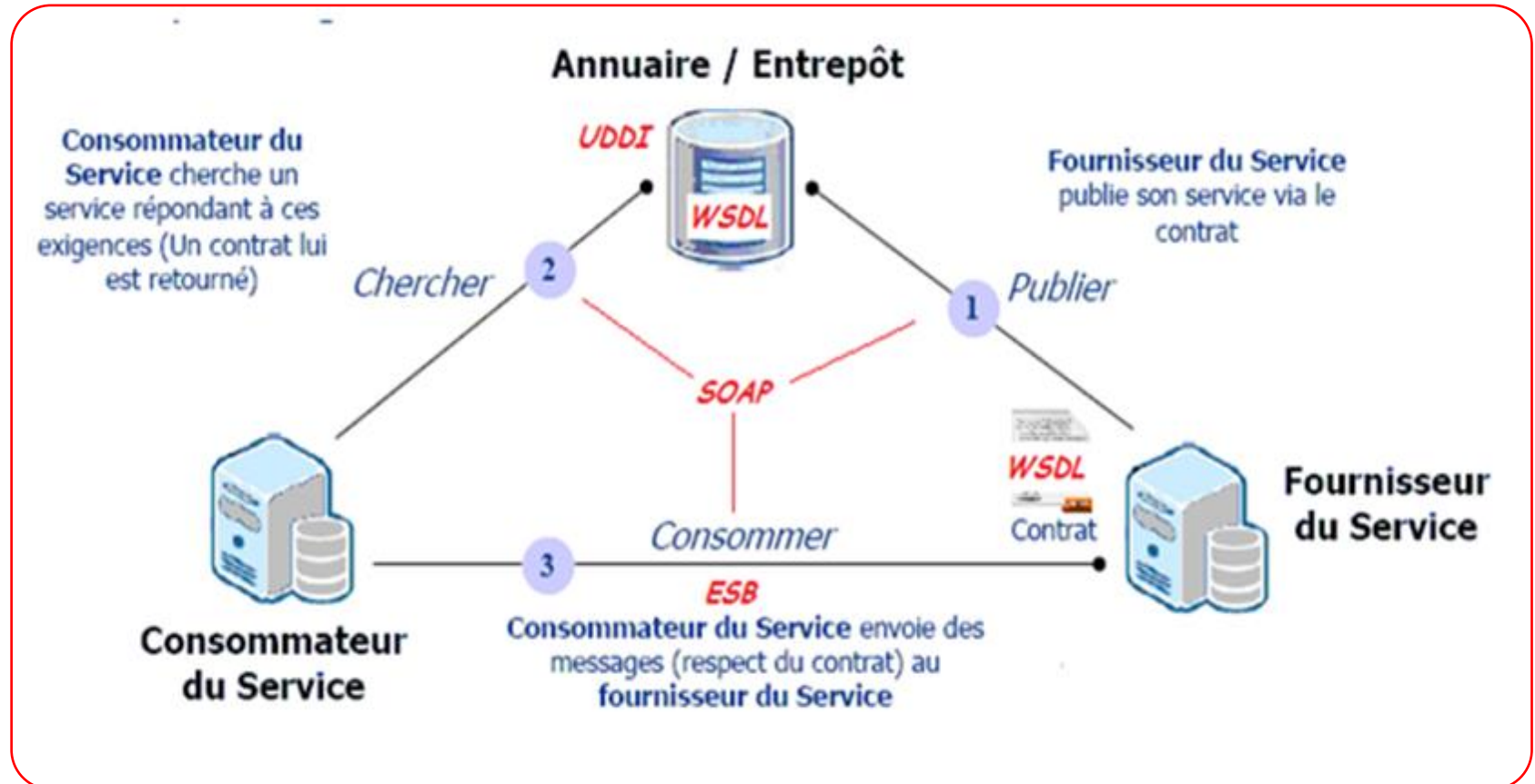
La SOA se base sur



CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

4- PRINCIPE DE BASE :

L'architecture de



WSDL: Web Service Description Language

UDDI: Universal Description Discovery and Integration

SOAP: Simple Object Access Protocol

ESB : Enterprise Service Bus

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

5- COMPARAISON :

a- SOA vs POC & POO

POO

Encapsulation des informations dans un objet doté d'un ensemble de méthodes.

- *Structure et architecture de l'application peu visible (Difficile).*
- *Interactions entre objets enfouies dans le code (les relations entre les différentes parties du système ne sont pas déclarées).*
- *Évolution / modification difficile.*
- *Gestion de la consistance d'un changement délicate*
- *Couplage fort: ce qui rend difficile la réutilisation et accroît la complexité des Systèmes OO.*

POC

Applications distribuées,
Ensemble de composants logiciels,
collaborent, tâche commune,
distants, réseau, hétérogènes,...

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

SOA

Services, annuaire, fournisseur,
client, contrat, standard, ...

Points communs :

- *Aspect modulaire (POO, POC)*
- *Service = boîte noire.*
- *Indépendance de la localisation.*
- *Neutralité vis-à-vis des protocoles de transport.*

Caractéristiques propres :

- *Périmètre fonctionnel exposé au consommateurs.*
- *Découplage (services, client/fournisseur).*
- *Utilisation de standards*
- *Adaptation facile*

5- COMPARAISON :

a- SOA VS CODE MOBILE :

Un *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. Trois catégories principales existantes:

1. Evaluation à distance : 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,

2. Code à la demande : l'utilisateur ne dispose pas de code, mais il demande à un serveur de lui envoyer ce code afin de l'exécuter sur sa machine et bien sûr obtenir un résultat.

3. Agent mobile : il est 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.

5- COMPARAISON :

a- SOA VS CODE MOBILE :

Code mobile

Pièce maitresse = Code (composant logiciel)+ Mobilité

Evaluation à distance

Code à la demande

Agent mobile

- *Points attractants :*
- *Tourne autour d'un code (composant logiciel).*
- *Ce code est mobile.*
- *Ce code retourne un résultat à l'utilisateur.*

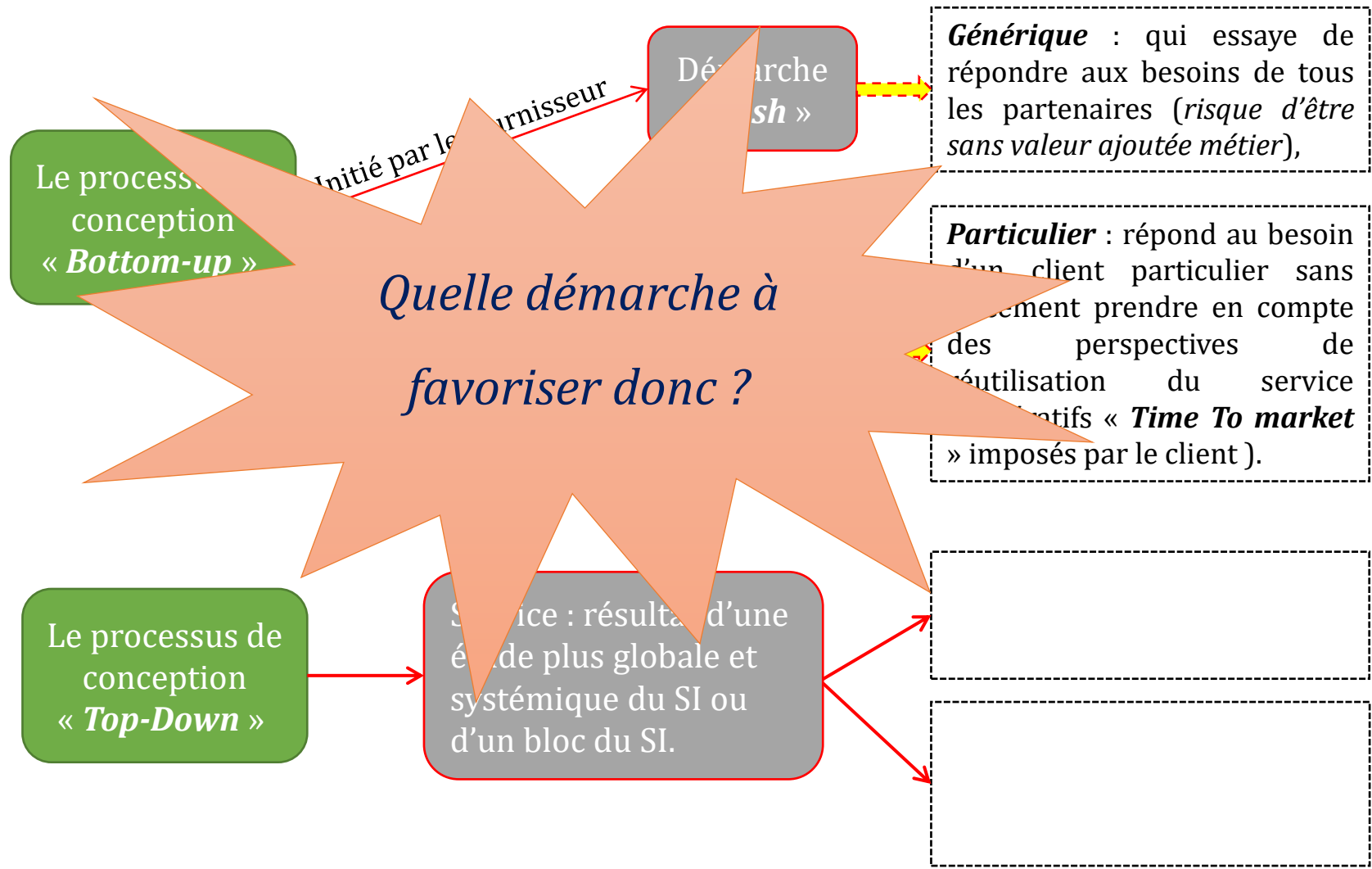
SOA

*Pièce maitresse = Service
(normalisé, standardisé)*

- *Architecture client/serveur.*
- *Pas de code chez le client.*
- *Exécution à la demande du client.*
- *Exécution chez le fournisseur.*
- *Résultat envoyé au client.*

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

6-La démarche suivie dans la conception de service :



CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

6-La démarche suivie dans la conception de service :

Démarche favorable :

Dans l'absolu, il coexistent selon la conception et de considération les

- Démarche itérative

Processus
« Bottom-Up »
Push

Processus
« Bottom-Up »
Pull

Processus
« Top-Down »

Processus itératif et
collaboratif de mise en
œuvre d'un service



particulier. Les 3 peuvent néanmoins, en phase de travail, prendre en

leurs.

- Penser un service : Phase d'émergence du/des service(s)
essaie de répondre à tous les besoins.

Phase de mise en œuvre

que qu'un mono-service qui

- Ne pas considérer l'étude d'urbanisation, s'il y en avait une, comme une cible en elle-même, mais plutôt comme un fil conducteur pour la cible.

CH1: INTRODUCTION AUX ARCHITECTURES ORIENTÉES SERVICES

7- EVALUATION :

Au-delà de tous

Avantages

- ❖ *Accès à distance via un navigateur Web.*
- ❖ *Pas de logiciel sur le poste client (client léger)*
- ❖ *Possibilité d'interagir par n'importe quel appareil connecté à Internet.*
- ❖ *Compatibilité avec différents systèmes d'exploitation et différentes plates-formes*
- ❖ *Système d'information flexible (anticiper les évolutions technologiques),*
- ❖ *Réutilisation de code.*
- ❖ *Une plus grande tolérance aux pannes (maintenance facile)*

Inconvénients

- ❖ *Recentralisation des systèmes (charge très importante sur le serveur).*
- ❖ *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).*
- ❖ *Lenteur d'exécution (services asynchrones).*
- ❖ *Manque de maturité de standards (course à la spécification entre W3C et OASIS).*

FIN