CHAPITRE 2. LES SERVICES

Plan

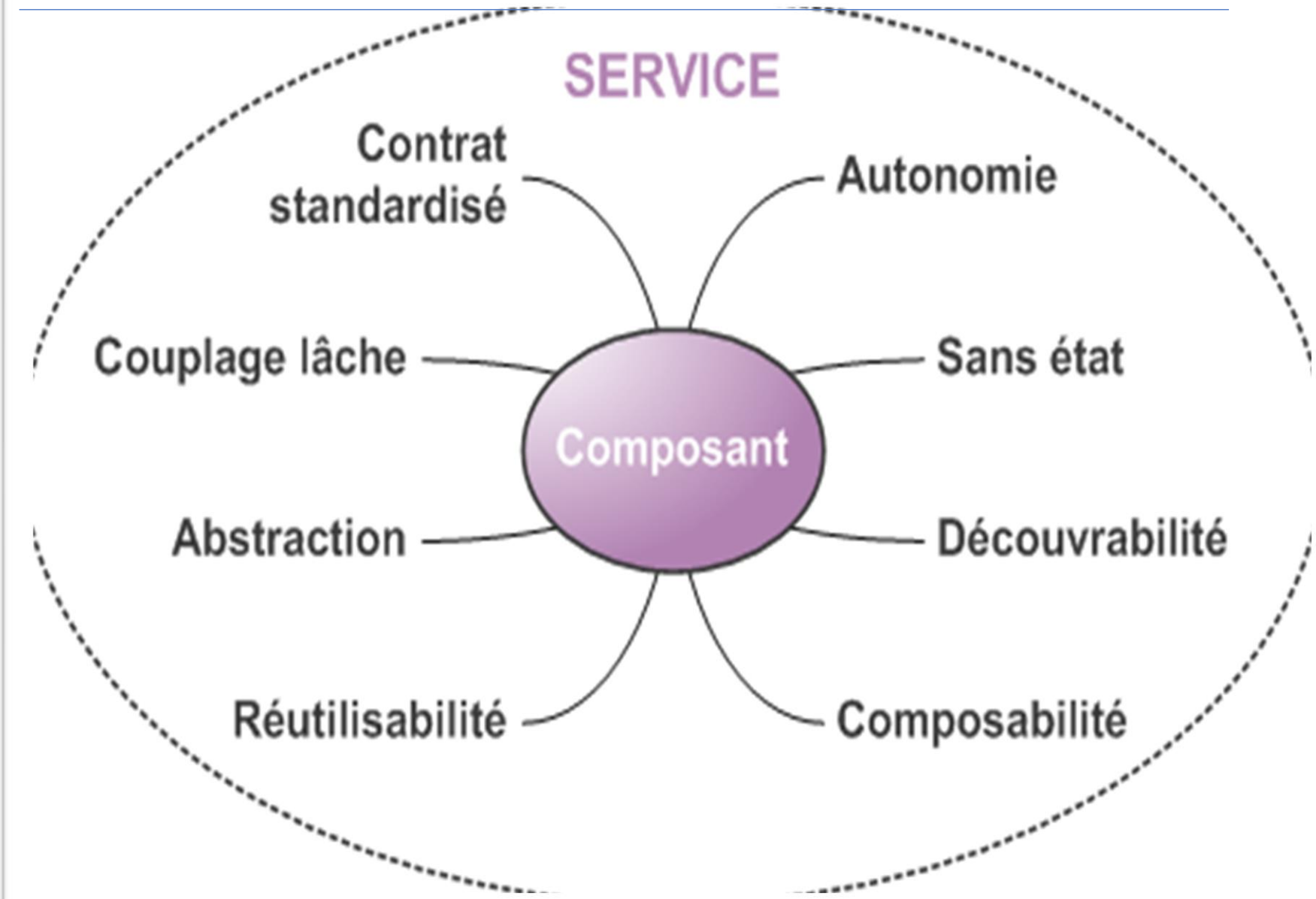
1- INTRODUCTION :

Le service est l'élément clé dans la SOA, cependant cette notion reste si difficile à avoir un consensus dans sa définition.

Les discussions autour de ce sujet débouchent souvent sur :

- des réponses alambiquées et incertaines,
- ou des débats enflammées et souvent stériles.

2- DÉFINITIONS :



2- DÉFINITIONS :

Définition adoptée :

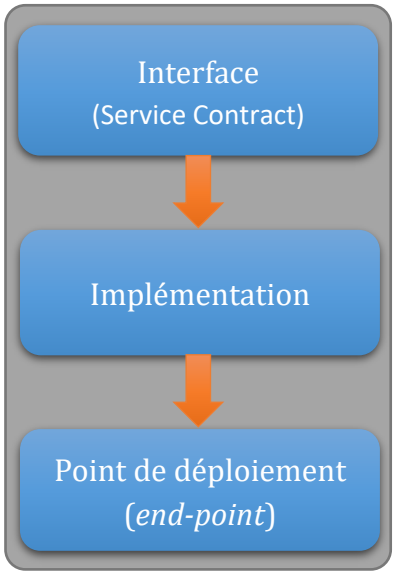
Un Service est un composant logiciel distribué, exposant les fonctionnalités à forte valeur ajoutée d'un domaine métier, et qui vérifie les huit caractéristiques (aspects) suivantes :

- ❑ Contrat standardisé
- ❑ Autonomie
- ❑ Couplage lâche
- ❑ Sans état
- ❑ Abstraction
- ❑ Découvrabilité
- ❑ Réutilisabilité
- ❑ Composabilité

3- Structuration de service :

Un service est composé de trois parties principales :

1- L'interface : qui constitue la partie « publique » et contient entre autres la liste des opérations, les entrées et les sorties, types de données échangées, protoclols utilisés. Elle décrit **ce que fait le service**, sans expliquer comment il le fait.



2- L'implémentation : qui constitue la partie « privée » et correspond au développement des fonctionnalités du service (le code, algorithmes, BD, règles). Même si l'implémentation dépend de la plateforme utilisée, l'approche SOA permet d'abstraire l'hétérogénéité technique grâce à la publication d'une interface commune appelable.

3- Le point de déploiement (end-point) : qui représente le point par lequel le service peut être invoqué, généralement il correspond à une adresse URL.

Un service = Interface (quoi ?) + Implémentation (comment ?) + Endpoint (où ?)

4- CARACTÉRISTIQUES :

1- Contrat standardisé :

L'ensemble des services d'un même Système Technique sont exposés à travers des **Contrats** respectant les mêmes règles de **Standardisation** .

Le contrat de service définit un accord entre le fournisseur et le consommateur :

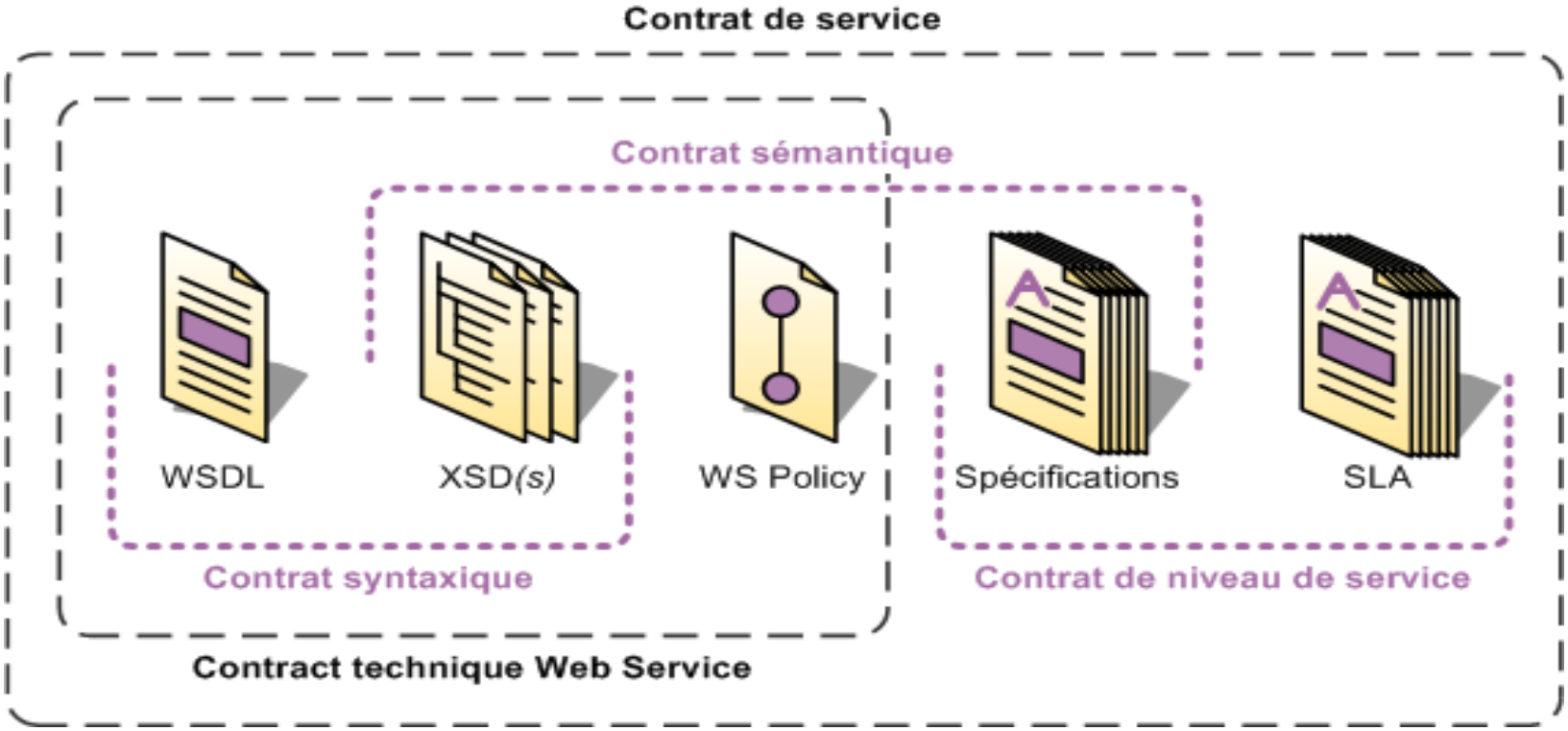
- **Contrat syntaxique (**Document technique) **:** décrire l'interface du service (nom du traitement, ses paramètres I/O, Endpoint, ses contraintes structurelles sur les données ..).

- **Contrat sémantique :** description informelle du service (valorisation des msg, les exceptions, les près et post conditions techniques (volume de données(10 Mo),...) ou métier (écriture comptable équilibrée..), ..).

- **Contrat de niveau de service :** QoS et Service Level Agreement SLA (temps max de réponse, plages horaires d'accessibilité, temps de reprise après interruption, ..).

4- CARACTÉRISTIQUES :

1- Contrat standardisé :

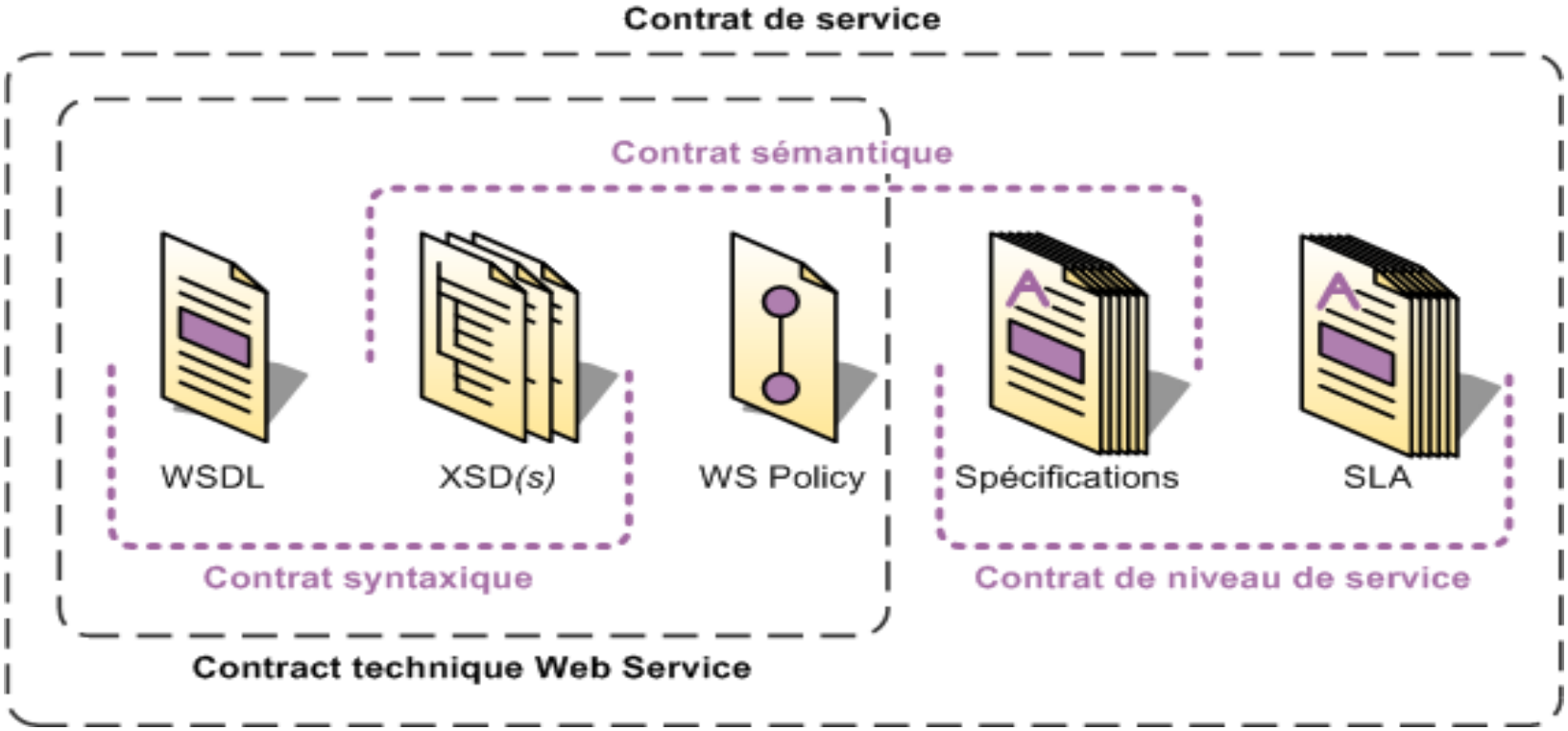


Document WSDL : décrivant l'interface du service (les modalités d'accès).

Ensemble de schémas XML (X shema Defintion): qui définissent les types de données échangées.

4- CARACTÉRISTIQUES :

1- Contrat standardisé :



Ensemble de Policies : définissent les règles d'utilisation du service .

Documentations complémentaires : complètent le contrat sémantique.

Service Level Agreement (SLA): Fixe les limites de performance (disponibilité, temps de réponse).

4- CARACTÉRISTIQUES :

1- Contrat standardisé :

Standardisation du contrat :

Le contrat de service est une notion complexe qui ne se limite pas à la (simple) définition d'interfaces.

Un contrat de service est constitué de nombreux éléments (techniques ou non) qui forment un fond documentaire pour lequel il est préférable (voire indispensable) de respecter un formalisme commun.

L'utilisation de ce formalisme commun est le meilleur moyen de construire un modèle cohérent et donc facile à comprendre et à partager

4- CARACTÉRISTIQUES :

1- Contrat standardisé :

Standardisation du contrat :

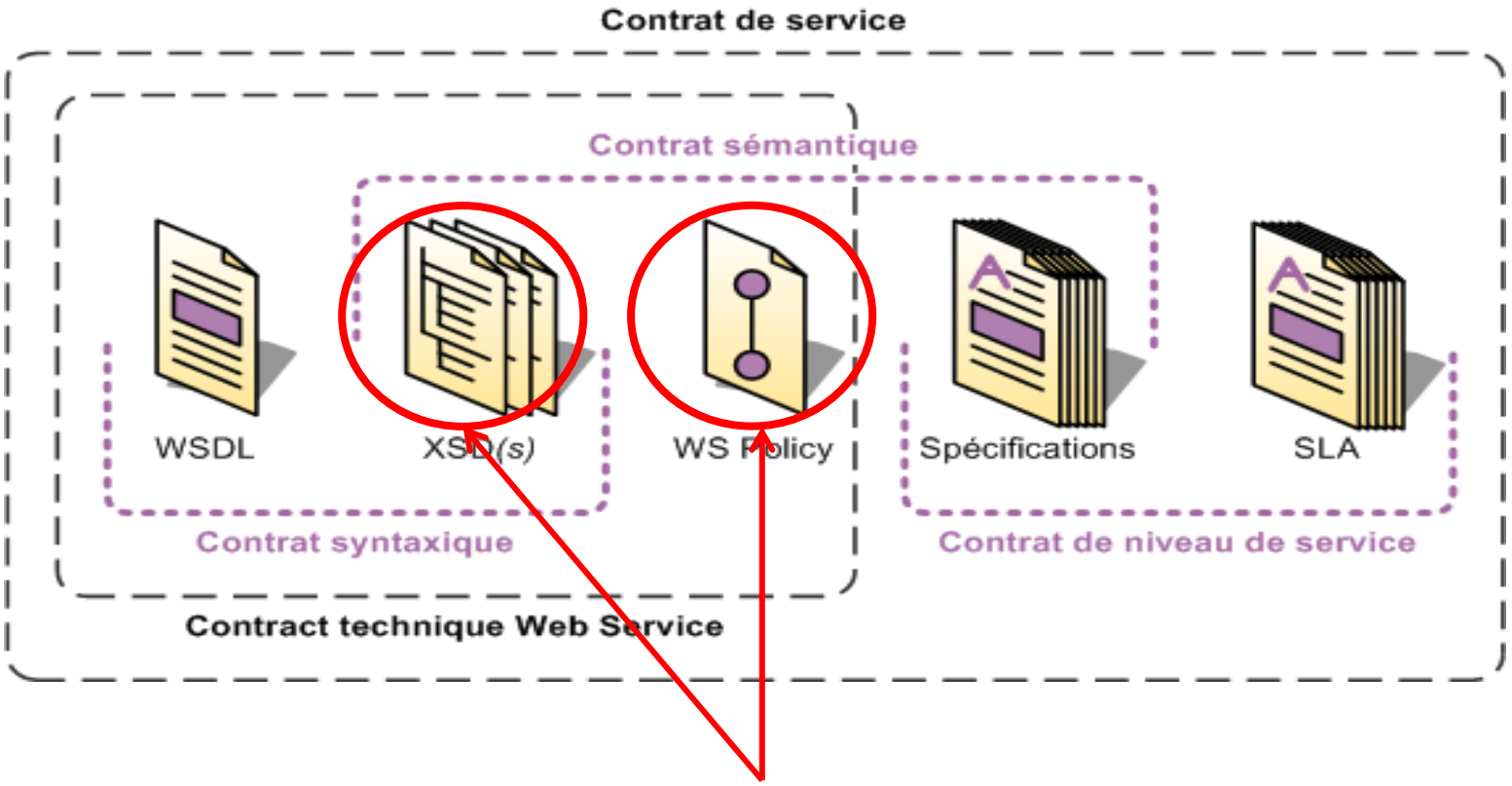
La standardisation des éléments du contrat permet :

- ✓ Cadrer la définition des contrats (faciliter la communication).
- ✓ l'utilisation d'un formalisme commun et de règles de standardisations facilite la centralisation des éléments constituant les contrats. Ce qui encourage la réutilisation (XSD & Policies).
- ✓ C'est le meilleur moyen de construire un modèle cohérent et donc facile à comprendre et à partager.

4- CARACTÉRISTIQUES :

1- Contrat standardisé :

Standardisation du contrat :



Des bons candidats à la réutilisation vu qu'on général les mêmes règles d'utilisations et les mêmes structures de données sont partagées entre les services dans un seul système.

4- CARACTÉRISTIQUES :

2- Couplage faible (lâche) :

L'objectif du **couplage lâche** est de **réduire les dépendances entre le consommateur (client) et le fournisseur (service)** afin de permettre l'évolution du système d'information sans impacts majeurs.

- **Pourquoi éviter le couplage fort ?**

Dans une architecture distribuée, deux types de couplage peuvent poser problème :

1) Couplage technique:

Le consommateur doit connaître le **protocole technique** (SOAP, REST) utilisé par le fournisseur.
Problème : toute évolution technologique du service impacte les clients.
Exemple : Si un service passe de SOAP à REST, et que les clients doivent être modifiés, cela signifie qu'ils étaient **fortement couplés** au protocole.

2) Couplage fonctionnel :

Le consommateur dépend du **format des données (**JSON, XML, CSV) **et des structures internes** du fournisseur.
Problème : toute modification du service entraîne des modifications chez tous les consommateurs..

4- Caractéristiques :

2- Couplage faible (lâche) :

- **Les dépendances inévitables :** Un service reste naturellement dépendant :
  - ✓ **De son implémentation :**
    Le service dépend de la façon dont il a été programmé (langage, architecture, choix techniques). Si l'implémentation change, le comportement ou les performances peuvent changer.
  - ✓ **Des ressources techniques utilisées :**
    Il dépend des serveurs, bases de données, réseaux ou matériels nécessaires à son fonctionnement. Si ces ressources sont indisponibles ou limitées, le service est impacté.
  - ✓ **Des technologies sous-jacentes :**
    Il repose sur des frameworks, systèmes d'exploitation, plateformes ou outils spécifiques. Une mise à jour ou une incompatibilité peut affecter le service.
  - ✓ **Des services qu'il compose :**
    S'il utilise d'autres services (API, microservices), il dépend de leur disponibilité et de leur bon fonctionnement.
  - ✓ **Des processus métier :**
    Il est lié aux règles et objectifs de l'entreprise. Si les besoins métier changent, le service doit évoluer pour rester adapté.

- Ce qu'il faut réduire, c'est la dépendance entre **le consommateur et la logique interne du service**.

4- CARACTÉRISTIQUES :

2- Couplage faible (lâche) :

- **Le rôle central du contrat :**

Le principe clé du couplage lâche est que le consommateur doit dépendre uniquement du **contrat du service**, et jamais de son implémentation.

- Deux approches existent :

❌ **Contract Last (à éviter)**

- ✓ Le contrat est généré à partir de l'implémentation.
- ✓ Le contrat devient dépendant de la technologie utilisée.
- ✓ Cela crée un **couplage fort**.

✅ **Contract First (à privilégier)**

- ✓ Le contrat est défini **avant** l'implémentation.
- ✓ Le service s'adapte au contrat.
- ✓ Le consommateur dépend uniquement du contrat.
- ✓ C'est un **couplage positif et maîtrisé**.

4- Caractéristiques :

2- Couplage faible (lâche) :

- Un système est en **couplage lâche** lorsque :
  - ✓ Le consommateur connaît uniquement le **contrat**
  - ✓ Le contrat est indépendant de l'implémentation
  - ✓ Les changements internes du service n'impactent pas les clients

- Le couplage lâche permet :
  - ✓ L'évolution technologique du service sans impacter les consommateurs
  - ✓ Une meilleure maintenabilité
  - ✓ Une plus grande flexibilité du SI
  - ✓ Une réduction des coûts d'évolution

4- CARACTÉRISTIQUES :

3- Abstraction :

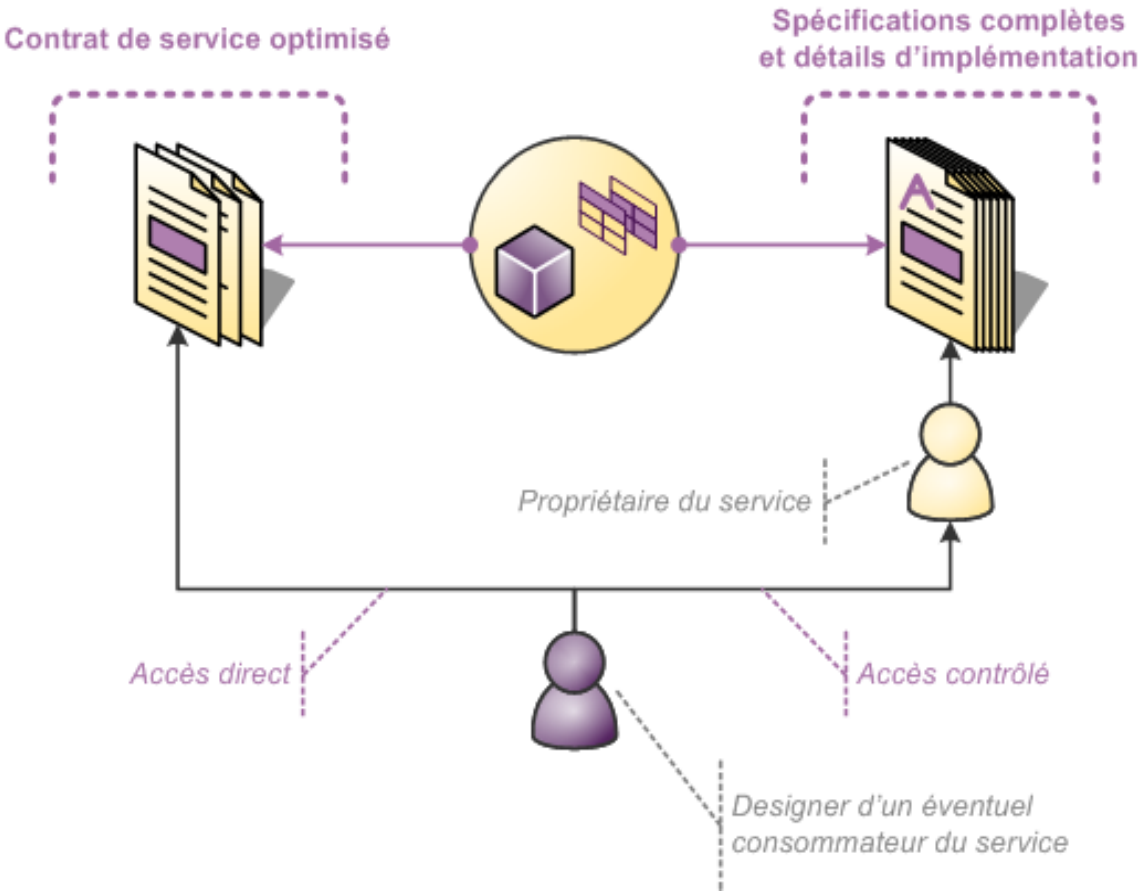
- Le principe d'abstraction consiste à fournir les services du SI sur un modèle ***boîte noire.***

- Les seules informations fournies au consommateur sont celles contenues dans son contrat, ainsi il est quasiment indispensable de n'y mettre que les informations essentielles à son invocation (pas de détails d'implémentation).

- Le fournisseur du service peut permettre un accès contrôlé à ce genre de détails à des concepteurs particuliers de consommateurs.

4- CARACTÉRISTIQUES :

3- Abstraction :



4- CARACTÉRISTIQUES :

3- Abstraction :

- Cependant cette restriction d'accès aux informations du service ne doit pas violer le principe de la prédictibilité du service.

- La ***prédictibilité*** d'un service signifie que son comportement et la réponse qu'il donne suite à une requête ne doivent pas varier (suite à une concurrence d'accès par exemple). (*Respect du contrat du service*).

- Ainsi le principe de l'abstraction de service signifie qu'il faut mettre dans le contrat du service toutes les informations et uniquement les informations nécessaires et suffisantes à son invocation.

4- CARACTÉRISTIQUES :

4- Réutilisabilité :

- La réutilisabilité d'un service désigne la capacité du service à répondre aux besoins de plusieurs types de consommateurs (est-ce que plusieurs consommateurs vont utiliser ce service ?).

- Les services identifiés comme réutilisables sont ainsi mis à la disposition d'un large spectre de services composites, de processus, d'applications, …

Les principaux freins qui peuvent influencer mal sur le degré de réutilisation de service sont :

a-La centralisation et la standardisation des ressources

1) Centralisation des ressources :
regrouper les éléments communs du système (logique métier, structures de données, contrats de services, politiques, etc.) dans un **ensemble de services partagés**, accessibles par plusieurs applications ou consommateurs.

2) Standardisation des ressources :
définir et appliquer **des normes communes** pour la conception et l'utilisation des services (formats de données, conventions de nommage, structures des contrats, protocoles, règles d'utilisation).

4- CARACTÉRISTIQUES :

4- Réutilisabilité :

Les principaux freins qui peuvent influencer mal sur le degré de réutilisation de service sont :

b-La démarche suivie dans la conception de service :

- **Approche Bottom-Up:**
  - ✓ On part de l'**existant technique** (applications, bases de données, composants) pour construire des services.
  - ✓ On expose des fonctionnalités déjà implémentées sous forme de services.
  - ✓ Logique **:** Technique ➜ Service ➜ Processus métier

- **Approche Top-Down**
  - ✓ On part d'une **analyse métier globale**, puis on identifie les services nécessaires.
  - ✓ On conçoit les services à partir des processus métier.
  - ✓ Logique **:**Processus métier ➜ Services ➜ Implémentation technique

4- CARACTÉRISTIQUES :

4- Réutilisabilité :

Les principaux freins qui peuvent influencer mal sur le degré de réutilisation de service sont :

c- La granularité du service :

- La granularité d'un service est définie par le nombre de fonctions recouvertes par celui-ci.

Mauvaise granularité : on parle de mauvaise granularité de service lorsque celui-ci couvre trop ou trop peu de fonctionnalités :

Un service qui fait trop de choses causera des mauvaises performances et risquerait d'être moins réutilisable à cause des fonctionnalités non utiles pour le consommateur.

Un service qui fait trop peu de choses sera sans valeurs ajoutée métier et risque d'être non réutilisable de son besoin à collaborer avec d'autres services pour répondre aux besoins du consommateur.

- En effet ce facteur est très critique, et une mauvaise granularité du service va surement limiter son réutilisation

4- CARACTÉRISTIQUES :

4- Réutilisabilité :

Les principaux freins qui peuvent influencer mal sur le degré de réutilisation de service sont :

d- La résistance au changement :

- Dans certaines organisations, ce sont les **urbanistes SI** qui contrôlent les relations entre services.
- Les problèmes de **propriété des composants** (chaque équipe possède "son" service) limitent le partage.
- Le **financement par projet** pousse les équipes à développer leurs propres solutions au lieu de réutiliser l'existant.
- Résultat :
  - ✓ duplication des services
  - ✓ faible mutualisation
  - ✓ réutilisation théorique mais peu appliquée

4- Caractéristiques :

5- Autonomie :

- L'**autonomie** signifie qu'un service est capable d'exécuter sa fonction **de manière indépendante**, sans dépendre fortement d'autres services ou du contexte dans lequel il est invoqué.
- Son objectif est de garantir :
  - ✓ La **prédictibilité** du service,
  - ✓ La **fiabilité,**
  - ✓ La **stabilité des performances,**
  - ✓ Un meilleur fonctionnement en accès concurrents
- Son principe est que plus un service exerce un **contrôle fort sur son environnement d'exécution**, plus il est autonome et prédictible. Il doit limiter :
  - ✓ Le partage de ressources
  - ✓ Les dépendances externes
  - ✓ Les interactions inutiles avec d'autres services

4- Caractéristiques :

5- Autonomie :

On distingue principalement **deux niveaux:**

a) Autonomie de niveau service (autonomie partielle)

- Le service est **logiquement indépendant**, mais il peut encore partager certaines ressources techniques.
- **Elle est caractérisé par :**
  - ✓ Frontières du service clairement définies
  - ✓ Indépendance fonctionnelle vis-à-vis des autres services
  - ✓ Peut partager(Base de données, Serveur d'application, Infrastructure technique

b) Autonomie pure (autonomie complète)

- Le service contrôle **entièrement** :
  - ✓ Sa logique métier
  - ✓ Ses données
  - ✓ Ses ressources techniques
- Il ne partage rien de critique avec d'autres services.
- **Elle est caractérisé par :**
  - ✓ Contrôle exclusif de ses ressources
  - ✓ Pas de dépendance critique externe
  - ✓ Exécution hautement prédictible

4- CARACTÉRISTIQUES :

6- Sans état (stateless):

- Un service est **sans état** lorsqu'il **ne conserve pas d'informations** sur les requêtes précédentes d'un client.
- Chaque requête est traitée **indépendamment** des autres.
- Elle est importante car, Être sans état permet de garantir :
  - ✓ La **prédictibilité** du service
  - ✓ Une meilleure gestion des accès concurrents
  - ✓ Une meilleure **scalabilité**
  - ✓ Une consommation réduite de ressource

4- CARACTÉRISTIQUES :

6- Sans état (stateless):

- **Problèmes liés aux services avec état :**

1 Complexité

- ✓ Augmente la complexité de l'implémentation
- ✓ Rend la lecture, les tests et la maintenance plus difficiles

2 Réutilisabilité réduite

- ✓ Le comportement dépend du contexte interne
- ✓ Ce comportement n'apparaît pas clairement dans le contrat
- ✓ Rend la composition plus difficile

3 Performance

- ✓ Stockage d'informations en mémoire ou disque
- ✓ Consommation supplémentaire de ressources

4- CARACTÉRISTIQUES :

7- Découvrabilité :

- La découvrabilité désigne la capacité d'un service à être **identifié, localisé et compris** afin de favoriser sa réutilisation dans le système d'information.
- Chaque service doit être accompagné d'un **ensemble de métadonnées** permettant de décrire ses caractéristiques et de faciliter sa recherche.
- La découverte des services s'effectue principalement **lors de la phase de conception**, car la découverte automatique à l'exécution (runtime) reste encore limitée.

4- Caractéristiques :

7- Découvrabilité :

- La découvrabilité repose généralement sur l'utilisation d'un **repository de services**, qui constitue un **inventaire centralisé** des services disponibles.
- Ce repository stocke les **artefacts associés aux services**, tels que : spécifications, contrats, SLA, policies, schémas XML, WSDL et interfaces.
- Grâce à ces informations, un **concepteur de services ou d'applications** peut rechercher un service existant, analyser ses métadonnées et vérifier s'il répond aux besoins fonctionnels et non fonctionnels.
- La découvrabilité facilite ainsi **l'identification et la réutilisation des services** au sein de l'architecture SOA.

4- CARACTÉRISTIQUES :

8- Composabilité :

- La composabilité désigne la capacité d'un service à **participer à la composition avec d'autres services** afin de construire des applications ou des processus métier plus complexes.
- Elle permet de **réutiliser les services existants** pour créer des services composites ou des processus de plus haut niveau.
- Ce principe repose sur la **décomposition de la logique métier** en plusieurs services réutilisables.
- La composabilité contribue à appliquer le principe de **séparation des préoccupations (Separation of Concerns)** dans l'architecture SOA.

4- CARACTÉRISTIQUES :

8- Composabilité :

- Elle nécessite de définir une **granularité appropriée des services** :
- Un service trop large limite la réutilisation car il regroupe plusieurs traitements spécifiques.
- Un service trop fin n'apporte pas une valeur métier suffisante.
- L'objectif est donc de concevoir des services **modulaires et réutilisables**, pouvant être combinés pour répondre à différents besoins métier.