

SOA : Architecture Logique : *Principes, structures et bonnes pratiques*

Auteur: Gilbert Raymond

gilbert.raymond@softeam.fr

Version 2.1

avril 2011

Softeam

21 avenue Victor Hugo

75016 Paris

Table des matières

1	Introduction	3
2	Principes et motivations.....	4
3	Les éléments de base de l'architecture	6
3.1	Composant de service	6
3.2	Bus d'entreprise	7
3.3	Contrat de service	8
3.4	Données d'échanges et données persistantes.....	9
4	Typologie et modèle en couches logiques.....	11
4.1	Composant Entité.....	13
4.2	SOA et Processus.....	14
4.3	Composant Processus	15
4.4	Composant Fonction	16
4.5	Composant Utilitaire et public	17
4.6	Composant Présentation.....	17
5	Démarche et identification.....	19
5.1	Démarche orienté processus	19
5.2	Démarche orientée donnée	20
5.3	Démarche orientée application.....	20
5.4	Gestion des versions	21
5.5	Spécification étendue et tests.....	22
6	Cartographie et modèles.....	24
6.1	Cartographie des services	24
6.2	Structure générale du système	24
6.3	Dépendance d'interfaces	25
7	Structuration du SI et application	26
7.1	Application composite.....	26
7.2	Visibilité et structuration du SI.....	27
8	Mise en œuvre.....	28
9	Glossaire	29



1 Introduction

L'architecture orientée service (SOA) s'est imposée aujourd'hui comme un thème majeur pour les systèmes d'information d'entreprise. Plus qu'une nouvelle technologie ou méthode, c'est la convergence de plusieurs approches existantes, et l'émergence d'un style d'architecture et de gouvernance de SI.

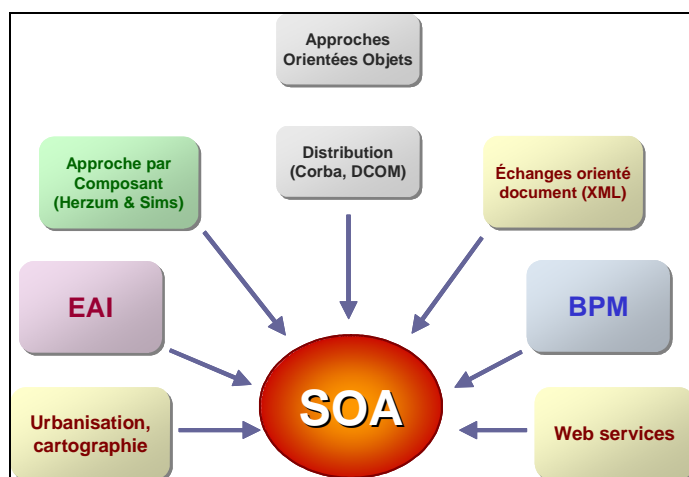


Figure 1 - SOA : à la convergence de solutions

Cependant, il ne s'agit pas d'une simple juxtaposition de techniques disparates. SOA intègre ces différentes pratiques dans un cadre organisé : l'architecture des systèmes, fortement guidée par le métier. En effet, les expériences trop focalisées sur une approche unique ou guidées par la technique n'ont pas apporté de réponses satisfaisantes.

Dans cette optique, l'architecture logique occupe une place centrale. Instrument privilégié pour la construction et la maintenance du système, pivot sur lequel s'articulent le métier et sa traduction logicielle, elle constitue la référence pour l'organisation des projets, la construction technique et le plan de progression. Toutefois, l'architecture logique n'est pas un modèle abstrait, ni une cartographie fonctionnelle cible lointaine. Il s'agit plus pragmatiquement de la description des constituants du système et de leurs relations.

Ce document présente l'architecture logique dans le cadre de notre approche SEA (Service Enterprise Architecture).

2 Principes et motivations

SOA (Service Oriented Architecture) est un style d'architecture organisé à partir de services métiers communs mutualisés pour un ensemble de lignes métiers ou d'applications.

SOA is an approach to designing software that dissolves business applications into separate "services" that can be used independent of the applications of which they're a part and computing platforms on which they run.

Jay DiMare, IBM Global Services, 2006

La motivation fondamentale vient du constat suivant : Le cloisonnement en silos applicatifs indépendants (blocs monolithiques) est une des sources majeures des difficultés rencontrées pour le traitement des évolutions et la maintenance des systèmes.

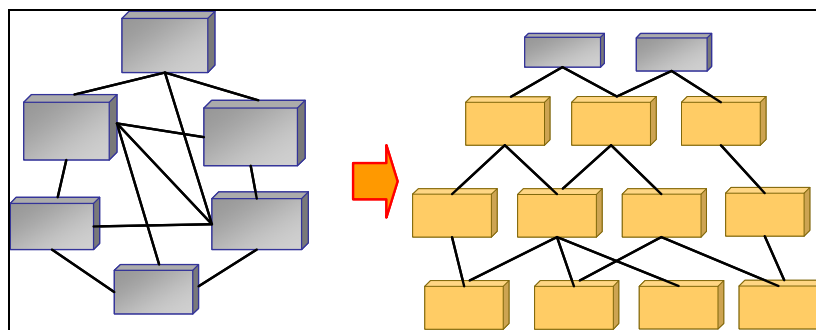


Figure 2 - Architecture orientée application vs architecture orientée service

A la recherche de l'agilité. Cette notion n'est pas nouvelle: Les Systèmes d'Information sont en constante évolution et les obstacles rencontrés sont largement débattus depuis des années, mais ces questions acquièrent aujourd'hui une acuité particulière. Les organisations sont confrontées à des demandes de changement toujours plus étendues et plus fréquentes. Ces changements sont liés à la fois aux réorganisations (fusion, acquisitions), à l'ouverture des processus et à la multiplication des évolutions des marchés: On l'observe par exemple dans le secteur des services de telecom, dans les banques assurances, le transport et l'énergie, ou dans le commerce en ligne. Les applications construites et structurées pour répondre à des besoins particuliers dans un contexte donné ne sont plus adaptées aux réalités présentes. Les changements technologiques ajoutent encore des contraintes supplémentaires. Aux delà d'un certain stade, les coûts induits par les modifications deviennent prohibitifs, et les délais incompatibles avec les demandes métiers. Le système devient trop rigide, prisonnier de son architecture antérieure. Les évolutions réalisées dans ce cadre détériorent la qualité du système et renforce encore ces difficultés.

Quelques constatations majeures :

- L'agilité d'un système dépend au premier ordre du système réel déployé. Les différentes représentations, modèles ou cartographies, si elles facilitent la tâche, et sont souvent nécessaires, n'éliminent pas la nature des obstacles rencontrés.
- Les infrastructures technologiques transverses (middleware, web services, ...) ne constituent jamais en soi de solutions radicales. On observe au contraire que les stratégies trop ancrées sur les technologies masquent les réalités métiers et peuvent aboutir à des résultats décevants et contre productifs.



- Les systèmes logiciels sont élaborés, développés par des équipes. Le travail humain reste largement majoritaire dans nos métiers et l'organisation, la gestion efficace des hommes est une des clés de la réussite.

L'architecture orientée service se base sur les principes suivants:

- **Diviser pour régner** : Substituer la découpe strictement applicative par une structuration en composants plus réduits et potentiellement plus simples à faire évoluer.
- **Alignement métier** : Construire et organiser le système à partir des réalités métiers, qui doivent se retrouver dans ses constituants.
- **Neutralité technologique** : Assurer une indépendance totale entre les interfaces et les implémentations. L'élément qui utilise un service ne doit pas être contraint ni par la technologie d'implémentation, ni par sa localisation (potentiellement distribué).
- **Mutualisation** : Favoriser la réutilisation de services métiers par plusieurs lignes métiers ou applications. Permettre la construction de services de haut niveau par combinaison de services existants.
- **Automatisation des processus métier**. Isoler la logique des processus métiers sur des composants dédiés qui prennent en charge les enchaînements de tâches et les échanges de flux d'information.
- **Echanges orientés Document**. Les informations échangées par les services possèdent une structure propre, guidée par les besoins métiers. On privilégie la transmission de contenus complets et utilisables au profit d'accès direct aux structures de type objet ou relationnel.

La plupart des acteurs SOA préconisent une mise en œuvre progressive, excluant les opérations de type « big bang », avec une cohabitation entre les constituants divers (legacy, anciennes applications, ERP etc...) et les services SOA.



3 Les éléments de base de l'architecture

3.1 Composant de service

Le *composant de service* est la brique de base de l'architecture [Figure 3]. Il se décompose en deux parties : *la vue externe* (ou spécification de service), qui expose la facette service proprement dite, et *la vue interne*, qui décrit le contenu du composant¹². La vue externe est constituée par un ensemble d'opérations de service regroupées en interfaces (au sens UML), et de l'appareillage pour les utiliser (types des données échangées, contrat de service, propriétés, etc....). Cette spécification est décrite en général par un fichier WSDL³ ou équivalent.

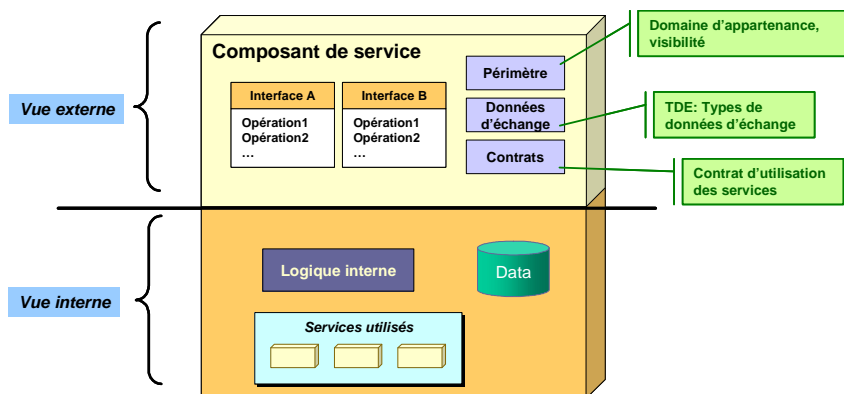


Figure 3 – Structure du composant de service

La vue interne contient des informations relatives à la logique interne comme le détail des traitements ou les bases de données manipulées. On y trouve également les références vers les services utilisés par le composant. Cette vue est masquée aux consommateurs du composant de service. Elle est employée notamment par les architectes SI, qui travaillent sur la vision globale du système.

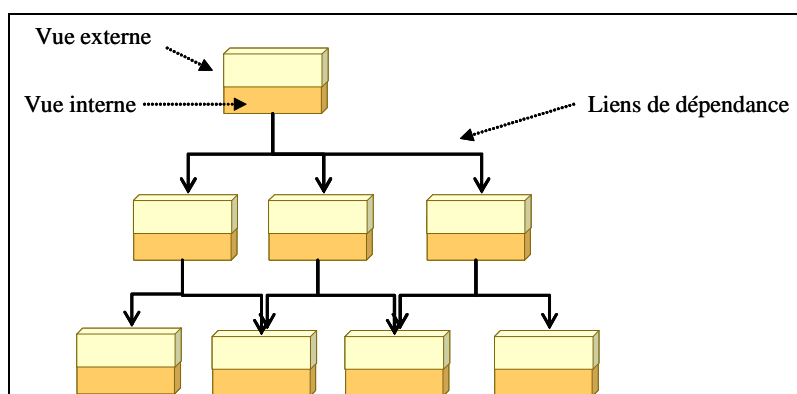


Figure 4 - Représentation schématique du système

¹ La notion de composant ou de composant de service se retrouve notamment dans SCA (Service Component Architecture) proposé par l'organisme Open SOA (www.osoa.org)

² On retrouve des notions similaires dans Togaf9® chapitre 22.7 (www.opengroup.org)

³ Web Services Description Language. www.w3.org



Schématiquement, l'articulation des composants de service, avec leurs liens de dépendance, constitue la structure du système [Figure 4]. Chaque composant de service peut invoquer les opérations de service d'autres composants, dans la mesure où les règles de visibilité sont respectées.

Composants et services. Dans cette approche, le *service* désigne le point de vue du consommateur, c'est à dire la vue externe du composant de service. Par exemple le *service* de gestion des commandes = vue externe du Composant « gestion des commandes ». Le catalogue des services publiés sur un périmètre est constitué par l'ensemble des vues externes des composants de service.

Les composants de service sont identifiés et définis avant tout suivant une logique métier, indépendamment de la technologie. On distingue pour ce faire le *composant logique*, qui fixe la structure du système dans le cadre de l'architecture logique, et le *composant logiciel*, qui est sa traduction technique déployée.

3.2 Bus d'entreprise

Le système est constitué d'un ensemble de *participants* communiquant qui jouent le rôle de *consommateur* et de *fournisseur* de service. Les consommateurs de service peuvent être divers : des applications, progiciels ou d'autres composants de service.

Les composants de service sont les fournisseurs de service du système, dans lesquels on distingue deux familles: les composants qui prennent en charge directement l'implémentation des services et ceux qui délèguent cette implémentation à un tiers (mainframe, ERP, application existante). Il faut néanmoins préciser que dans ce cas de figure, le composant n'est pas toujours assimilé à un simple passe-plat. Des traitements d'adaptation sont fréquemment nécessaires (format des données, encapsulation de service) afin de mieux intégrer les besoins des consommateurs de service.

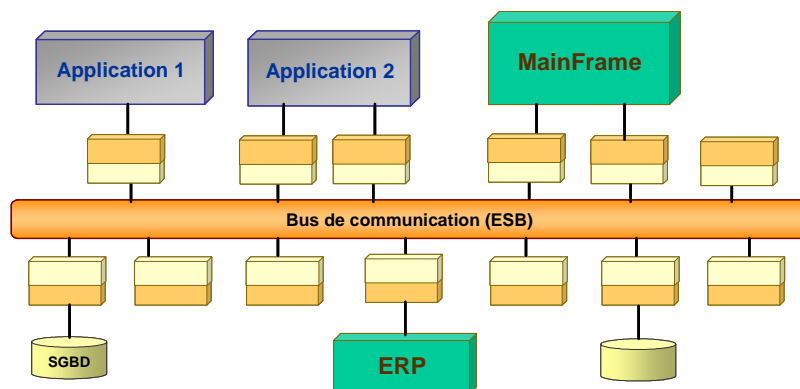


Figure 5 – ESB et Composants de service

Le bus d'entreprise (ESB) agit comme la colonne vertébrale reliant ces participants d'une manière banalisée à travers les interfaces de services. Cela permet notamment les modifications d'implémentation ou le remplacement des « legacy » sans remanier la structure de fonctionnement du système.

3.3 Contrat de service

Le contrat de service⁴ joue un rôle majeur : il détaille les conditions d'utilisation du service sous forme de pré et post conditions, protocoles, et contraintes (QoS, SLA⁵). Les contraintes non fonctionnelles [Tableau 1] permettent de fixer les termes du contrat opérationnel entre consommateur et fournisseur de service.

Tableau 1 - Contraintes non fonctionnelles

Type de contraintes	Exemples
Disponibilité	Taux d'indisponibilité par an, Plages horaires Contraintes de maintenance
Performance	Temps de réponse moyen, minimum Nombre de transactions par seconde
Fiabilité	Taux d'erreur
Sécurité	Politique de droits d'accès, Non répudiation
Administration	Journalisation, Tableaux de bord, statistiques

Le protocole d'utilisation d'un composant de service définit les séquences valides d'invocation des ses opérations [Figure 6]. A noter que les composants de service ne sont pas des objets : ils ne conservent aucun état (une seule instance du composant est disponible). Les conditions portent sur des attributs des données d'échange transmises lors de l'invocation des opérations : dans l'exemple [Figure 6] l'état est un attribut de typefacture transmis en entrée de chaque opération du composant Facture.

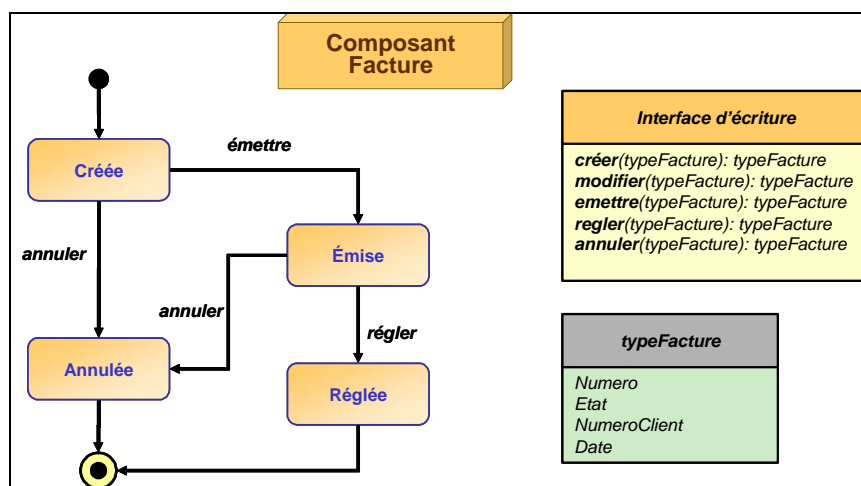


Figure 6 - protocole d'utilisation du composant Facture

Dans la même optique, les pré et post conditions détaillent encore les conditions d'utilisation sur les opérations de service⁶.

⁴ Voir notamment: Reference Model for Service Oriented Architecture. OASIS. www.oasis-open.org

⁵ QoS: Quality of Service. SLA: Service Level Agreement.

⁶ Reprise des méthodes de *programmation par contrat*. fr.wikipedia.org/wiki/Programmation_par_contrat

3.4 Données d'échanges et données persistantes

La distinction entre les *données d'échange* et les *données persistantes* est inhérente aux architectures SOA, qui isolent les bases de données à l'aide de services d'accès [Figure 7]. Les *données d'échange* sont les informations véhiculées entre les participants (consommateurs ou fournisseurs de service) à travers l'invocation des opérations de service. Les données persistantes sont les informations contenues et gérées dans les bases de données. Ces informations sont structurées de façon habituelle (par exemple SGBD en mode relationnel), dans le cadre de référentiels ou de bases applicatives.

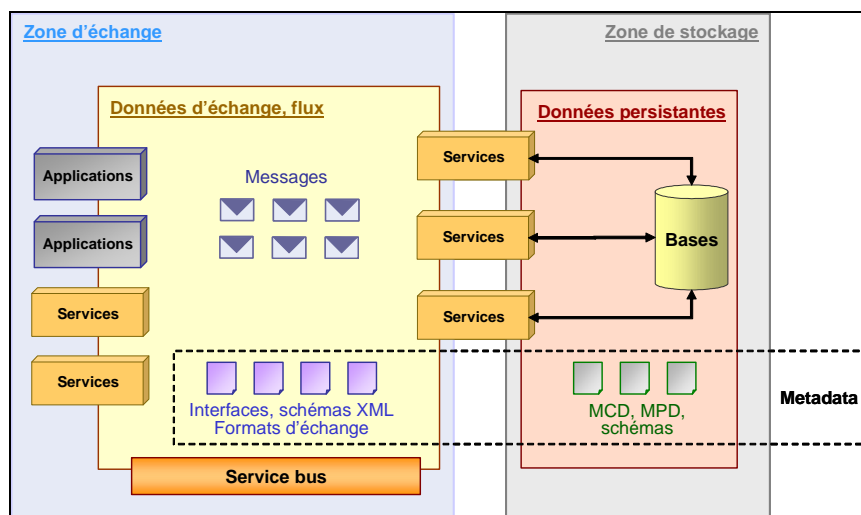


Figure 7 - Données d'échange et données persistantes

Les types de donnée d'échange (TDE) établissent la sémantique, la structure et le format de ces données. Ils peuvent être définis à l'aide de schémas XML (et éventuellement de classes UML). Chaque opération de service précise les types de donnée d'échange en entrée et en sortie.

Tableau 2 - Opérations de service: exemples

Opération de service	Entrée(s)	Sortie(s)
créerCommande	TDE_Commande	
validerCommande	TDE_Commande	TDE_EtatValidation

Les caractéristiques des TDE sont les suivantes :

On privilégie les messages « gros grain », qui regroupent un ensemble d'informations exploitables par le métier, et qui minimisent le nombre d'invocation d'opération de service [Figure 8].

La structure et l'organisation des TDE n'est pas contraint par des normalisations (entité-relation ou orienté objet). On parle de contenu orienté document, ou message, dans lequel on trouve une copie des éléments proche de la vision métier. (Par analogie avec le fonctionnement d'une messagerie, ou chaque message est constitué d'éléments dont la cohérence est validée par le propos du message lui-même).

```
<commande>
  <client>
    <nom> Durand </nom>
    <noContrat> 27615 </noContrat>
  </client>
  <montant> 522 </montant>
  <date> 15012007 </date>
  <ligneCommande>
    <noProduit> 432 </noProduit>
    <quantité> 3 </quantité>
  </ligneCommande>
  <ligneCommande>
    <noProduit> 603 </noProduit>
    <quantite> 1 </quantite>
  </ligneCommande>
</commande>
```

Figure 8 – Exemple de donnée d'échange "TDE_Commande" (format XML)

Dans le cadre SOA, la gouvernance des données intègre la gestion des données persistantes, des données d'échange et de leurs liens. La maîtrise de cette gestion est fondamentale et doit être traitée avec une attention particulière. En effet, le contenu et la structure des données d'échange sont en grande partie issus des données persistantes et le bon fonctionnement du système nécessite la description détaillée de ces relations.

4 Typologie et modèle en couches logiques

La mise en œuvre de services, aux travers de composants de service est-elle suffisante ? Force est de constater que les SI sont des systèmes complexes et souvent hétérogènes. La prolifération d'éléments, fussent ils des services métiers, s'échangeant toutes sortes de messages ne constitue pas une image très rassurante pour un DSI (on parle de la dérive Spaghetti Oriented Architecture).

JBOWS (Just a Bunch of Web Services). An effective, functioning service-oriented architecture requires governance, and the ability to share services across multiple business units and enterprises. It's easy to build Web services. You could build 10 of them in an afternoon. But, then you end up with a JBOWS architecture (Just a Bunch of Web Services), which will grow into a different sort of SOA — a Spaghetti-Oriented Architecture. Joe McKendrick , december 2006, ZD Net.

Il existe un consensus aujourd'hui pour bâtir les architectures de système à partir d'une typologie de services bien établie, organisée en couches logiques (Tableau 3). En général les aspects processus s'appuient sur des services plus basiques plus proches des données.

Tableau 3 - Les différentes propositions de typologies de services

Herzum & Sims ⁷	ESOA ⁸	Microsoft ⁹	IBM ¹⁰	Togaf ¹¹	Wikipedia ¹²	Types SEA
	Front end Application	Presentation Layer	Presentation		Presentation	Présentation
Processus	Process centric	Business Process	Business process choreography	Process services	Process	Processus
	Intermediary	Business Service	Composite service	Application services	Functionality	Fonction
Entité	Basic	Data Service	Service	Data services	Data	Entité
Utilitaire						Utilitaire
	Public					Public

Nous avons choisi pour SEA de distinguer 4 types de composant: *Présentation*, *Processus*, *Fonction*, *Entité*, organisés en 4 couches logiques de stabilité croissante, auxquels nous ajoutons les composants *Utilitaire* et *Public*, en charge des fonctions transverses et des échanges avec les systèmes externes.

⁷ Business Component Factory, Peter Herzum & Oliver Sims, Wiley Computing Publishing 2000

⁸ Enterprise SOA, Dirk Krafzig, Karl Banke, Dirk Slama, The Coad Series, 2005

⁹ An Overview of Service-Oriented Architecture in Retail, Moin Moinuddin, Microsoft, January 2007

¹⁰ Bernhard Borges, Kerrie Holley and Ali Arsanjani, IBM , 15 Sep 2004, SearchWebServices.com

¹¹ Togaf9 chapitre 22 (www.opengroup.org)

¹² "This is the type of the service to help distinguish it in the layer in which it resides: Data, Functionality, Process, and Presentation". http://en.wikipedia.org/wiki/Service-oriented_architecture



Les couches logiques de stabilité croissante établissent la règle de base de dépendance : un composant ne peut pas utiliser un composant d'une couche d'un niveau supérieur (par exemple, un composant Entité ne doit pas utiliser un composant Fonction ou Processus).

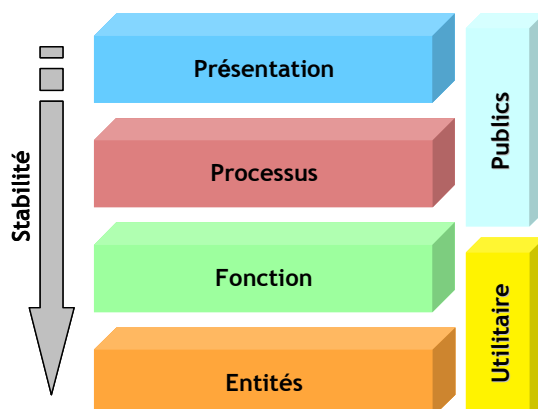


Figure 9 - Modèle en couches logiques

Chaque type de composant joue un rôle spécifique :

- Composant **“Présentation”** : Mise en oeuvre du dialogue avec l'utilisateur : IHM, gestion de la session utilisateur (Ce n'est pas un composant de service à proprement parler)
- Composant **“Processus”** : support de processus métiers complets (rôle d'orchestration); s'appuie notamment sur des composants de type “Fonction” et “Entité”
- Composant **“Fonction”** : Composition de services. Adaptations fonctionnelles ou traitements localisés.
- Composant **“Entité”** : Service d'accès aux données persistantes (CRUD¹³), aux bases de données et référentiels.
- Composant **“Utilitaire”** : fournisseur de services d'infrastructure ou transversaux (messagerie, tableau de bord, éditique, annuaire)
- Composant **“Public”** : dédiés aux services accessibles à l'extérieur du SI (B2B, partenaires)

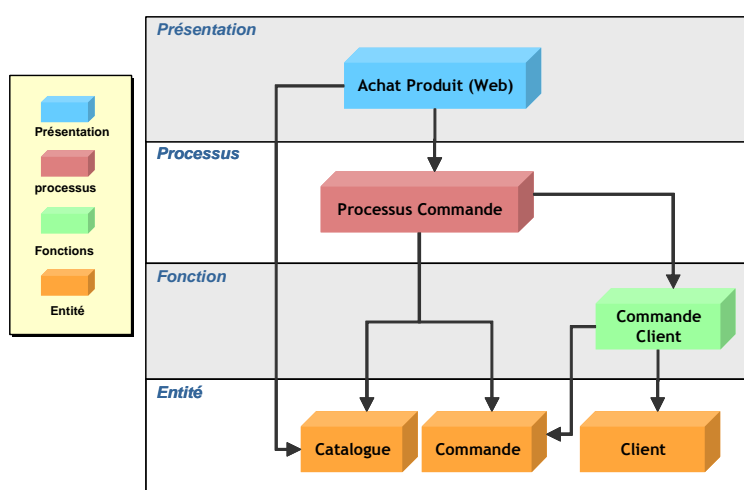


Figure 10 - Architecture de composants de services: exemple

¹³ CRUD : Create, Read, Update, Delete

4.1 Composant Entité

Les composants de service de type Entité sont focalisés sur un objet métier clé du système (par exemple Client, Contrat, Commande, ...). Leur rôle est de permettre un accès aux informations relatives à cet objet métier, le plus souvent associé à une base de données. On trouve typiquement les opérations de lecture, écriture ou de requête [Figure 11]. On impose que tout accès à un objet métier clé passe par le composant Entité correspondant qui est unique. Par exemple, la création, modification ou lecture d'un objet Client passe obligatoirement par les opérations du composant Entité Client.

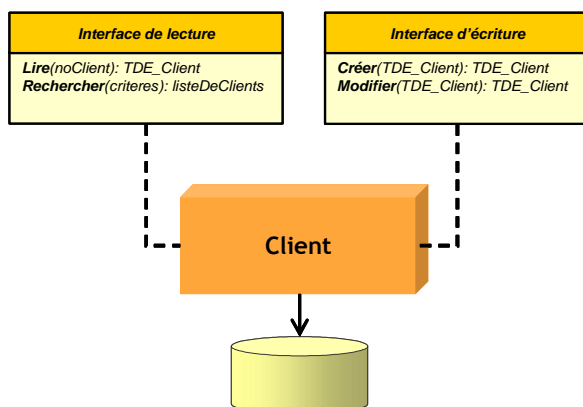


Figure 11 - Composant de service "Client" avec ses deux interfaces (lecture et écriture)

Les types de donnée d'échange (TDE) représentent la structure des flux échangés via les opérations de service (entrées et sorties des opérations). Plusieurs TDE sur le même objet peuvent être déclarés [Figure 12], en fonction du détail demandé ou du point de vue.

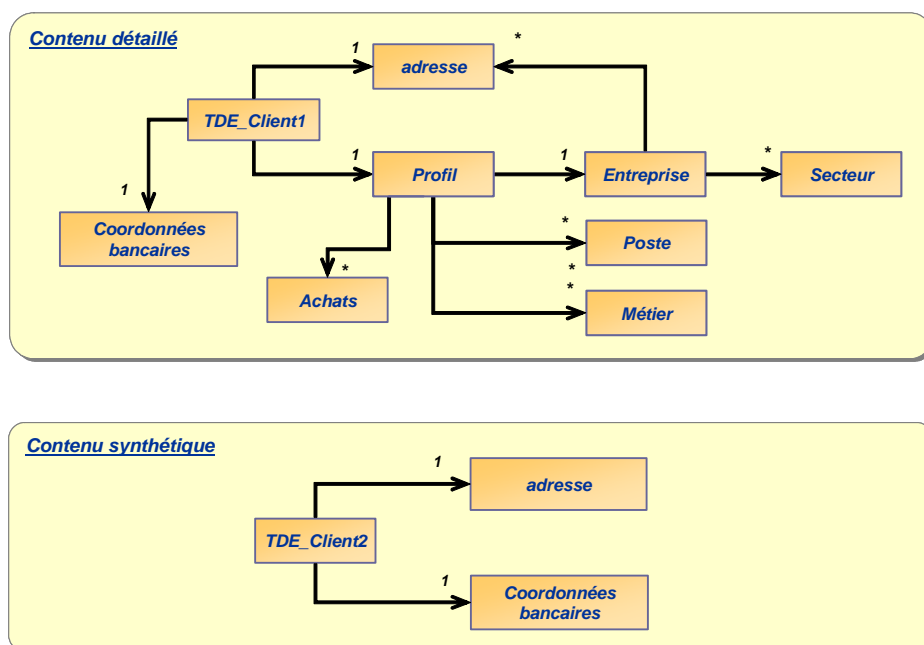


Figure 12 - Deux types de donnée d'échange (TDE) de l'objet métier Client

Il y a une relation étroite entre le composant Entité et le modèle des objets métiers: Pour chaque objet métier clé, on doit trouver un composant Entité correspondant. L'identification des objets métiers clé dépend du contexte métier et reprend les pratiques d'analyse existantes. Par exemple,

les objets « mineurs » ont peu de sens utilisés seuls (*adresse* du client). A l'inverse, les objets métiers clé interviennent directement dans les processus, sous forme de flux, ou sont les plus utilisés comme tels par l'utilisateur du système. Finalement, les messages échangés lors de l'exécution des opérations de service sont constitués par une grappe d'éléments dont la racine est un objet métier clé. Techniquement, ces messages sont généralement transmis sous la forme de documents XML, et les types de donnée d'échange (TDE) décrit par un schéma (XSD) correspondant.

4.2 SOA et Processus

Dans le cadre SOA, l'automatisation des processus est un axe majeur, avec les notions d'orchestration, composition de services ou de chorégraphie. Il s'agit de centraliser la logique d'un processus dans un composant dédié, qui prend en charge l'enchaînement et les règles de gestion associées [Figure 13]. Cette approche tend à réduire les impacts liés aux évolutions du processus.

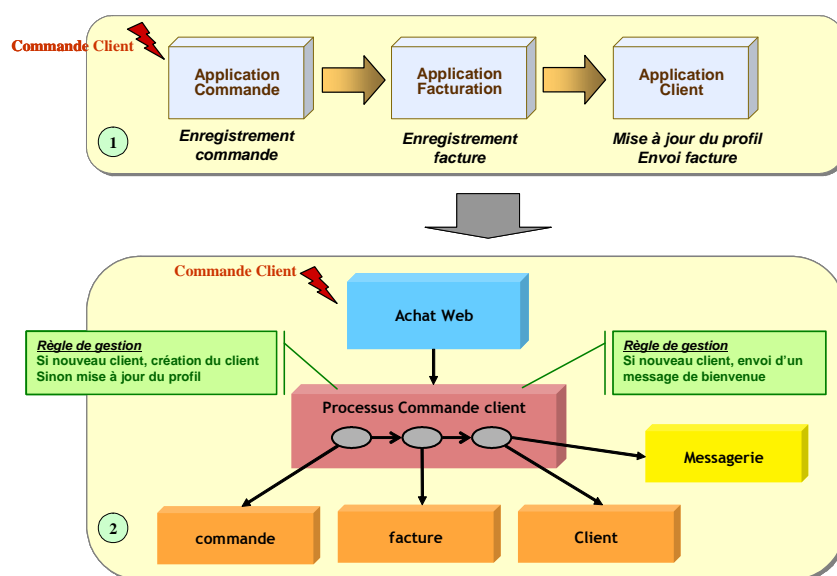


Figure 13 - Automatisation de processus. 1) orientée applications 2) orientée services

Une clarification est néanmoins nécessaire :

On distingue les *processus métiers transverses* et les *processus de traitement*. Les premiers représentent les processus de bout en bout de l'entreprise, qui délivrent une valeur ajoutée tangible à l'extérieur par une collaboration de plusieurs unités et acteurs. Les seconds (processus de traitement) représentent le déroulement d'une activité spécifique et localisée.

Pour simplifier on parlera de *processus métiers* et de *processus de traitement*.

On peut utiliser une identification plus formelle entre processus métier et processus de traitement : un *processus métier* est interruptible et possède un état qu'il doit conserver entre deux interruptions. A l'inverse, un *processus de traitement* n'est pas interruptible et ne maintient pas d'état en dehors de son exécution.

Un processus métier peut durer plusieurs jours, plusieurs mois ou plus : par exemple le traitement d'un sinistre d'assurance ou la livraison d'un produit commandé [Figure 14]. Le processus de traitement au contraire a une durée limitée et relève plus simplement d'une opération informatique habituelle (par exemple le contrôle et enregistrement d'un dossier, le destockage d'un produit).

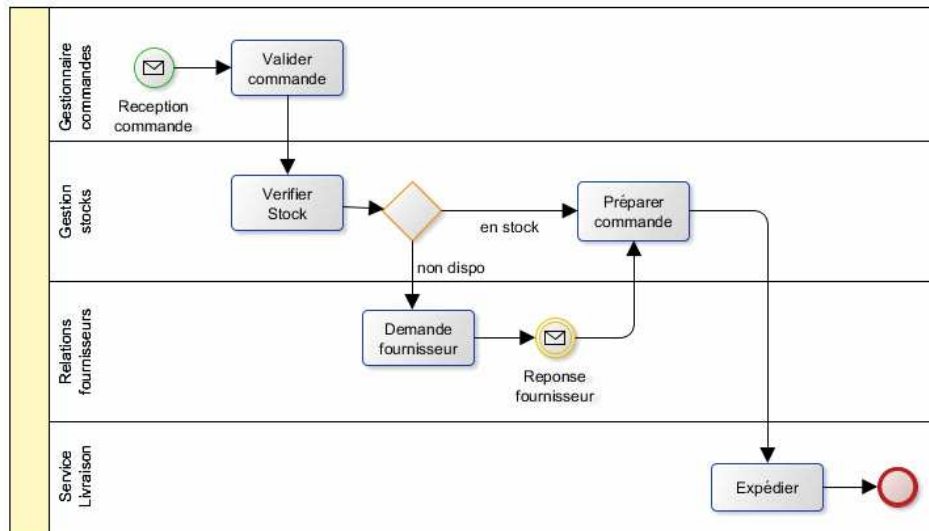





Figure 14 - Processus métier transverse : exemple (diagramme BPMN)

L’automatisation des processus métiers (transverses) est prise en charge par les composants de type Processus. Cette automatisation est appelée *Orchestration de service* (par analogie avec l’orchestre qui comprend un grand nombre de musiciens différents collaborant à l’exécution d’une symphonie sous le contrôle du chef d’orchestre).

Les composants de types Fonction sont en charge des processus de traitement. Ils jouent également le rôle d’adaptation et d’agrégation (composition de service) entre les Entités et les processus métiers ou la vision utilisateur.

Tableau 4 - Propriété des composants de service de type Processus, Fonctions et Entités

Type	Rôle	Type de participant	Granularité ¹⁴
 Processus	Processus métier transverse. Orchestration de service	Fournisseur et consommateur de service	Granularité élevée. Transverse par nature.
 Fonction	Processus de traitement, composition de services, adaptation	Fournisseur et consommateur de service	Granularité moyenne
 Entité	Accès à un objet métier clé	Fournisseur de service	Granularité fine. Focalisé sur un objet métier clé

4.3 Composant Processus

La distinction entre processus métier et processus de traitement n’est pas une simple question de vocabulaire. Le type de question posé par l’automatisation des processus métiers est tout à fait spécifique : la gestion des évènements et des états du processus, les actions de compensations¹⁵, le

¹⁴ La granularité est un indicateur informel lié au périmètre fonctionnel couvert par le composant.

¹⁵ Les actions de compensation définissent les traitements à réaliser en cas d’erreur ou d’exceptions dans le déroulement du processus, pour retrouver un état correct.



nombre et la diversité des services utilisés. On est dans le domaine du BPM¹⁶, avec ses techniques sous-jacentes comme BPEL, les méthodes de description adaptée (voir le langage BPMN), ou la supervision de processus (BAM).

Les opérations de service Processus sont liées aux événements du processus : démarrage, arrêt, ou spécifiques au métier.

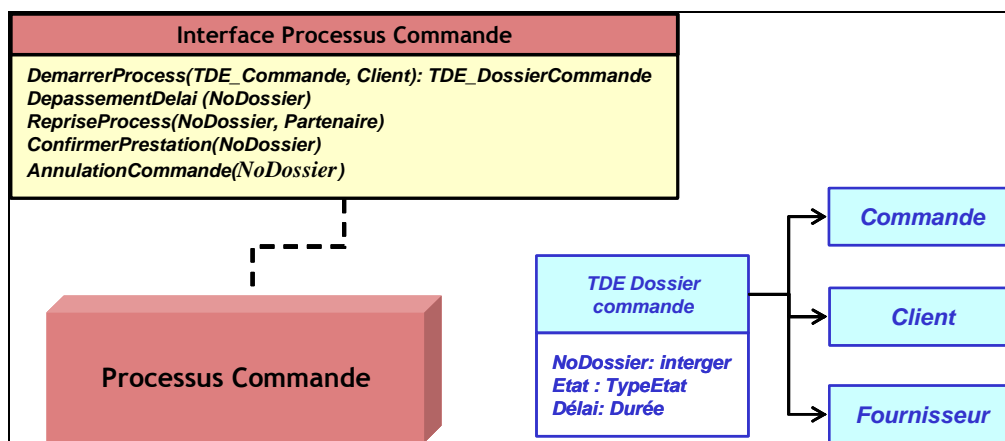


Figure 15 - Composant processus: interface et type de donnée d'échange

Le type de donnée d'échange du processus contient toutes les informations utiles ainsi que l'état du processus. Celui-ci est sauvegardé à chaque suspension du processus (attente d'évènement). Le processus devient un objet à part entière, avec un état propre, distinct de l'état des autres objets métiers (commande, facture). Cette distinction facilite le traitement des évolutions du processus métier, par le faible impact induit sur les autres composants. L'ajout d'une double validation dans le processus de commande va modifier le processus sans impact sur les états de l'objet commande.

Processus humains et workflow

Il faut noter que l'on peut distinguer deux types de processus automatisés : Les processus avec pas ou peu d'intervention humaine dans leur déroulement. C'est d'ailleurs un des objectifs de l'automatisation des processus : rationaliser et réduire le poids des tâches manuelles dans l'exécution des processus métiers.

Les processus métiers à forte intervention humaine (un grand nombre d'activités sont réalisées par des acteurs humains) relèvent historiquement des outils de workflow. Ces outils prennent en compte l'organisation des équipes, la transmission des informations et l'affectation des tâches entre les différents acteurs impliqués dans le processus. Cependant, il n'y a pas de frontière toujours nette entre ces deux domaines : dans la pratique, l'intervention humaine est souvent nécessaire pour traiter les situations exceptionnelles.

4.4 Composant Fonction

Les composants Fonction occupent une place charnière dans cette architecture. On a vu précédemment qu'ils se chargent des processus de traitement, mais plus généralement ils

¹⁶ BPM : Business Process Management. Gestion de processus métier. BPEL : Business Process Execution Language. BPMN : Business Process Modeling Notation (www.bpmn.org). BAM : Business Activity Monitoring.

fournissent les services proches de la vision utilisateur, par composition de services de type Entité. On retrouve cette notion de composition dans les TDE des services de type Fonction, qui sont souvent définis comme une agrégation de TDE de composants de type Entité [Figure 16].

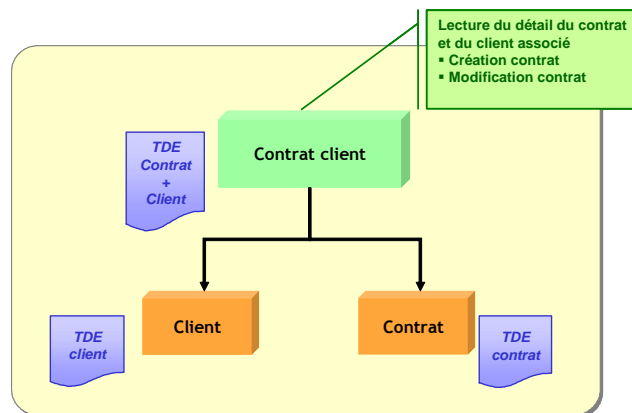


Figure 16 - Composant Fonction et agrégation de données d'échange

Dans la pratique, les composants Fonction sont les moins aisés à identifier. Les composants Processus proviennent directement des processus métier de l'entreprise, et les composants Entité des modèles d'information (objet métier ou base de données). Les composants Fonction sont mis en place graduellement par consolidations successives du système.

4.5 Composant Utilitaire et public

Les composants *Utilitaires* fournissent les services transverses, et relativement indépendants du métier de l'entreprise, comme les annuaires, la messagerie ou l'édition. Généralement stables, les composants utilitaires sont souvent implémentés par des progiciels largement diffusés, et sont peu risqués.

Les composants *Publics* sont dédiés aux échanges avec l'extérieur de l'entreprise (B2B e-Business). Le choix est de réserver ces communications à un type de composant particulier, compte tenu de ses particularités et des risques spécifiques. Typiquement, ces composants prennent en charge l'adaptation des formats de donnée (différence d'encodage, de structure ou de format), la sécurité, et tous les ajustements propres aux dialogues avec les partenaires.

Les composants *Publics* et *Utilitaires* ne sont pas associés à une couche logique particulière. Ils peuvent être en relation avec des composants de service indépendamment de leur type.

4.6 Composant Présentation

Les composants Présentation pilotent le dialogue entre le système et les acteurs externes. Ils assurent la gestion des interfaces homme machine et la maintenance du contexte session de l'utilisateur. Techniquement, ce type de composant utilise les infrastructures éprouvées (client riche ou client web, tiers de distribution).

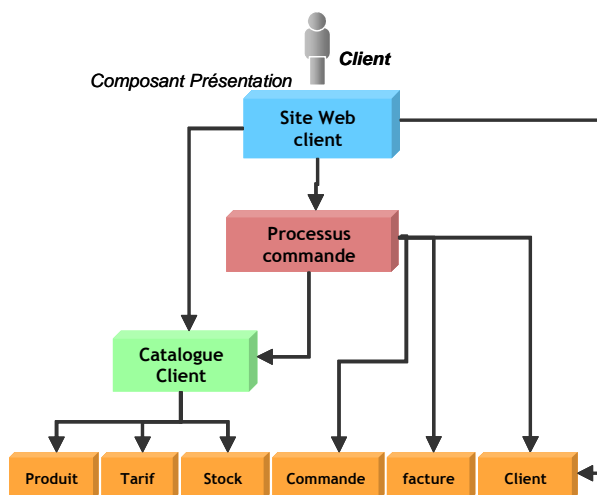


Figure 17 - Composant Présentation

Les composants Présentation ne sont pas des composants de service proprement dit : ils ne fournissent pas de services, sauf à l'utilisateur. Ils sont consommateurs de service pour tout autre type de composant. Dans le cadre d'une architecture SOA, les composants Présentation sont à la fois un point d'entrée sur le système et responsables de l'intégration des contenus pour l'utilisateur. Ils gèrent également le dialogue, l'évolution des données et leurs modifications au cours d'une session.

5 Démarche et identification

Il n'existe probablement pas de démarche universelle pour l'identification et la construction des services. Le contexte de l'entreprise, les méthodes ou les modèles d'urbanisation utilisés sont autant de facteurs qui devront être pris en compte. On peut néanmoins distinguer plusieurs démarches types :

- Démarche par processus métiers
- Démarche orientée données
- Démarche orientée applications

Naturellement, ces démarches types ne s'excluent pas (et ne sont pas exhaustives). Dans la réalité, le système se constitue par une articulation de ces différentes approches, par consolidations successives, en parallèle avec la vision globale de l'architecture.

5.1 Démarche orienté processus

La démarche orienté processus s'appuie sur une analyse des processus métiers de l'entreprise (ou d'un domaine particulier), dans l'objectif de déployer des composants de services de type Processus. Les principales activités sont les suivantes :

Cartographie des processus métiers : inventaire des processus métiers, avec leurs propriétés et leurs liens. Cette cartographie reste au niveau macro, sans détailler le déroulement de chacun des processus.

Identification des composants Processus : Détermination des processus métiers à automatiser. On privilégie les processus à forte valeur ajoutée pour l'entreprise (on évite les processus internes comme la gestion du personnel) et les plus évolutifs, de façon à aboutir à un réel apport métier et un retour sur investissement notable. A noter que certains processus peuvent être déjà automatisés, mais disséminés sur plusieurs applications. Dans ce cas, la reprise de la logique des ces processus par un composant dédié (type Processus) entraîne une rénovation des applications impactées qui facilitera la prise en compte des futures évolutions.

Elaboration des modèles de processus métiers. Ces modèles représentent l'enchaînement des processus métiers (activités, flux, acteurs) avec un point de vue maîtrise d'ouvrage. Les flux échangés par les activités établissent une première structure des types de données d'échange (TBE) utilisés. Ces modèles sont réalisés à l'aide de notations BPM ou de diagrammes d'activité UML [Figure 14].

Modèles de processus exécutable. Le passage de modèles métiers aux modèles exécutables nécessite la description détaillée de tous les éléments (l'ensemble des chemins, les erreurs, les compensations éventuelles), et de préciser les services consommés par de processus. Ce travail permet généralement d'identifier de nouveaux services ou d'ajouter des opérations aux services existants (notamment services de type Fonction ou Utilitaire). Les modèles sont de type BPEL, exécutables à l'aide d'un outil ou traduits dans un langage de programmation. Cette représentation sert de base pour la spécification des services, avec l'ensemble des opérations nécessaires.



5.2 Démarche orientée donnée

Cette démarche aboutit à la mise en place de composants de type Entité, avec la définition des types de donnée d'échange (TDE) associés. Elle assure un accès banalisé aux informations gérées par les bases de données.

Modèle des objets métier. Ce modèle représente la structure et les propriétés des objets métiers. Typiquement, on utilise le formalisme de classes UML, en s'appuyant sur les structures propres des bases de données existantes.

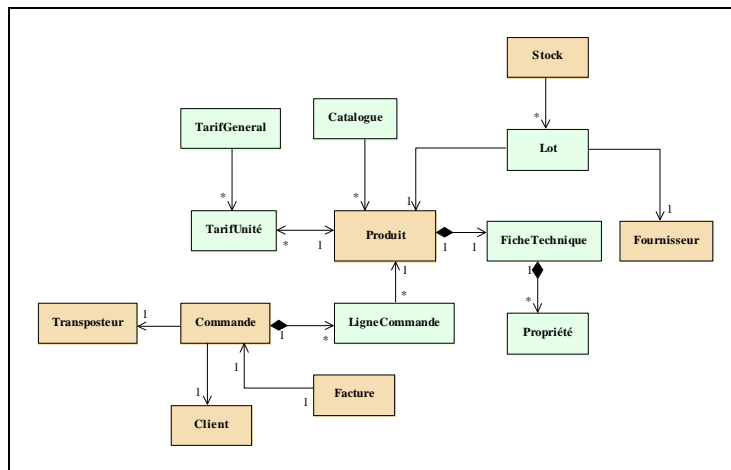


Figure 18 - Modèle des objets métier (notation UML)

Identification des objets métiers clé. Comme il a déjà été évoqué précédemment, cette identification s'appuie fortement sur la connaissance fonctionnelle. Chaque objet métier clé donne lieu à un composant de service de type Entité, qui fournit les services d'accès (création, modification, suppression et requête).

Définition des TDE. Pour chaque objet métier clé (et donc pour chaque composant de type Entité), un ou plusieurs formats d'échange est défini. Ce format est naturellement basé sur les modèles des objets métiers, et consiste à « découper » celui-ci sous la forme de format de type document (schéma XML), qui seront échangés lors de l'invocation des opérations du service [17].

Branchement des composants. Par leur nature particulière, les composants de type Entité obligent à une modification dans les accès à l'information. En effet, ces composants sont par définition d'unique canal vers les données persistantes : les autres formes d'accès doivent donc être remaniées (par exemple les lectures directes vers les bases de données).

5.3 Démarche orientée application

L'objectif est la restructuration de certaines applications par mutualisation de services.

Cartographie des applications. C'est un modèle des applications existantes intégrant notamment les échanges de flux inter applicatifs.

¹⁷ Voir notamment Enterprise Oriented Architecture, James McGovern, Oliver Sims, Ashish Jain, Mark Little, Springer, 2006, chapitre 2.

Identification des services mutualisés. L'analyse des flux et leurs caractéristiques (volume, fréquence) signale les applications fortement dépendantes. De plus, une connaissance plus détaillée des applications peut faire apparaître des redondances fonctionnelles ou des duplications historiques. Ces éléments se conjuguent pour isoler les services potentiellement mutualisables, et d'obtenir une découpe plus simple et plus cohérente.

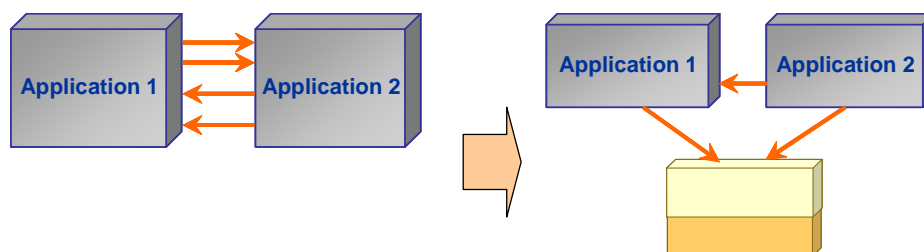


Figure 19 - restructuration par mutualisation de service

Restructuration. Les composants de services doivent apparaître comme des éléments stables, avec un consensus d'interprétation sans entraîner une refonte complète du domaine. Dans le cas contraire, une reconstruction globale est souvent préférable, avec une mise à plat de l'analyse métier. Les composants de services résultants sont généralement de type Fonction, proches de l'utilisation métier.

Remarque. La mutualisation de service n'est pas un objectif en soi. Il est inutile de dépenser du temps (et du budget !) dans la restructuration d'applications stables qui fonctionnent correctement.

5.4 Gestion des versions

Comme tout élément logiciel, les composants de services sont soumis aux évolutions (maintenance ou modifications fonctionnelles). La mutualisation de service a des impacts plus ou moins importants sur les consommateurs de ces services, en fonction du type d'évolution [Figure 20] : l'ajout d'une nouvelle interface ou d'une opération, la modification de l'implémentation (sans modification des interfaces de service), la modification d'interface existante, la modification des types de donnée d'échange.

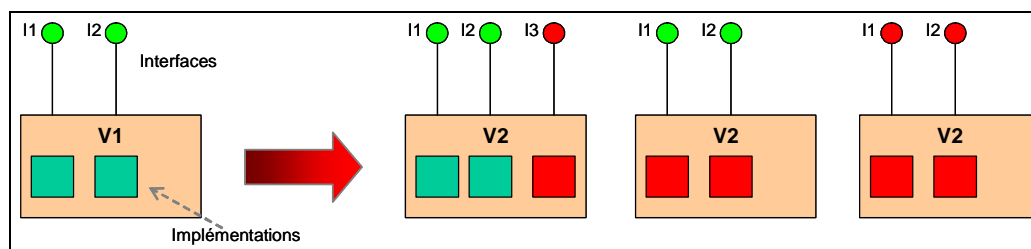


Figure 20 - Types d'évolution: Ajout d'interface, modification d'implémentation, modification d'interface

En toute rigueur, le déploiement d'une nouvelle version d'un composant de service entraîne un coût (et un délai) supplémentaire dû en particulier aux tests nécessaires des différents consommateurs du service. Cette contrainte peut dans certains cas devenir contradictoire avec la souplesse recherchée, et un déploiement de plusieurs versions d'un même composant est une solution envisageable.

Par exemple [Figure 21], si 2 applications A et B utilisent le même service, et que pour les besoins de l'application B le service est modifié, la coexistence de deux versions du service permet le

déploiement rapide de la nouvelle version sans impact sur l'application A. Le bus de communication prend en charge le routage en fonction du consommateur de service.

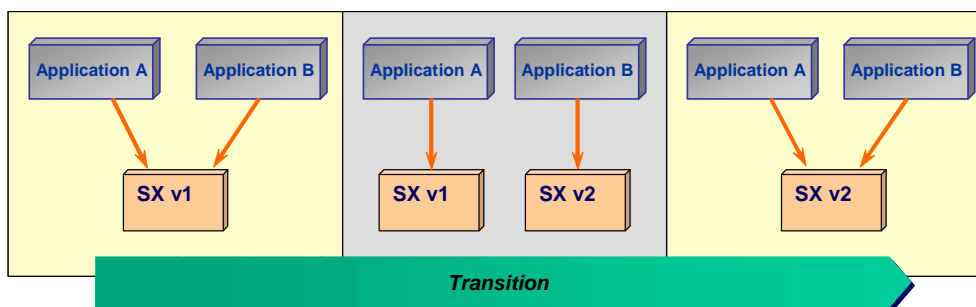


Figure 21 - Transition utilisant plusieurs versions d'un service

Cependant, on évitera la multiplication de ce genre de situation, qui devra conserver son caractère transitoire. En effet, la prolifération de versions de service engendre des difficultés de gestion qui annulent à terme ses effets bénéfiques. L'utilisation de règles appropriées permet de réduire ce type de risque, par exemple :

- Le nombre de composants de service déployés en plusieurs versions ne doit pas dépasser 15% du nombre total des composants.
- Pour un composant de service donné, le nombre de versions déployées est au maximum de 3

5.5 Spécification étendue et tests

La spécification de service a été définie plus haut avec ces différents éléments : interfaces et opérations, contrats et protocoles, types de donnée d'échange. En considérant le composant de service comme un mini progiciel du point de vue des consommateurs de service, les éléments de test sont partis prenante de sa spécification. Certaines méthodes comme Extreme Programming¹⁸ assimilent d'ailleurs spécifications et tests externes. Sans reprendre totalement cette approche, la vision pragmatique qui consiste à définir un élément logiciel par sa capacité à répondre à un ensemble de scénarios déterminés n'est pas nouvelle et est à la base des opérations de recettes.

Dans cette optique, la livraison d'un composant de service intègre tous les éléments qui participent à sa validation : les scénarios d'utilisation externes, les exemplaires de données d'échange et données persistantes, les émulateurs éventuels (bouchons) qui permettent la mise en œuvre de ces tests.

¹⁸ Extreme Programming Explained, Kent Beck, Addison-Wesley, 1999

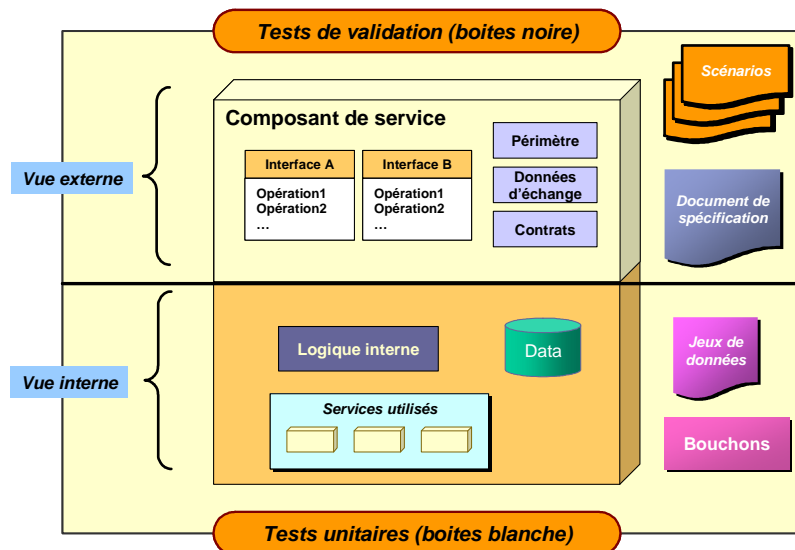


Figure 22 - Composant de service: spécification étendue

Au-delà du strict point de vue de validation, qui facilite les travaux d'intégration, les scénarios de test ajoutent une dimension concrète et exemplaire qui participe à la compréhension du rôle du composant. On retrouve cette approche avec les cas d'utilisation UML, qui précisent à l'aide de scénarios types, la dynamique d'utilisation et les résultats attendus.

L'ensemble de ces éléments disponibles pour le consommateur de service est appelé *spécification étendue*, qui pourra inclure également tous les documents et éléments utiles à la définition du service.

6 Cartographie et modèles

6.1 Cartographie des services

La cartographie des services est une vue synthétique des composants de services et de leur répartition dans les différentes couches logiques (Présentation, Processus, Fonction, Entité).

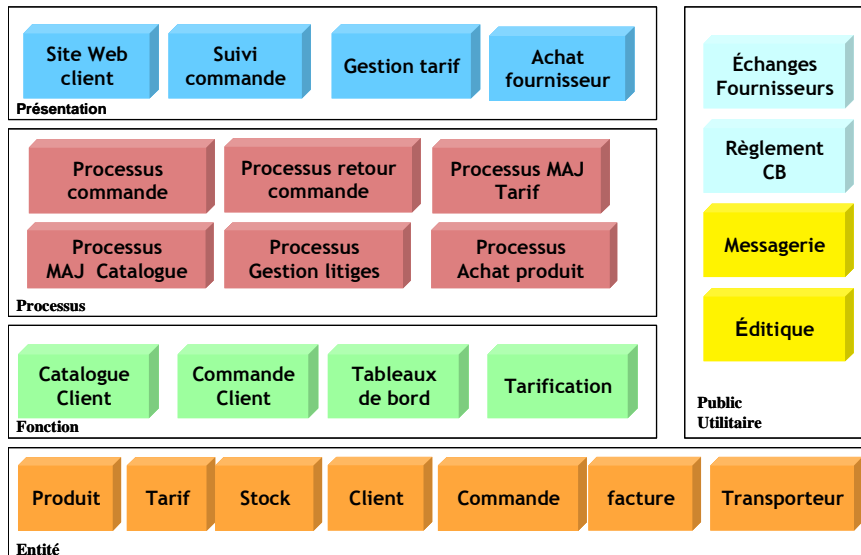


Figure 23 - Cartographie générale des composants de service

6.2 Structure générale du système

Le système contient naturellement tous les constituants du système. Dans la pratique, on va utiliser différentes vues (diagrammes) en fonction du type de tâche ou du niveau de détail désiré. La description des dépendances (liens d'utilisation) est particulièrement importante pour une analyse du système.

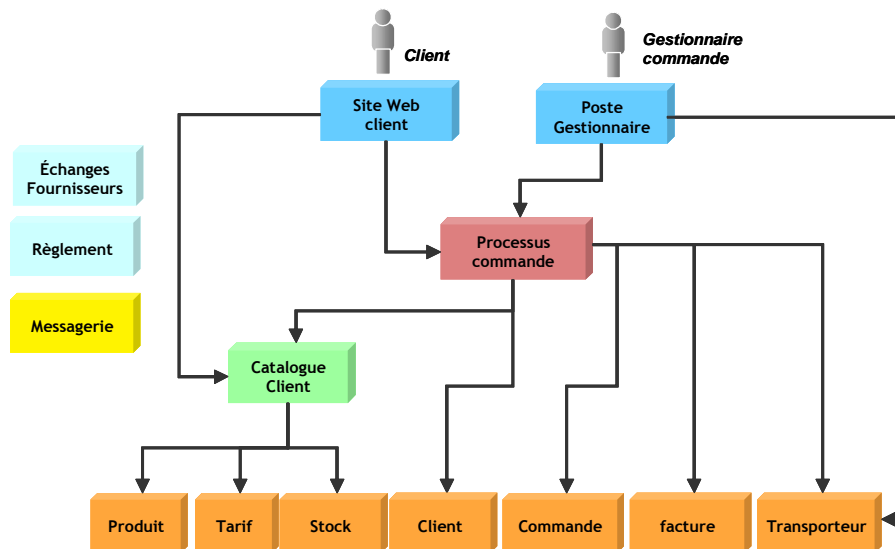


Figure 24 - Représentation des liens de dépendance entre composants



6.3 Dépendance d'interfaces

Chaque composant de service peut potentiellement exposer plusieurs interfaces (qui contiennent chacune des opérations de services). Le lien de dépendance au niveau des interfaces permet une connaissance plus fine de la structure du système et facilite les analyses d'impact.

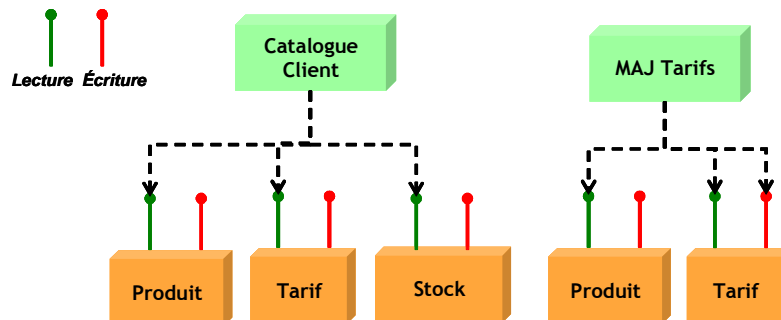


Figure 25 - Utilisation des interfaces entre composants

La notation UML2 offre une représentation directe de composants à l'aide des notions de Port, Part, Composant et Connecteur.

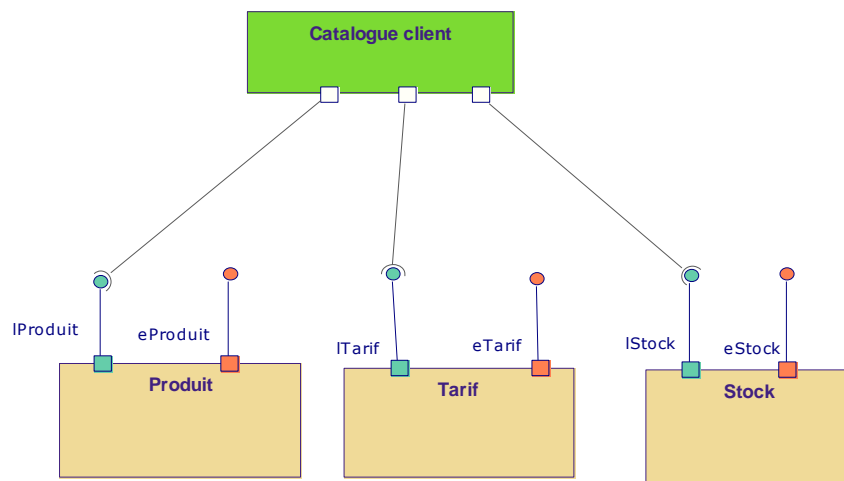


Figure 26 - Composants de service: représentation UML2

Au-delà d'une certaine taille, l'emploi d'un outil de modélisation est un atout indispensable pour la gestion du système. Il facilite l'élaboration en cohérence des différents points de vue, l'audit automatique et l'analyse d'impact, la production d'état (HTML) pour la diffusion et le partage d'information. La mise en œuvre des techniques MDA¹⁹, par la production des constituants logiciels à partir des modèles assure une meilleure maîtrise des changements en réduisant la distance entre les aspects logiques et techniques.

¹⁹ MDA : Model Driven Architecture. www.omg.org/

7 Structuration du SI et application

7.1 Application composite

À la vue de ce type d'architecture, on peut légitimement se demander où sont passées nos applications ? S'agit-il d'une disparition de l'entité logicielle qui structure la majorité des SI d'aujourd'hui ?

Les applications sont les éléments tangibles du point de vue des utilisateurs du système. Dans cette mesure, elles jouent un rôle essentiel : la réponse aux besoins concrets, et la valeur ajoutée du système se concentre à un moment sur le dialogue acteur-système. La mise en œuvre de SOA n'implique pas la suppression des applications : Au contraire, centrée sur les objectifs métiers, SOA tend à améliorer la mise à disposition d'applications toujours plus performantes et adaptées à ses utilisateurs.

Le syndrome architectural. Comme avec d'autres approches, il faut se garder de l'écueil « substituer les moyens aux objectifs ». Les architectures comme les technologies ne constituent pas des buts en soi. Il s'agit d'instruments (outils) mis en œuvre pour répondre à des demandes métier, qui sont mis en œuvre dans le cadre d'applications concrètes.

Une application composite est constituée par un ensemble de composants qui concourent pour répondre aux besoins dédiés à une ligne métier ou une utilisation spécifique du système. Typiquement on va trouver dans une application composite un composant Présentation (IHM, session utilisateur) qui s'appuie sur des composants de services de diverses natures (Processus, Fonction, Entité ...).

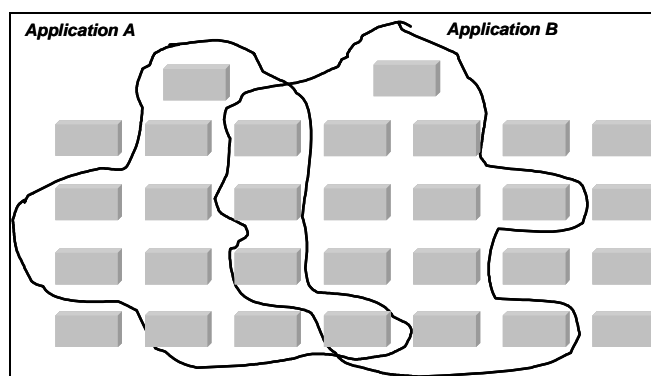


Figure 27 - Applications composites

En terme d'organisation, les projets applicatifs ont en charge la livraison de l'application auprès des clients utilisateurs (internes ou externes) correspondant aux cahiers des charges. Cette responsabilité implique la mise en œuvre de toutes les activités nécessaires : analyse, conception, intégration, validation, construction de l'IHM, développement de composants spécifiques.

Dans une architecture SOA, ces projets applicatifs vont potentiellement réutiliser et partager des composants de service, qui n'appartiennent pas à une application particulière. Le développement, la maintenance et la gestion de ces composants mutualisés sont pris en charge par une organisation transverse qui assure la coordination fonctionnelle et technique.

Conflits d'intérêt. Il existe un risque de conflit d'intérêt entre les responsables des projets applicatifs et les objectifs SOA. Les premiers sont évalués par leurs capacités à fournir dans les délais les éléments correspondants aux cahiers des charges en respectant les coûts fixés. La prise en compte de facteurs transverses peut être perçue comme une contrainte additionnelle. Le rôle de la direction est fondamental pour assurer les coordinations et les actions permettant une collaboration efficace avec les équipes transverses.

Remarque : SOA n'implique pas la refonte totale du SI, et dans la pratique, des applications « anciennes modes » coexistent avec les applications composites. Pragmatiquement certaines vont partiellement utiliser des services tout en conservant leurs structures initiales.

7.2 Visibilité et structuration du SI

La notion de périmètre de visibilité d'un service n'est pas une question annexe et impacte l'organisation aussi bien que l'architecture. Elle précise les éléments qui ont le droit de l'utiliser comme consommateur du service.

En général la structuration des systèmes d'information se retrouve dans l'organisation des responsabilités des équipes (domaine, sous-système etc.). Les composants de service, comme tout élément du système d'information doit être positionné dans cette organisation générale, qui n'est pas modifiée par cette opération.

L'impact du périmètre de visibilité est notable : un service limité à une application et un service publié « en ligne » ne sont pas de même nature. De même, un service disponible sur un large périmètre nécessite des efforts supplémentaires sur les plans contractuels et techniques.

Bonne pratique. Toujours définir le périmètre d'un service publié. La mutualisation de service implique un coût supplémentaire qui dépend en partie du périmètre de visibilité. La disponibilité globale non structurée de tous les services risque de provoquer des dysfonctionnements majeurs sur les plans organisationnel et technique.

Concrètement, le périmètre de visibilité d'un service est défini à partir des unités d'organisation du système (domaine, sous-système etc....) [Figure 28]. Il peut également être défini de manière plus explicite, par énumération des consommateurs autorisés.

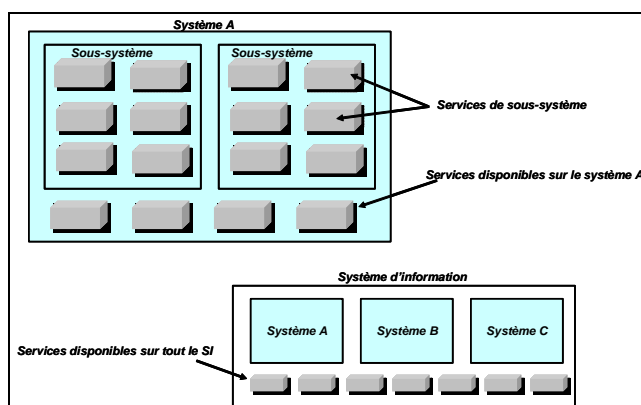


Figure 28 - Périmètre de visibilité

Le périmètre de visibilité d'un composant peut varier au cours de sa vie. Par exemple, un service exposé dans un premier temps sur un sous-système, accroît son périmètre avec le temps pour couvrir tout le SI. Cette modification de visibilité, même à fonctionnalité équivalente implique généralement une reformulation du contrat de service.

8 Mise en œuvre

L'architecture logique, même si elle joue un rôle central, doit être articulée avec les autres axes de construction : Le processus de mise en œuvre, la gouvernance des données, les structures d'organisation, la méthodologie, les infrastructures techniques et les développements. Ces points dépassent le périmètre de ce document, mais leurs maîtrises constituent un facteur clé de la réussite. **Togaf**²⁰ propose un cadre de référence d'évolution des systèmes d'information d'entreprise. La démarche proposée est découpée en un ensemble de phases (Figure 29) et accompagnée de bonnes pratiques, instruments et aides.

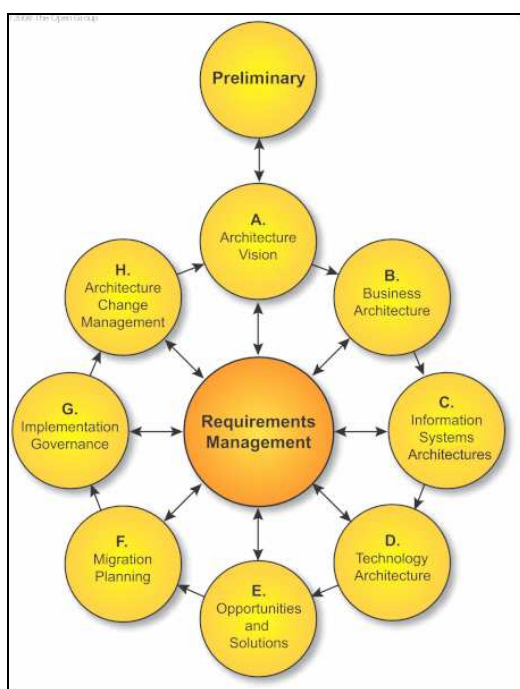


Figure 29 - Phases Togaf

Ce cadre doit le plus souvent être adapté au contexte de l'entreprise, il propose cependant une structure et des pratiques de référence qui s'appliquent quel que soit le style d'architecture (SOA ou pas). La mise en avant des objectifs métiers, le caractère itératif du processus, la participation de toutes les parties prenantes, la qualité de la gouvernance sont parmi les éléments notables de Togaf[®] qui s'impose comme une référence incontournable pour les architectures d'entreprise.

La mise en œuvre de SOA peut apporter une véritable valeur ajoutée. Cependant, comme pour tout changement, le succès passe par une maîtrise des risques et des objectifs. Les expériences passées ont montré que les écueils existent : le manque de résultats clairs dans le déploiement des EAI, certainement due à une orientation trop focalisée sur les solutions techniques et les offres éditeurs.

Une vision clairement orientée métier, du pragmatisme dans les décisions sans oublier des organisations motivées sont autant de facteurs qui faciliteront la réussite de la démarche.

²⁰ www.opengroup.org/



9 Glossaire

Application composite	Application constituée par assemblage d'un ensemble de composants.
BAM	Business Activity Monitoring. Fournit un tableau de bord, des indicateurs de mesure de la performance métier, des outils de monitoring, de reporting et permet le contrôle du rendu fonctionnel des processus métier ; associé de façon classique à un outil de BPM et considéré à juste titre comme une application à part entière.
BPEL	Business Process Execution Language. Conçu par IBM, BEA et Microsoft, c'est la représentation XML d'un processus exécutable, qui peut être déployée sur n'importe quel moteur de processus métier. L'élément premier d'un processus BPEL est une « activité », qui peut être l'envoi d'un message, la réception d'un message, l'appel d'une opération (envoi d'un message, attente d'une réponse), ou une transformation de données. L'activité est définie par la combinaison de Services Web. BPEL utilise WSDL pour décrire les actions d'un processus.
BPM	Business Process Management. Terme générique désignant à la fois la modélisation et la traduction, à l'aide d'un "moteur", des processus modélisés dans la réalité de l'entreprise. A cela s'ajoute, bien évidemment, des outils – indicateurs, tableaux de bord... – de suivi de l'exécution des processus et d'évaluation de leur performance (voir BAM).
BPMN	Acronyme pour Business Process Management Notation (Notation pour la Gestion des Processus Métiers) Utilisée pour la modélisation graphique des processus métier (pas l'exécution). Particulièrement pertinente dans le cadre de la modélisation des processus, de leur analyse et de leur simulation. Cette notation est sous le contrôle de l'OMG (Object Management Group).
Composant de service	Composant logiciel <i>fournisseur</i> de service divisé en vue externe et vue interne. La vue externe est la partie <i>service</i> du composant, qui peut être utilisé par d'autres éléments du système, jouant le rôle de <i>consommateur</i> de service. La vue externe (ou spécification de service) est constituée par un ensemble d'opérations de service regroupées en interface, le contrat et toutes les informations liées à son utilisation (voir QoS et SLA).
Composant Entité	Composant de service qui donne un accès aux informations liées à un objet métier clé. Les opérations sont de type CRUD (Create, Read, Update, Delete).
Composant Fonction	Composant de service qui prend en charge un processus de traitement, ou une adaptation à une vision métier particulière, sous forme de composition de services.
Composant Présentation	Composant de service dédié à l'interaction entre acteurs humains et le système. Il prend en charge les IHM et la gestion de la session utilisateur.



Composant Processus	Composant de service qui automatise un processus métier.
Composant Public	Composant de service qui assure les échanges avec les systèmes externes (B2B, e-business)
Composant utilitaire	Composant de service dédié à des fonctions transverses (annuaire, messagerie, éditique) non spécifiques au métier de l'entreprise.
MDA www.omg.org	Le MDA est une démarche de réalisation de logiciel, proposée et soutenue par l'OMG. L'idée fondamentale est que les fonctionnalités du système à développer sont définies dans un modèle indépendant de la plate-forme (Platform Independent Model, PIM), en utilisant un langage de spécifications approprié, puis traduites dans un ou plusieurs modèles spécifiques à la plate-forme (Platform Specific Model, PSM) pour l'implémentation concrète du système.
OASIS www.oasis-open.org	Organization for the Advancement of Structured Information Standards. Consortium international qui travaille pour la normalisation et la standardisation de formats de fichiers ouverts basés notamment sur XML.
OMG www.omg.org	L'OMG (Object Management Group) est une association américaine à but non-lucratif créée en 1989 dont l'objectif est de standardiser et promouvoir le modèle objet sous toutes ses formes. L'OMG est par exemple à l'initiative des normes UML et CORBA ou encore MDA et BPMN.
Open SOA www.osoa.org	Organisme international qui propose des normes SOA : SCA (Service Component Architecture) et SDO (Service Data Objects). Partenaires : BEA, IBM, IONA, Oracle, SAP, SUN, Siemens, TibCo.
Processus de traitement	Processus qui représente le déroulement d'une activité localisée, de courte durée et non interruptible.
Processus métier	Processus de bout en bout de l'entreprise, qui délivre une valeur ajoutée tangible à l'extérieur par une collaboration de plusieurs unités et acteurs. Un processus métier peut avoir une longue durée de vie et est potentiellement interrompu.
Processus métier transverse	Voir Processus métier
QoS	Quality of service. La qualité de service est une notion née chez les opérateurs de télécommunications vers 1997. On parle de contrat de niveau de service quand une entreprise exige de son opérateur une haute disponibilité de son réseau. La gestion du niveau de service s'est, depuis, étendue au système d'information.
SCA	Service Component Architecture (voir open SOA) Service Component Architecture aims to provide a model for the creation of service components in a wide range of languages and a model for assembling service components into a business solution - activities which are at the heart of building applications using a service-oriented architecture.
Silo	Terme utilisé pour désigner une découpe verticale d'un système, par applications (silos applicatifs). « <i>Application architectures are typically created as independent work efforts that</i>



are usually owned by a specific organizational domain or entity. Applications are usually built to automate business applications within a single organization domain. Those single-domain applications are said to be "siloeed." » Avoiding common pitfalls in SOA adoption, Tilak Mitra, Executive IT Architect, IBM, juin 2006.

- SLA** Service Level Agreement. Contrat définissant les engagements du fournisseur de service quant à la qualité de sa prestation, et les pénalités engagées en cas de manquement. Cette qualité doit être mesurée selon des critères objectifs acceptés par les deux parties. Ex : temps de rétablissement du service en cas d'incident
- TDE** Type de Donnée d'Echange. Structure des informations échangées par les messages ou les flux inter composants. Type des paramètres des opérations de service. Par opposition, les *données persistantes* sont les informations contenues dans les bases de données.
- Togaf** www.opengroup.org **The Open Group Architecture Framework**, également connu sous l'acronyme **Togaf**, est un ensemble de concepts et un standard industriel couvrant le domaine des architectures informatiques d'entreprise, qui peut être utilisé librement et sans coûts par toute entreprise souhaitant développer ou modifier son architecture.
- Togaf a été développé et est continuellement amélioré depuis le milieu des années 1990 par différentes personnes appartenant à un certain nombre de départements informatiques d'importantes sociétés, ainsi que par des fournisseurs de conseils ou de solutions informatiques. Ce travail étant effectué par l'intermédiaire du forum des architectures de l'Open Group.
- W3C** <http://www.w3.org> Le World Wide Web Consortium, abrégé W3C, est un consortium fondé en octobre 1994 pour promouvoir la compatibilité des technologies du World Wide Web telles que HTML, XHTML, XML,, CSS, PNG, et SOAP
- Web service** Il s'agit d'une technologie permettant à des applications de dialoguer à distance via Internet, et ceci indépendamment des plates-formes et des langages sur lesquelles elles reposent. Pour ce faire, les services Web s'appuient sur un ensemble de protocoles Internet très répandus (XML, HTTP), afin de communiquer. Cette communication est basée sur le principe de demandes et réponses, effectuées avec des messages XML.
- Les services web sont décrits par des documents WSDL (Web Service Description Language), qui précisent les opérations pouvant être invoquées, leurs signatures et les points d'accès du service (URL, port .). Les services Web sont accessibles via SOAP, la requête et les réponses sont des messages XML transportés sur HTTP.

