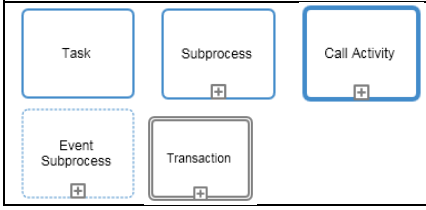
A2.3.1 Généralité

Tableau 8.1 : Groupe d'élément BPMn

Symbole	Elément
	Participant : regroupe par profile
	*Artifacts* : Informations descriptive
	*Gateways* : mode d'unification ou séparation
	Data : représente une source de données
	*Activities* : définit une tache ou ensemble de tâches

A2.3.2 Activités

Ce type d'objet permet d'identifier un processus, un sous-processus, une tâche. Une icône présente sur l'activité permet de comprendre visuellement à quel niveau nous sommes. Des icônes complémentaires permettent de distinguer les différents types de tâches (manuel, automatique, soumis à une règle métier, envoi, réception…)

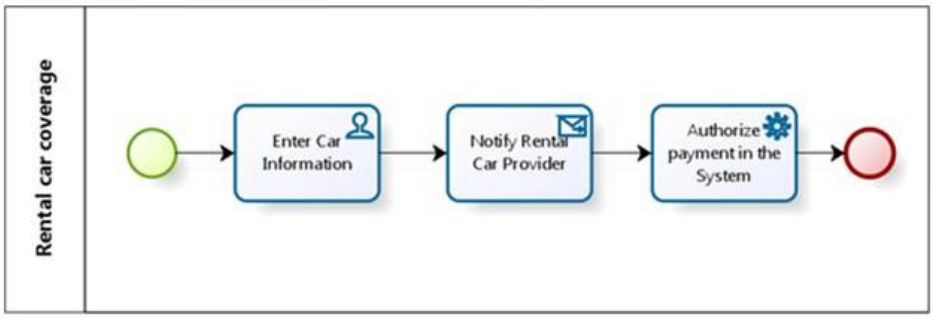


Figure 8.2 : Succession des activités dans un diagramme

A2.3.3 Connecteurs

Les connecteurs permettent d'interpréter les flux : ET, OU exclusif, OU inclusif… L'objet est représenté sous forme de diamant et sa signification est illustrée par l'icône à l'intérieur.

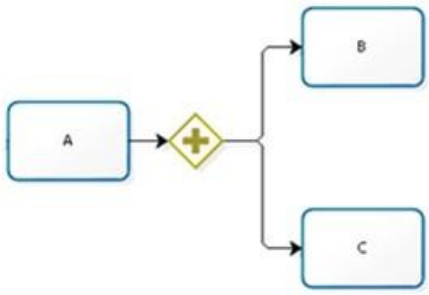


Figure 8.3 : Exemple d'utilisation d'un connecteur

A2.3.4 Evènements

Un événement est quelque chose qui survient. Il déclenche une action ou est le résultat d'une action. BPMN met à disposition 3 catégories d'événements, événement de début, de fin et intermédiaire, qui sont enrichis par une icône permettant d'illustrer le type d'événement (message, temporel, regroupant un sous-ensemble d'événements, …).

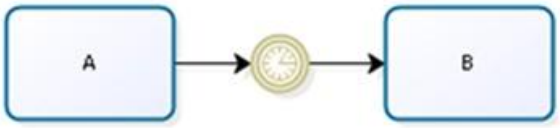


Figure 8.4 : Exemple d'utilisation d'un événement

Tableau 8.2 : Liste des événements BPMn

Elément	**Rôle**	**Start**	**Inter**	**End**
Event	Evénement quelconque.			
Message	Evénement impliquant la réception ou l'envoi d'un message.			
Error	Evénement impliquant le déclenchement d'une erreur.	N/A		
Cancel	Evénement déclenché lors de l'annulation d'une transaction.	N/A		
Compensation	Evénement déclenché lors de l'échec d'une transaction.	N/A		
Timer	Evénement lié à une date spécifique ou à un cycle temporel.			N/A
Rule	Evénement déclenché lorsqu'une règle devient vraie.			N/A
Link	Evénement permettant de connecter la fin d'un processus au début d'un autre.			

Multiple	Evénement résultant de la composition d'autres types d'événements.			
Terminate	Evénement utilisé pour indiquer que toute activité sur le processus doit être immédiatement suspendu.	N/A	N/A	

A2.3.5 Les Artifacts

Ce sont des objets additionnels utilisés pour mieux comprendre le schéma BPMN : comme les activités de même catégorie, les données traitées ou bien des commentaires ou annotations.

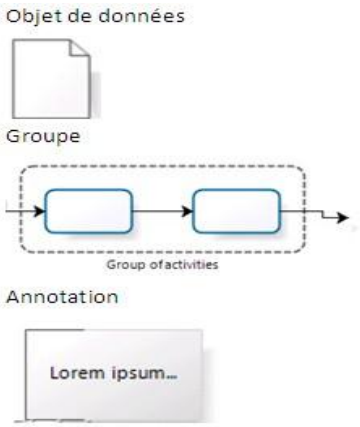


Figure 8.5 : Exemple d'utilisation d'artéfact

Tableau 8.3 : Liste des artéfacts BPMn

Elément	**Rôle**	**Icône de création**
DataObject	Artéfact produit et consommé par les activités du processus.	
Transaction	Une "Transaction" est une région du diagramme utilisée pour représenter l'existence d'une transaction englobant toutes les activités présentes.	
Group	Un "Group" est une région du diagramme utilisée pour rassembler des éléments du processus.	

A2.3.6 Les flux

Les flux permettent de relier et séquencer les éléments du processus entre eux.

Tableau 8.4 : Liste des connecteurs BPMn

Elément	Rôle	Icône de création
Flow	Flux utilisé pour séquencer deux activités du processus.	
ConditionalFlow	Flux conditionnel.	
DefaultFlow	Flux par défaut.	
MessageFlow	Flux utilisé pour faire transiter un message entre deux entités du processus.	

A2.4 Diagramme

BPMn dispose quatre configurations pour présenter un diagramme :

- privé ou public
- collaboration
- chorégraphie
- de conversation

A2.4.1 Diagramme Privé

Un processus privé est un type de modélisation qui permet de représenter un processus spécifique à une organisation en précisant que son contenu reste dans les limites du bassin et ne peut pas le traverser.
Le flux de messages peut traverser les limites du bassin pour montrer les interactions qui existent entre des processus privés qui sont séparés.

A2.4.2 Diagramme public

Le processus public est un processus privé plus des interactions avec à un ou plusieurs Participants en définissant les flux de messages, leur séquence, leur ordre etc.
Seules les activités de communication avec l'autre(s) participant sont présentées dans le processus public. Toutes les autres activités ou tâches internes du processus privé ne sont pas représentées.

A2.4.3 Collaboration

Le diagramme de collaboration permet de modéliser les processus et les interactions entre 2 entités au minimum.

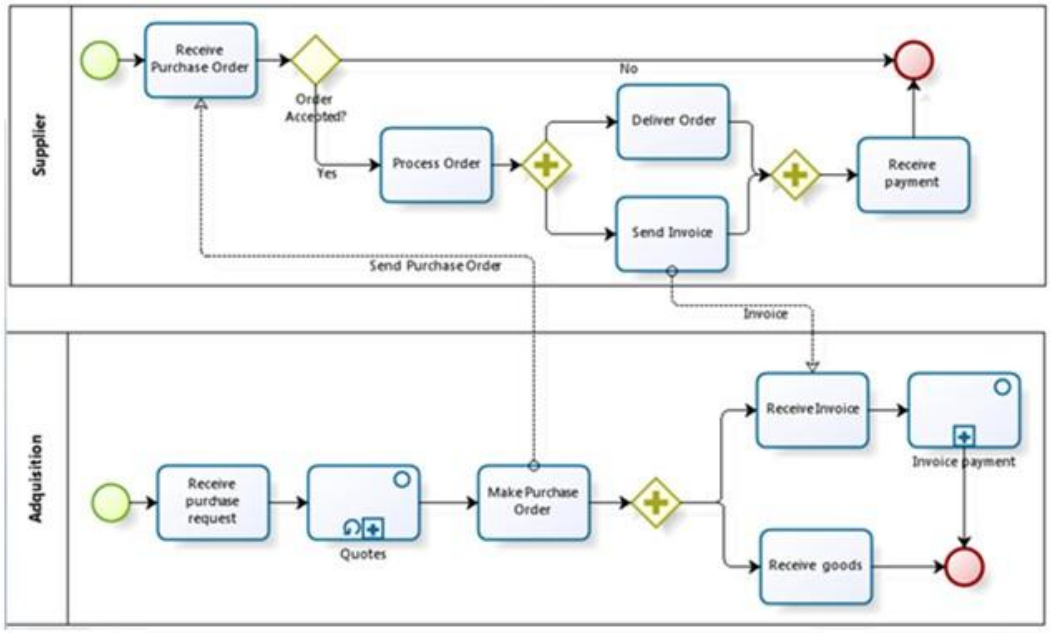


Figure 8.6 : Exemple diagramme de collaboration

Un diagramme de collaboration ne peut contenir qu'un seul bassin (département, service). Ce bassin peut contenir plus d'un couloir (qui correspondent à des rôles : manager, assistante, collaborateur). Pour chaque couloir des activités sont définies.
Pour illustrer les échanges entre 2 couloirs, on utilise des liaisons spécifiques appelées flux de message.

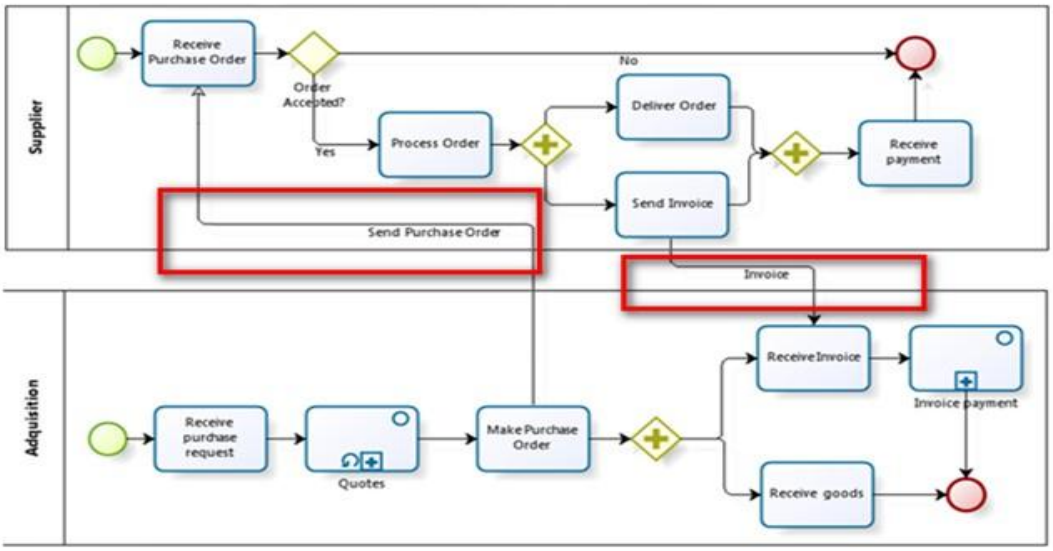


Figure 8.7 : Flux de message dans un diagramme de collaboration

A2.4.4 Diagrammes de chorégraphie

Un diagramme de chorégraphie est utilisé pour analyser la façon dont les participants échangent des informations afin de coordonner leurs interactions. On peut utiliser un diagramme de chorégraphie afin de développer et d'analyser en détails l'échange des messages associés à un nœud de conversation dans un diagramme de conversation.

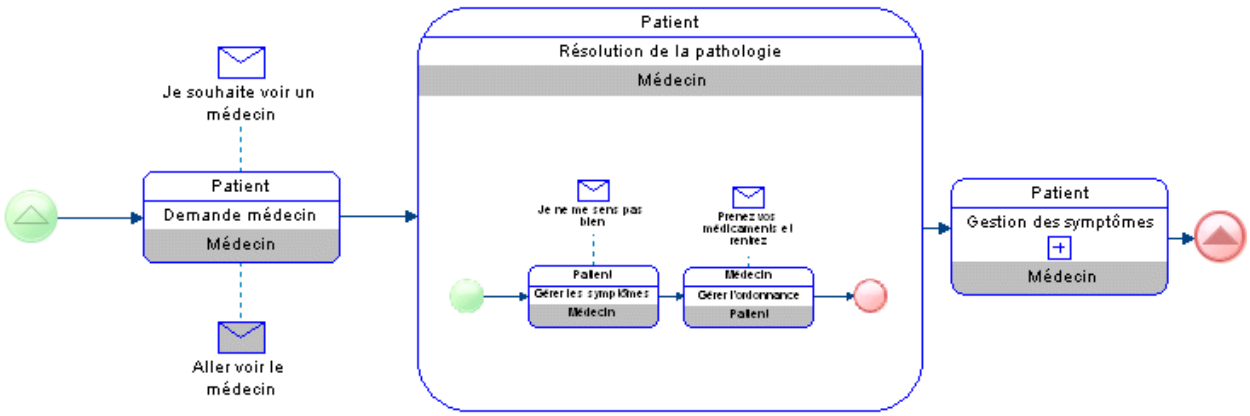


Figure 8.8 : Exemple d'un diagramme de chorégraphie

Cet exemple illustre l'échange de messages entre un patient et son médecin.

Une Chorégraphie est la modélisation d'un comportement attendu entre des Participants qui interagissent les uns avec les autres et qui veulent coordonner leurs activités ou leurs tâches à l'aide de Messages. Dans ce type de modélisation, la focalisation n'est pas sur l'Orchestration (processus public ou privé), c'est-à-dire sur la manière dont est accompli le travail selon le point de vue des participants, mais sur les échanges de Messages entre les Participants.

A2.4.5 Diagrammes de conversation

Le Diagramme de conversation est la description informelle d'un Diagramme de collaboration à haut niveau. Il représente des échanges de Messages (présentés sous la forme d'un regroupement de flux de messages), logiquement reliés entre les Participants et qui concernent un objet d'affaires d'intérêt, par exemple, un ordre, un envoi, une livraison ou une facture. Il représente un ensemble de Flux de Messages qui est regroupé ensemble. Une Conversation peut impliquer deux ou plusieurs participants.
Un diagramme de conversation représente les conversations comme des hexagones, entre les participants (les bassins). Si l'on clique sur l'hexagone, on peut avoir le détail des Flux de Message échangés entre les Participants.
Les bassins qui servent à la représentation d'un Diagramme de conversation, ne contiennent habituellement pas de processus et sont donc présentés vides.
Aucun Diagramme de chorégraphie ne peut être placé entre les bassins du Diagramme de conversation.

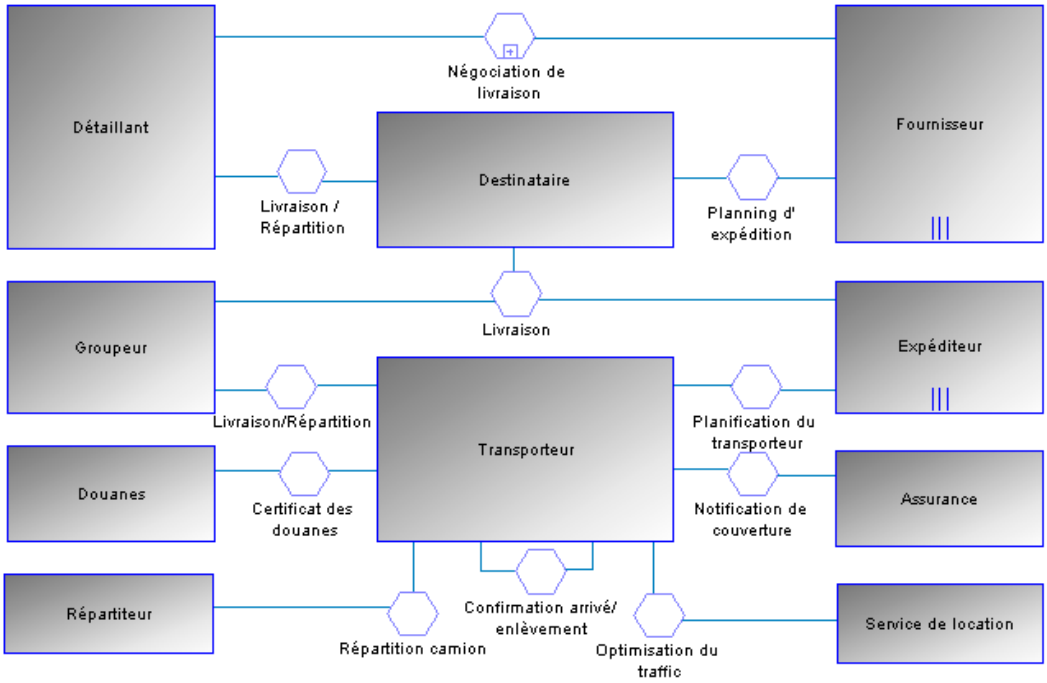


Figure 8.9 : Exemple d'un diagramme de conversation

Dans cet exemple, les diverses conversations associées aux livraisons d'un fournisseur à un détaillant sont analysées.

ANNEXE 3 : Architecture Bus d'événement

L'ESB (Enterprise Service Bus) est le composant central d'une architecture orientée services (SOA). Il autorise une approche de l'intégration hautement distribuée et indépendante de la plate-forme. Il s'appuie sur les standards pour soutenir la messagerie, les Services Web, le routage et la transformation de données.

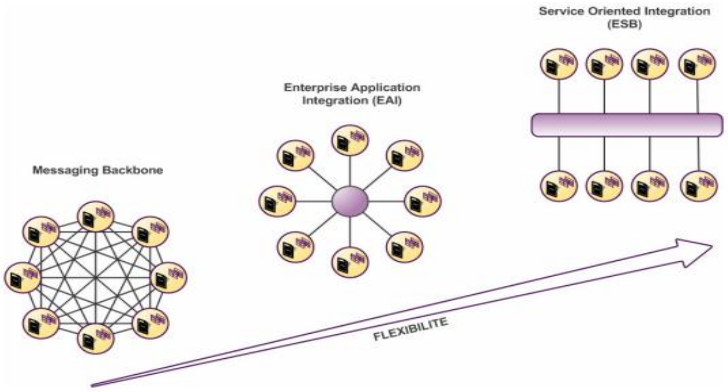


Figure 9.1 : Evolution de l'architecture de partage de message

A3.1 Architecture

Le bus de services se base sur des principes suivants :

- la découverte dynamique
- l'exposition et l'orchestration des services
- outil graphique permettant d'implémenter les processus et de les orchestrer
- la distribution forte
- la communication par message
- la communication entre services se fait par message représenté par un document auquel le service consommateur s'abonne.

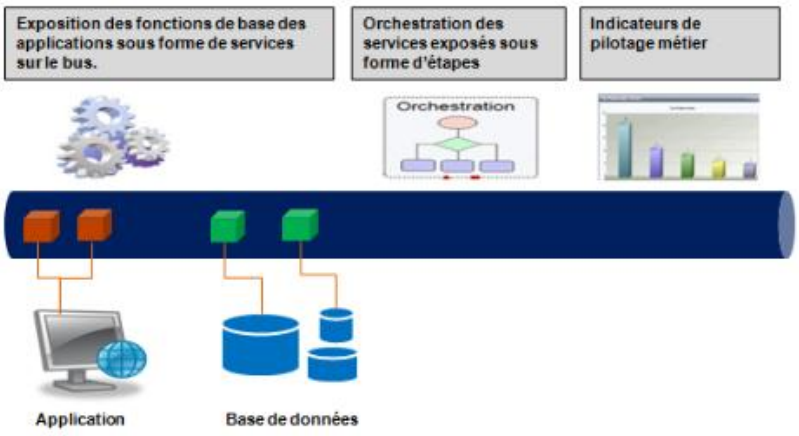


Figure 9.2 : Les éléments liés aux Bus

Son architecture est basée sur :

- échange asynchrone entre applications
- interopérabilité entre le bus et les applications grâce aux services
- transformation et enrichissement des données circulant sur le bus

ESB se concentrent sur les fonctions d'interconnexion et de médiation, et s'appuient pour cela sur un ensemble de standards parmi lesquels :

- Web Services pour gérer les communications synchrones.
- XML pour définir les formats des messages
- JMS2 pour adresser la communication asynchrone avec les MOM.
- JCA pour la connexion aux progiciels et systèmes exotiques (ERP, CRM, Mainframes, etc.)

A3.2 Fonctionnement

Le bus de service d'entreprise achemine les messages entre les applications, qui sont demandeurs ou fournisseurs de services. Le bus convertit les protocoles de transport ainsi que les formats des messages entre les demandeurs et les fournisseurs. Dans ce schéma, chaque application utilise un protocole différent (représenté par les différentes formes géométriques de leurs connecteurs) et utilise différents formats de message.

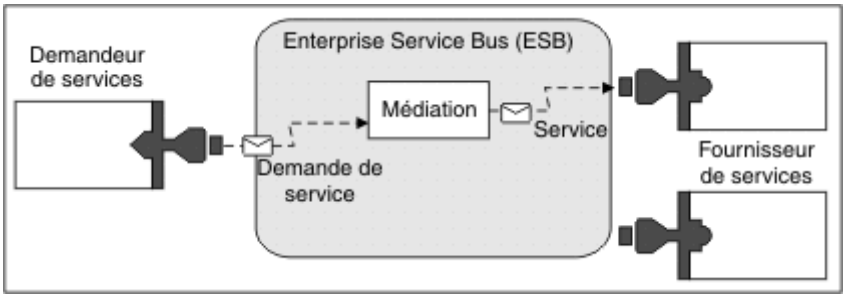


Figure 9.3 : Echange des informations au niveau du Bus

A l'aide des médiations, un bus de service d'entreprise exécute les actions suivantes entre le demandeur et le service :

- acheminement des messages entre les services : le bus de service d'entreprise offre une infrastructure de communication commune permettant de se connecter aux services et donc aux fonctions métier qu'ils représentent, sans que les programmeurs aient à écrire et gérer une logique de connectivité complexe.
- conversion des protocoles de transport entre le demandeur et le service : le bus de service d'entreprise est un moyen cohérent normalisé d'intégrer des fonctions métier qui utilisent des normes informatiques différentes. Cela permet de disposer de fonctions métier qui ne pourraient normalement pas communiquer, telles que la connexion d'applications dans

des silos départementaux ou la participation des applications de différentes sociétés aux interactions de service.
- conversion des formats de message entre le demandeur et le service : le bus de service d'entreprise permet aux fonctions métier de d'échanger des informations dans des formats différents, le bus garantissant que l'information distribuée à une fonction métier a le format correspondant à l'application.
- traitement des événements métier provenant de sources différentes : le bus de service d'entreprise prend en charge les interactions basées sur l'événement en plus des échanges de message pour traiter les demandes de service.

A3.3 Médiation et routage

La médiation est la principale valeur ajoutée d'un ESB. Dans ce domaine, il doit proposer une sémantique riche, susceptible de fiabiliser et de simplifier les échanges, tout en offrant une grande souplesse de mise en œuvre.
Les principales fonctionnalités que l'on peut attendre d'un ESB en matière de médiation et de routage :

- support de différentes sémantiques d'échange de messages (synchrone « requête-réponse », asynchrone point-à-point, asynchrone selon le modèle « publication-souscription »)
- gestion de règles de routage sur les messages.
- mécanismes de gestion de la priorité des messages.
- services de transformation et de conversion de messages (ex : XML <-> EDI, etc.).
- manipulation des messages (enrichissement, transformation, combinaison, découpage).
- validation des données entrantes ou sortantes.
- gestion de versions de services de façon transparente (systèmes évoluant à des rythmes différents).

A3.4 Apports du bus de services

Le concept service bus permet de mettre en place un système capable de :

- intégrer de concepts standards
- router de données

L'utilisation des documents XML et son mécanisme par le service bus permet de :

- d'avoir une architecture distribuée : Les fonctions du SI sont vues comme des briques réutilisables et faiblement couplées, donc pouvant être déployées indépendamment les uns des autres. Ces briques peuvent être orchestrées dans le cadre d'un processus technique ou métier.
- souplesse : Interface centralisée de gestion de configuration et de déploiement.

A3.5 JBI

JBI est une solution java pour implémenter une plateforme de bus d'événement. Il définit deux types de composants :

- les Service Engine (SE) qui sont des composants qui fournissent le logique métier et des services de transformation aux autres composants. Ils peuvent également être consommateur d'autres Service Engine.
- les Binding Component (BC) qui sont des composants qui assurent la connectivité pour les services extérieurs à l'environnement JBI. Cela peut impliquer les protocoles de communication ou des services fournis par le SI de l'entreprise.

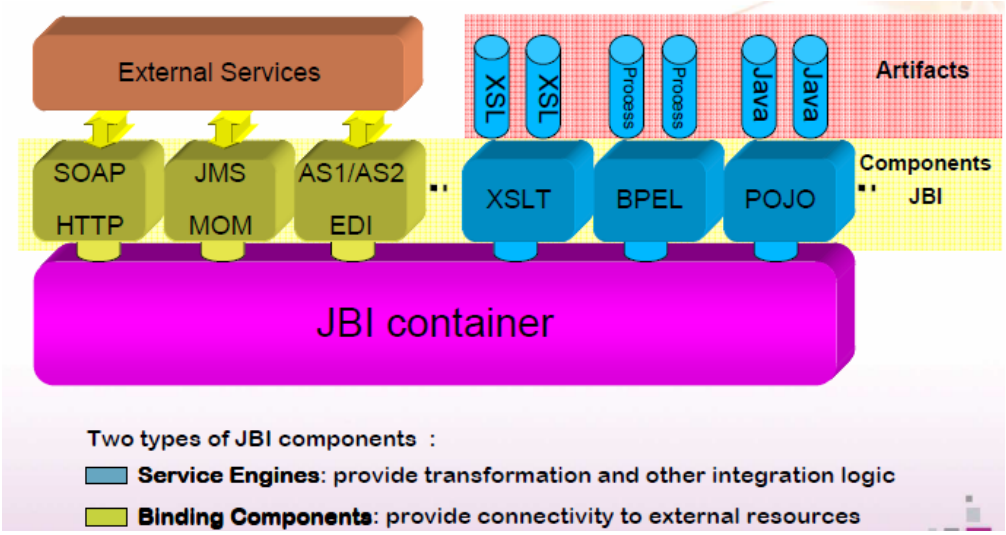


Figure 9.4 : Composants JBI

Exemple de SE :

- SE XSLT (eXtensible Stylesheet Language Transformations) qui permet d'appliquer une transformation d'un fichier/message XML à partir d'une feuille de style XSL,
- SE Validator qui permet de valider un message grâce à son XSD,
- SE CSV (Comma-Separated Value) qui permet de transformer un document au format CSV en document au format XML,
- SE EIP (Enterprise Integration Pattern) qui permet d'implémenter certains patrons d'intégration d'entreprise,
- SE JSR181 (Web Services Metadata for the java Platform) qui permet d'exposer un POJO (Plain Old Java Object) annoté comme un service JBI,
- SE POJO (Plain Old Java Object) qui permet de déployer des classes Java comme des services,
- SE BPEL (Business Process Execution Language) qui permet d'offrir un moteur d'exécution BPEL,
- SE SCA (Service Component Architecture) qui permet d'offrir une implémentation de SCA,

Exemple de BC :

- BC FileTransfert qui permet d'envoyer et de recevoir des fichiers,
- BC FTP (File Transfert Protocol) qui permet le transfert et le listage de fichiers sur un serveur ftp,
- BC JMS (Java Message Service) qui permet de s'interconnecter avec des Destinations (Queue/Topic) extérieures,
- BC SMTP qui permet d'émettre et de recevoir des fichiers d'un service de mail extérieur,
- BC SOAP (Simple Object Access Protocol) qui permet de s'interconnecter avec un service Web extérieur afin d'exposer des services JBI comme des services Web,
- BC XMPP (eXtensible Messaging and Presence Protocol) qui permet de s'interconnecter avec le protocole XMPP qui est un protocole standard de messagerie instantané (Google Talk, Jabber, …),

A3.5.1 Message

Un JBI Provider permet d'émettre et de recevoir des messages (ME) entre les divers services qu'il héberge au travers de composants. En fait, ces transmissions de messages suivent certains patterns qui sont définis au nombre de 4 (ce nombre n'est pas imposé et chaque JBI Provider est libre d'offrir plus) :

- In-Only

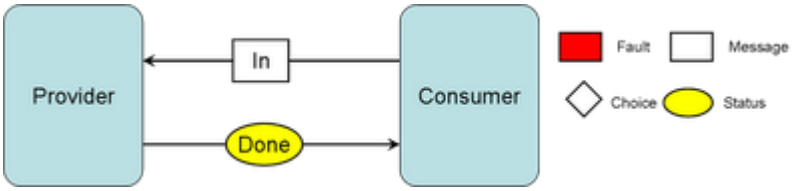


Figure 9.5 : Message « In only »

- In-Out

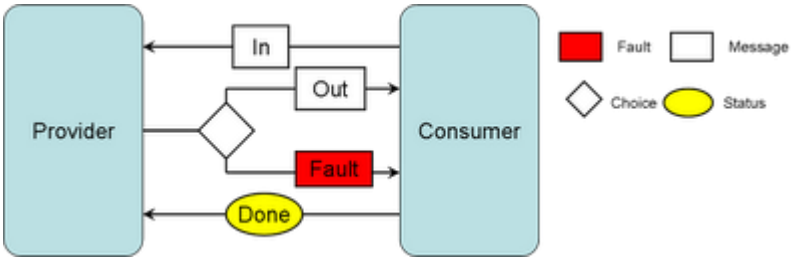


Figure 9.6 : Message « In/Out »

- In-Optional Out

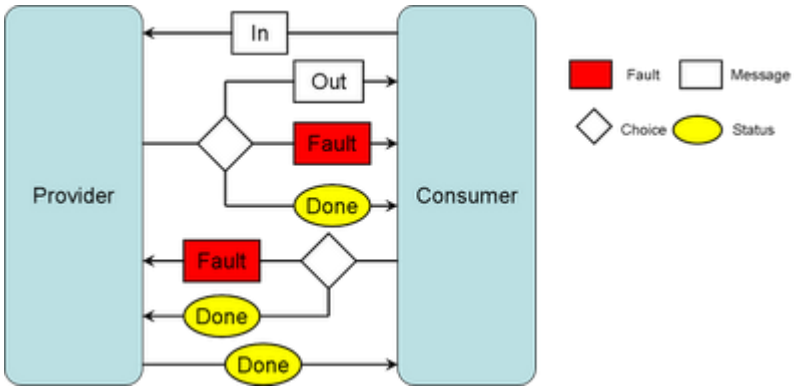


Figure 9.7 : Message « In optional Out »

- Robust In-Only

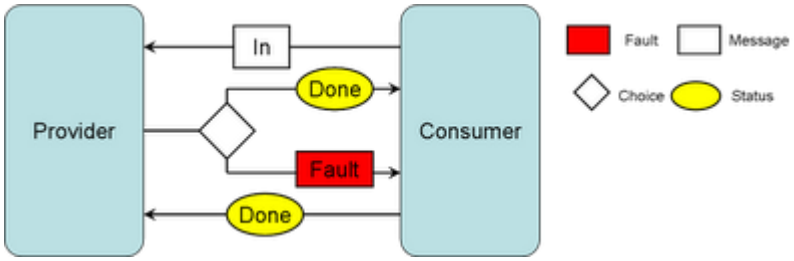


Figure 9.8: Message « robust In only »

Ce pattern d'échange (Message Exchange Pattern – MEP) est positionné par le composant émetteur (SU), et c'est au composant recevant le message de préciser les patterns qu'il supporte et comment il les supporte (par exemple, un BC JMS en mode fournisseur ne fournit pas de réponse s'il ne fait qu'émettre. Aussi, les MEPs qu'il supporte ne peuvent être que In-Only ou Robust In-Only).

A3.5.2 Cycle de vie du composant JBI

Un composant après son déploiement sur le système peut passer de trois états :

- démarré
- arrêté
- fermé

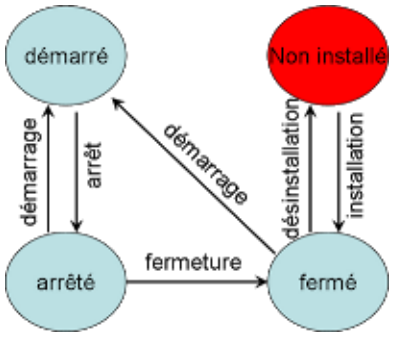


Figure 9.9 : cycle de vie d'un composant JBI

A3.5.3 Les composants : conteneurs de services

Pour qu'un service puisse s'exécuter dans un conteneur JBI il doit être en quelque sorte configurés et instanciés pour s'exécuter. Les composants agissent ainsi comme des conteneurs pour des instances de service :

- les instances des services sont packagées sous la forme de Service Unit (SU). Ces SU sont ensuite déployés dans l'environnement d'exécution JBI qui va installer chaque instance dans le composant (BC ou SE) fournissant le service.
- un SU d'un service exposé contient les parties suivantes :
    - Jbi.xml *deployment descriptor* : ce descripteur de déploiement est consommé par le moteur CXF qui est responsable de l'instanciation et l'activation du service.
    - service *implementation* : les classes d'implémentation du service incluant le code stub de WSDL.
    - WSDL *contract* : la copie serveur du contrat qui utilise le format *xformat binding* et je transport JBI destiné à communiquer avec le NMR.

Les SU doivent être packagés sous la forme de Service *Assembly* (SA) pour être « déployable ».

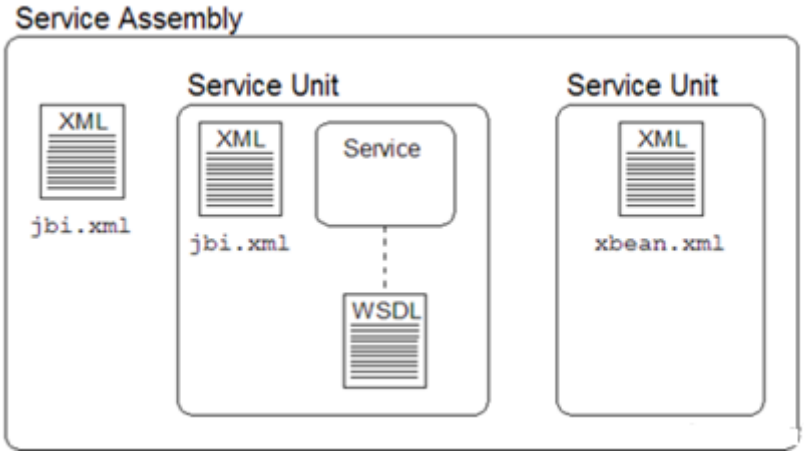


Figure 9.10 : Empaquetage de composant