Support de cours

Ingénierie des Processus

2ème année

Filière Génie Informatique et Digitalisation – EMI

Préparé par : Pr. Fatima-Zahra BELOUADHA

Partie 1 : Processus et BPM

Partie 1 : Objectifs

- Comprendre le concept de processus, les concepts BPM et s'initier au standard BPMN de modélisation des processus

Partie 1 : Plan

- Processus et BPM
- Standard BPMN

CH1. Processus et BPM

- Motivations de l'approche processus
- Processus : concept et types
- Processus métier/Processus collaboratif
- L'approche processus dans la pratique
- BPM
- Workflow Vs BPM
- BPMN pour la modélisation des processus
- BPEL pour l'exécution des processus
- Urbanisation, BPM et SOA

Motivations de l'approche Processus

Défis

Productivité

Production/inv estissement

Compétitivité

Part marché

Flexibilité

Partenariats

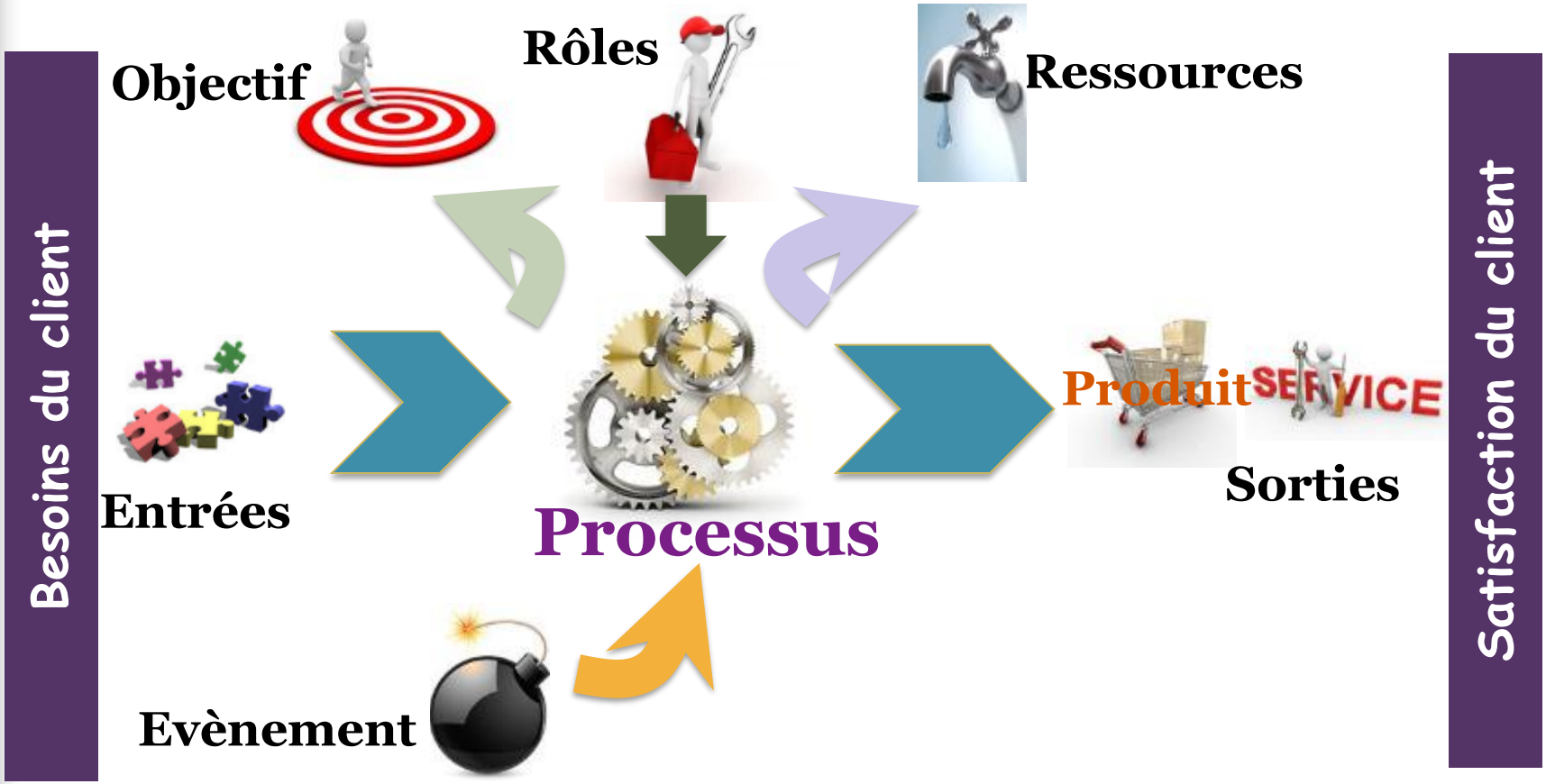
Adaptabilité

Changements sans perte d'efficacité

Solution

L'approche processus (ISO 9000, V2000)

Définition synthétique d'un processus

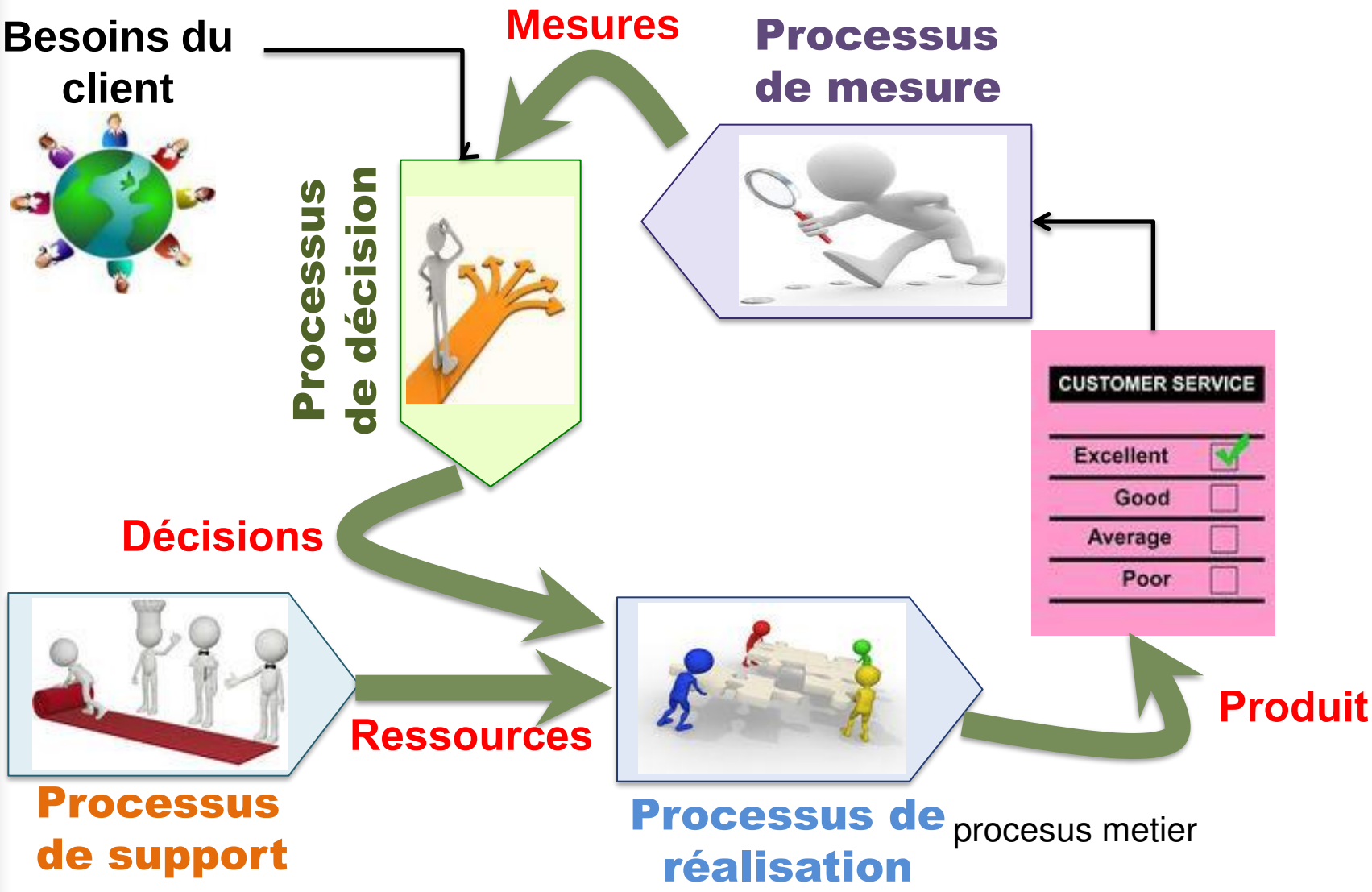


Ensemble d'**activités qui interagissent** et qui à partir d'un ensemble d'**entrées** produisent un ensemble de **sorties** pour atteindre un **objectif** donné. Ses activités sont exécutées par des machines ou des acteurs humains auxquels sont assignés des **rôles**, et peuvent susciter des **ressources** et sont parfois conditionnées par des **événements**. Le processus part des **besoins du client** avec comme but ultime la **satisfaction de celui-ci**

Terminologie*

- Activité : Ensemble de tâches élémentaires
- Entrée : Objet sur lequel opère une activité
- Sortie : Résultat produit par une activité
- Objectif : ce que le processus vise à atteindre pour répondre aux orientations de l'entreprise
- Rôle : Responsabilité
- Ressource : Moyen requis pour effectuer une activité (ex : BD ou outil logiciel).
- Evènement : Fait qui se produit et qui peut déclencher une activité (ex : échéance temporelle ou résultat qui se produit)

Types de processus



Objectifs des processus

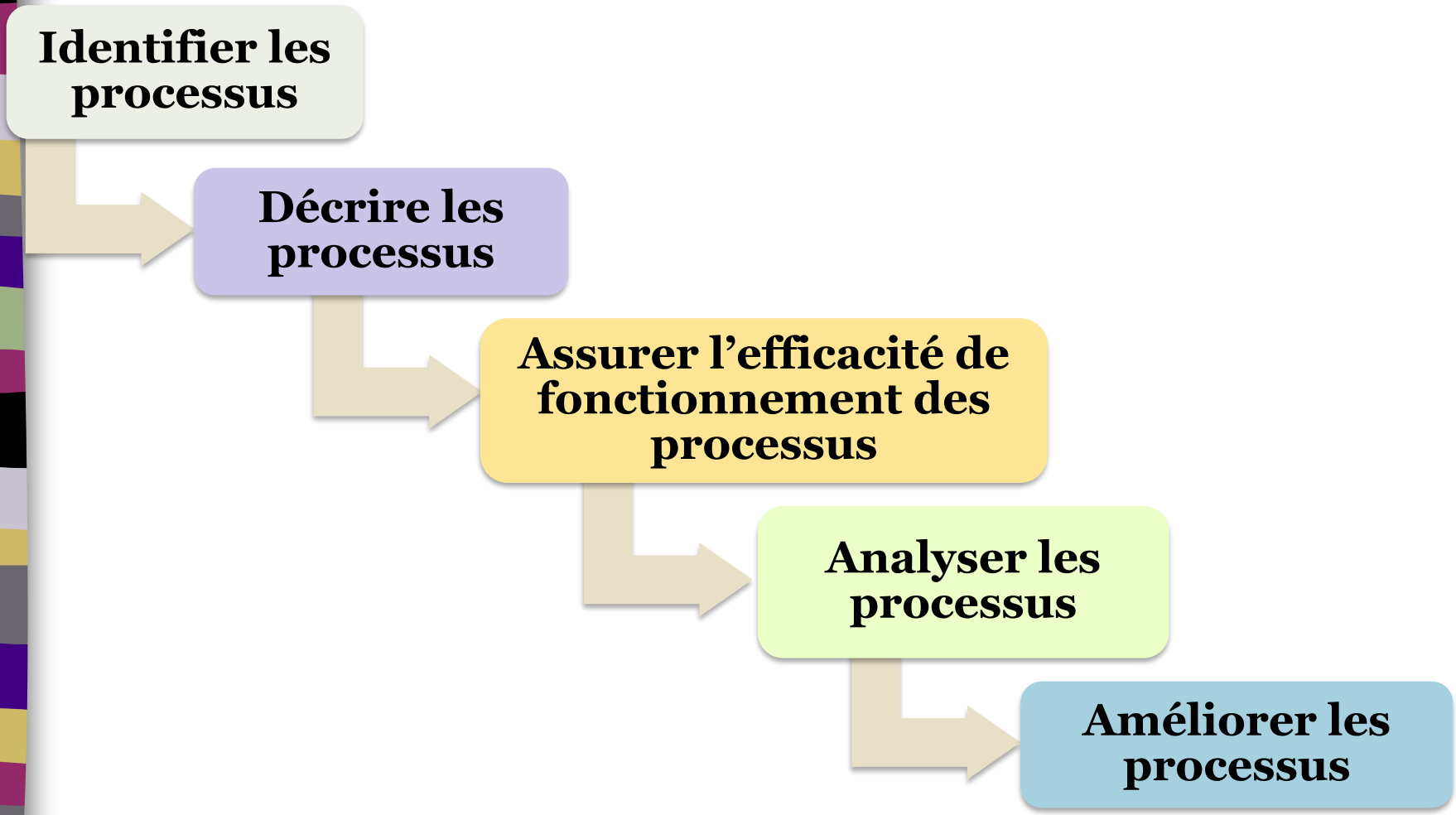
- Processus de support :
  - Mettre à la disposition des processus métier les ressources utiles pour assurer leur bon déroulement
  - Ex: gestion RH ou financière, admin. syst, formation…
- Processus de réalisation (opérationnels / métiers) :
  - produire un produit ou un service ou encore contribuer à sa réalisation
- Processus de mesure :
  - contrôler et mesurer la performance des activités menées pour l'amélioration continue des processus
- Processus de décision (de management) :
  - Diriger, piloter, prendre et valider des décisions qui peuvent affecter les processus métier

Processus métier

- Agencement d'activités confiées à différents participants, qui met en relief les interactions entre ces derniers sous forme d'échange d'informations et qui apporte une valeur ajoutée pour le client et/ou l'entreprise.

- Les participants peuvent être :
  - Des applications / services du SI
  - Des acteurs humains
  - D'autres processus métiers

L'approche processus selon ISO 9001



L'approche processus dans la pratique

Cartographier les processus

- **Mettre en relief les éléments, séquence et interaction**

Piloter les processus

- **Définir les méthodes adéquates, mettre à disposition les ressources et assurer les flux pour garantir la cohérence des processus**

Améliorer les processus

- **Surveiller, mesurer, corriger pour avoir des processus opérationnels, efficaces et efficients**

Problématique de la gestion des processus dans la pratique

- Usage d'outils différents par les équipes méthode, métier et techniques
  - Modélisation technique et implémentation (à la main) sans pouvoir capitaliser sur la modélisation métier
- Conséquences
  - Collaboration difficile
  - Time to Market élevé: de la modélisation à l'exécution des processus & de la modification des processus à son implémentation
  - Incohérence entre les modélisations fonctionnelle et technique
  - Décalages entre les besoins exprimés au départ et les applications réalisées

BPM : Business Process Management

- Enjeu
  - Unifier sous un seul outil toutes les visions des processus : métier, fonctionnelle et technique
- Objectif
  - Permettre aux décideurs, analystes métiers, équipes fonctionnelles et équipes techniques de collaborer pour la définition et l'évolutivité des processus métiers via un seul outil agrégeant les différentes visions

Avantage du BPM

- Permet de gérer le cycle de vie des processus métier
  - **Modélisation**
    - Modèles des processus et règles de gestion
  - **Exécution**
    - Exécution des processus modélisés en orchestrant l'intervention d'acteurs humains et d'applications
  - **Pilotage (Monitoring) :** BAM
    - Activité pilotée via des indicateurs et points de contrôle associés aux processus (performance)

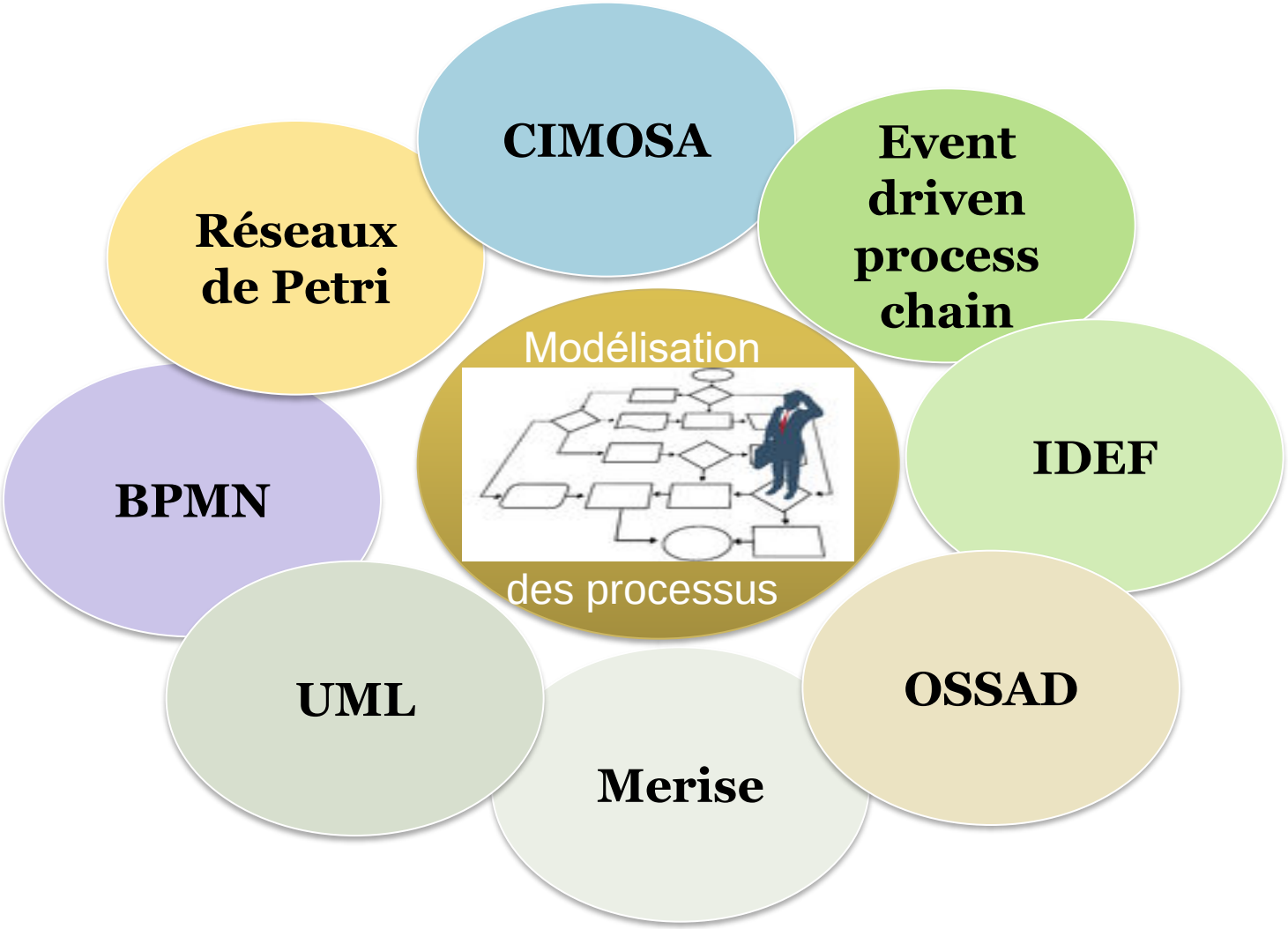
Qui est derrière BPM ?

- BPMI.org : BPM Initiative
  - Modélisation de processus : Aris, MEGA, Rational, Popkin
  - EAI : WebMethods, SeeBeyond, Vitria
  - Applicatif : IBM, BEA
  - Workflow : Fujitsu, FileNet, Staffware
  - ERP : SAP, PeopleSoft, Siebel

Standards BPM

- Modélisation, exécution et interrogation
  - BPMN : Business Process Modeling Notation (OMG)
  - BPML : Business Process Modeling Language/BPEL : Business Process Execution Language (standard OASIS)
  - BPQL : Business Process Query Language

Modélisation des processus



CH2. Standard BPMN

- Evènements
- Activités
- Pools/Lines
- Connexions
- Passerelles
- Artifacts
- Exemple de Diagramme BPMN

Standard BPMN

- Notation graphique pour la modélisation des processus (V1.0 en 2004, V2.0 Jan 2011)
- Définit un BPD où le processus est un réseau d'objets graphiques représentant des activités et le contrôle sur leur enchaînement
- Avantages :
  - Notation compréhensible par des analystes, des techniciens et des organisateurs
  - Mapping complet vers les langages d'exécutions : génération automatique du processus BPEL à exécuter par le moteur de processus.

BPMN : Evènements (1/3)

- Evènement
  - Fait (qqe chose) qui se produit au cours d'un processus métier et affecte son flux
  - A généralement une cause et un effet
- Types d'évènements

  - **None** : sans type particulier
  - **Message** : réception ou envoi d'un message
  - **Temporisateur** : délai écoulé, date, instant ou cycle ayant lieu
  - **Conditionnel** : règle validée
  - **Signal** : réception ou envoi d'un signal (différent de message car il n'a pas de processus destinataire spécifique)

BPMN : Evènements (2/3)

- Types d'évènements (suite)

  - Multiple : **seule une** cause ou seul un résultat **parmi plusieurs** est suffisant pour démarrer ou terminer un processus

  - Multiple parallèle : **plusieurs** causes ou résultats sont nécessaires pour démarrer ou terminer un processus

  - Escalation : **mécanismes** pour **accélérer l'achèvement** de l'activité doivent être exécutés (suite à une contrainte sur son exécution (**ex: deadline**))

  - **Error** : une **exception** a **terminé** ou devrait être générée pour terminer le processus

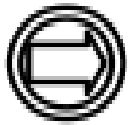
BPMN : Evènements (3/3)

- Types d'évènements (suite)

– Compensation : Mécanismes de compensation doivent être exécutés

– **Annulation** : une transaction doit être annulée

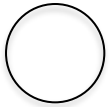
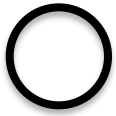
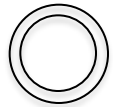
– Terminaison : Termine le processus (arrêt de toutes les activités sans compensation ni trait. d'exceptions)



– Lien : mécanisme pour **connecter 2 sections** d'1 Processus. Utilisé pour créer des boucles ou éviter de long lines d'un flux de séquence

BPMN: contextes des évènements (1/2)

- Représentation des évènements selon différents contextes de création

  - **Début**/Start : marquer le début d'un processus 
  - **Fin**/Terminate : marquer la fin du processus 
  - **Intermédiaire**/Intermediate : affecter le déroulement du processus sans être de début ou de fin 

BPMN: contextes des évènements (2/2)

- Représentation des évènements selon différents contextes de création
  - Sans interruption : indiquer que **l'activité en cours** ne sera pas interrompue. L'événement concerné est un événement intermédiaire **de frontière** ou un **sous-processus événement**
  - Capture/Catching : marquer qu'un processus doit **attendre jusqu'à la capture** de l'événement (**intermédiaire**)
  - Déclenchement/Throwing : marquer que l'événement (intermédiaire) créé est **déclenché** par un processus qui continue à le déclencher **jusqu'à sa capture**




BPMN : Liste synthétique des évènements

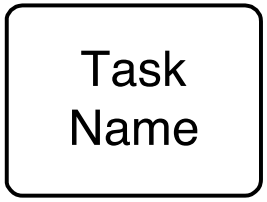
Types	Start			Intermediate				End
	Top-Level	Event Sub-Process *Interrupting*	Event Sub-Process *Non-Interrupting*	Catching	Boundary *Interrupting*	Boundary *Non-Interrupting*	Throwing	
None								
Message								
Timer								
Error								
Escalation								
Cancel								
Compensation								
Conditional								
Link								
Signal								
Terminate								
Multiple								
Parallel Multiple								

BPMN : Activités

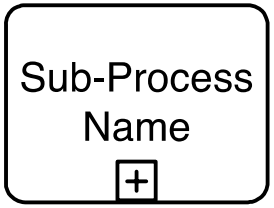
- Activité
  - Travail qu'une entreprise effectue au cours d'un processus
  - Peut être atomique (**tâche**) ou composite (**sous-processus**)
- Sous-processus
  - Composé de tâches et/ou d'autres sous-processus

BPMN : représentations des activités

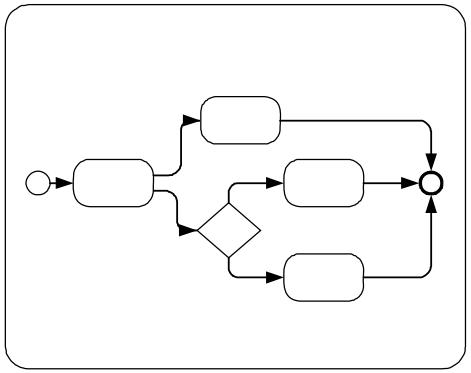
- **Tâche** :



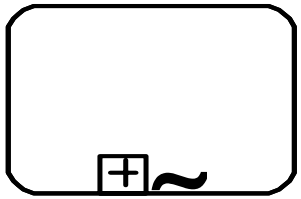
- **Sous-processus** (collapsed) :
  - Sans détail visible



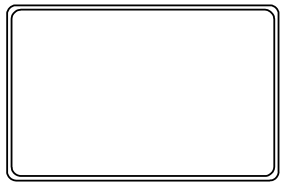
- **Sous-processus** (expanded) :
  - Avec détail
  - Flux de séquences ne traversent pas ses limites



- Sous-processus ad-hoc :
  - Groupe d'activités sans séquencement (à définir par ceux devant l'effectuer)

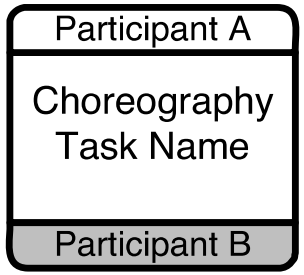


- Transaction :

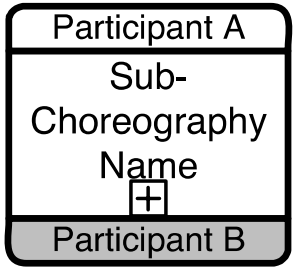


BPMN : Activités de chorégraphie*

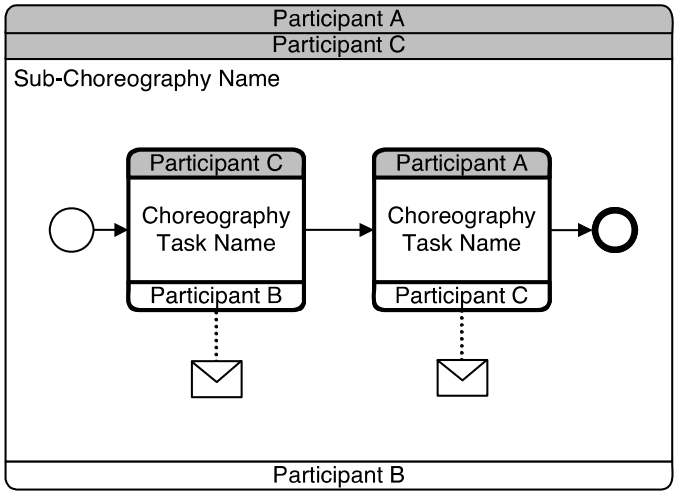
- Tâche de chorégraphie :
  - Représente 1 ou +eurs échanges de messages entre 2 participants



- Sous-chorégraphie : collapsed
  - Sans détail visible

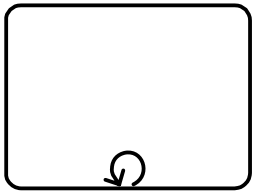


- Sous-chorégraphie : expanded
  - Avec détail visible
  - Flux de séquences ne traversent pas ses limites

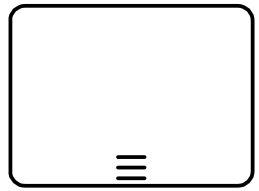


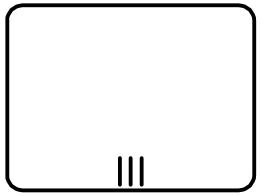
BPMN : Activités et boucles *

- Tâche de type **boucle** :
  - Répétition déterminée par les attributs de la tâche

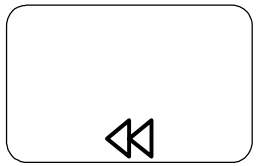


- Tâche multi-instances :
  - Se produit +eurs fois
  - Ses instances se produisent de façon séquentielle ou parallèle

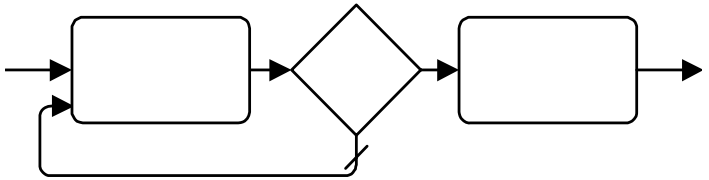




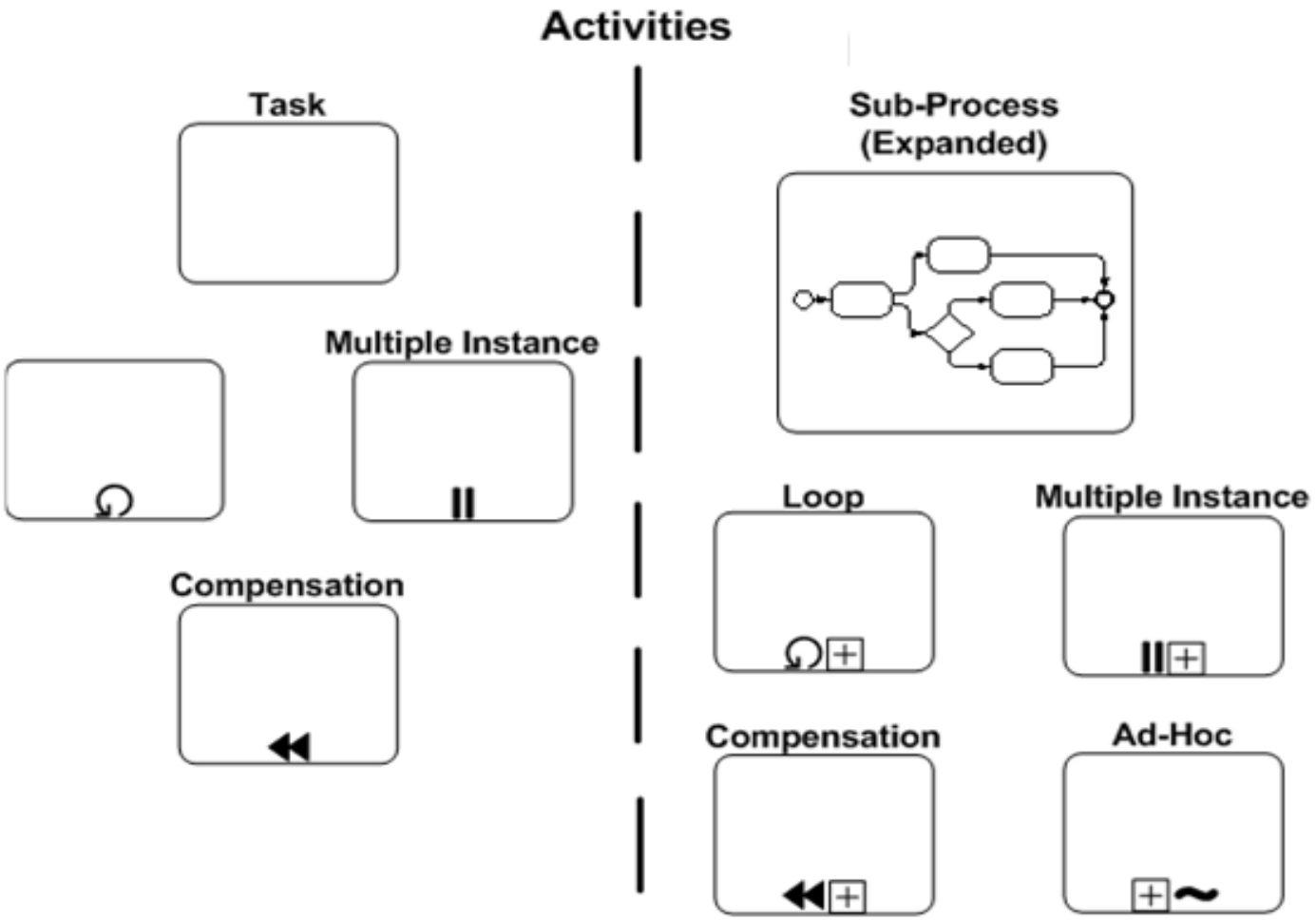
- Tâche de compensation de type boucle :
  - Se produit +eurs fois
  - Implique une sorte de compensation



- **Flux de séquence** de type boucle



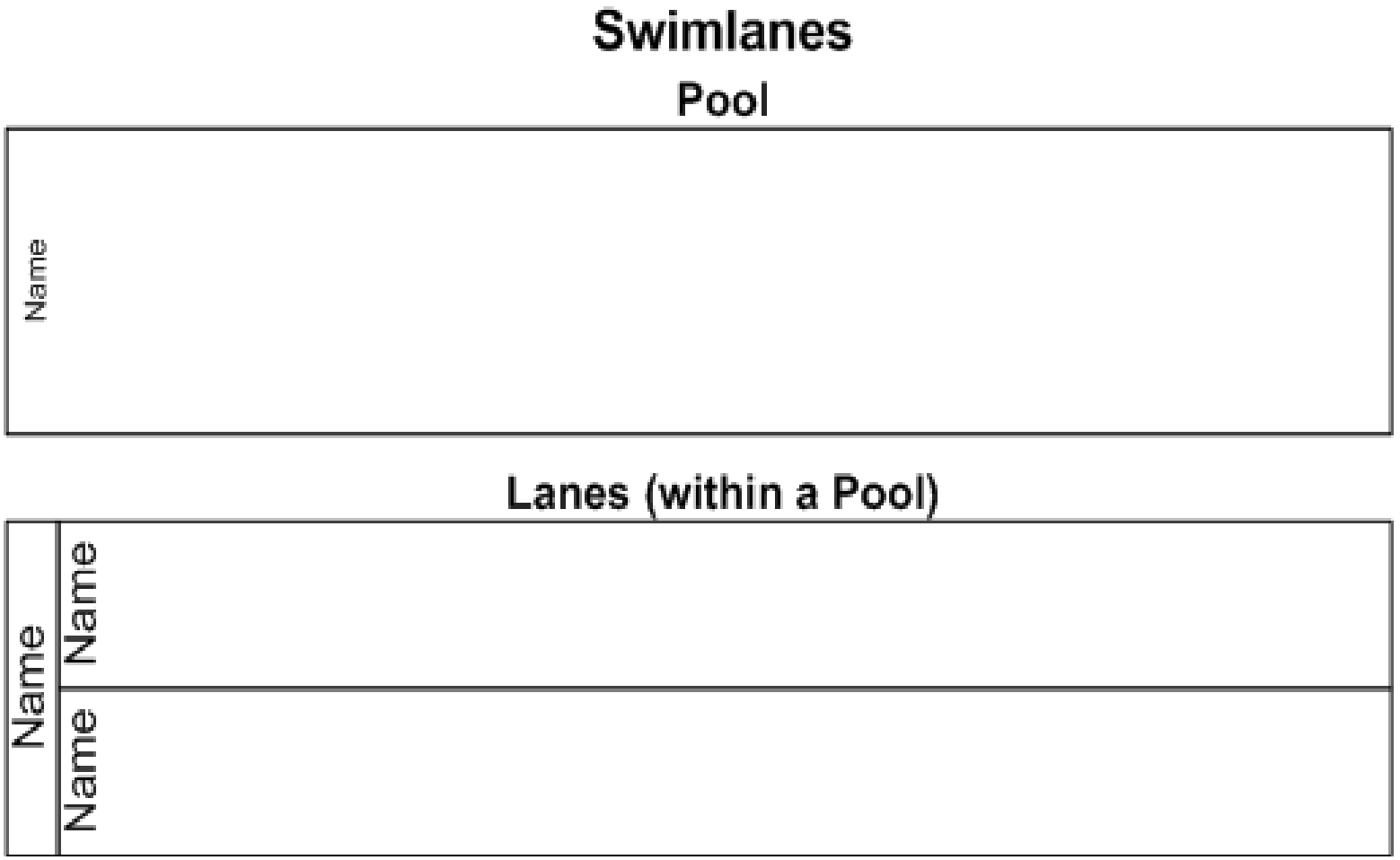
BPMN : Liste synthétique d'activités



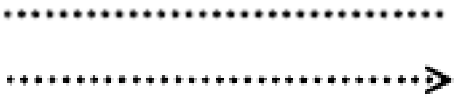
BPMN : pools/lanes (Couloirs)

- Pool
  - Comprend les **activités d'un participant** (acteur physique ou entité) dans un processus collaboratif
  - Conteneur graphique (swimlane) pour la **répartition des activités** dans un contexte B2B
- Lane
  - Sous-partition **verticale ou horizontale** d'un processus utilisée pour organiser et catégoriser les activités
  - Peut être utilisée au sein d'un pool pour représenter un **participant de l'entité** désignée par le pool (intervient dans certaines de ses activités)

BPMN : Notation des pools et lanes



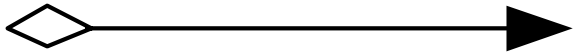
BPMN : Connections

- Flux de séquence 
  - Indique **l'ordre d'exécution** des activités dans un processus
- Flux de message 
  - Indique le flux de messages **entre** 2 participants d'un processus (**2 pools**)
- Association 
  - Associe une **information ou des artifacts** (ex : **annotations textuelles**) aux éléments graphiques du BPMN
  - Peut être orientée : indiquer la direction du flux

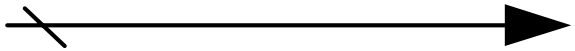
BPMN : Types de flux de séquence

- Flux **Normal**/incontrôlé : attaché directement à 1 activité/sans qu'il soit impacté par 1 condition

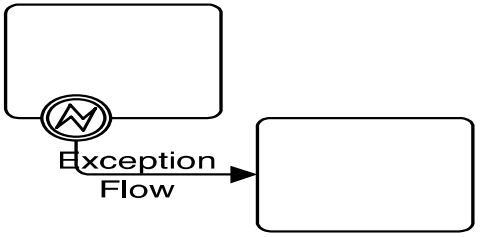
- Flux **conditionnel** : pris en compte quand une condition associée est validée (débute par un losange)



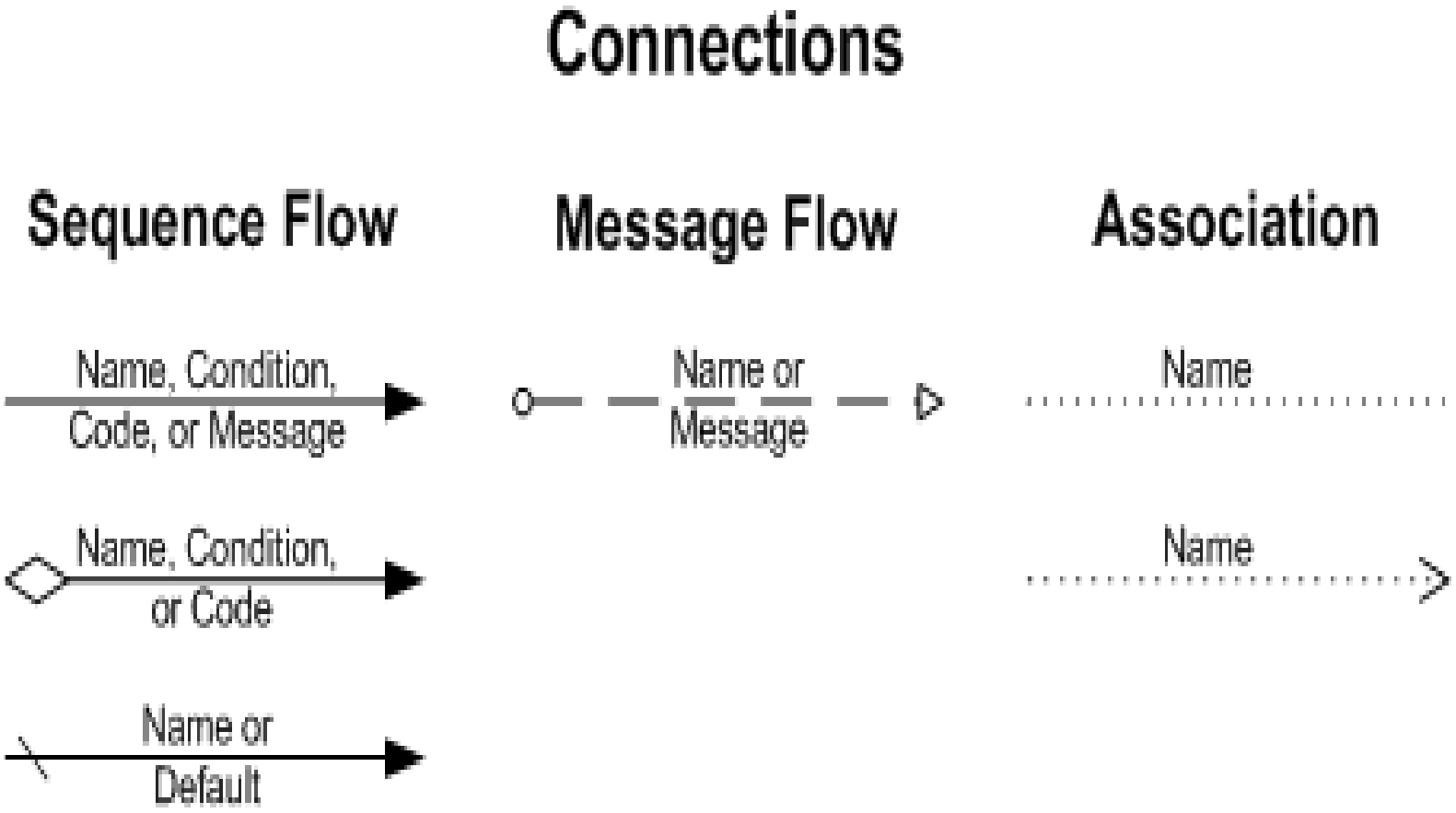
- Flux **par défaut** : pris en compte quand aucune des conditions associées à d'autres flux alternatifs n'est validée (exprime sinon)



- Flux d'exception : flux exceptionnel suite à 1 événement intermédiaire survenant lors de l'exécution



BPMN : Liste synthétique des connections



BPMN : Passerelles (Gateways)

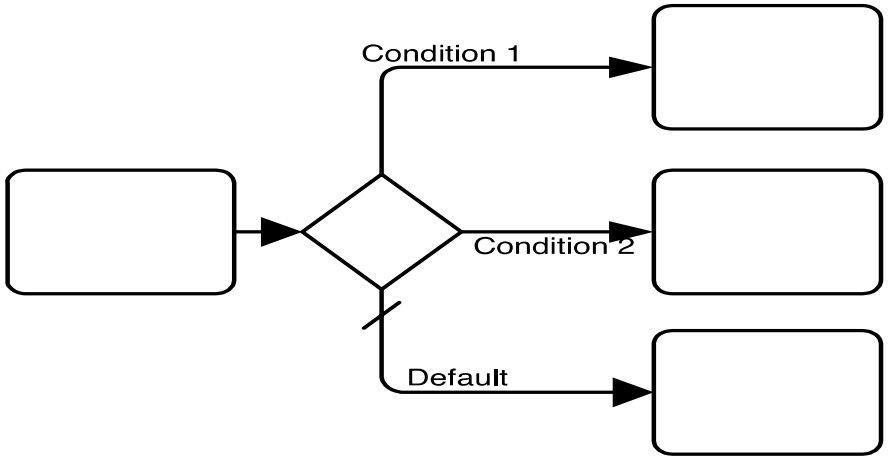
- Objets de contrôle permettant de **contrôler l'aiguillage du flux** du processus

  – Utilisées pour contrôler la convergence ou la divergence des flux de séquence

  – Déterminent le **type de comportement** du contrôle de flux : **branchement**, **forking**, fusion ou jointure…

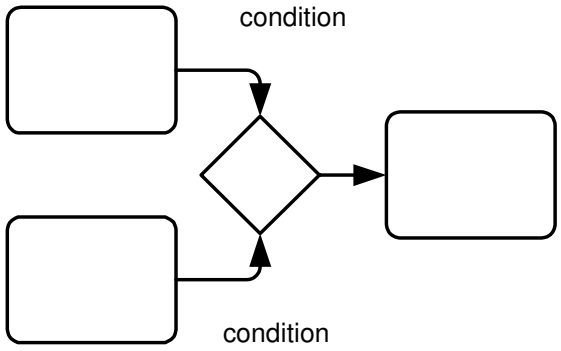
BPMN : Types de passerelles (1/7)

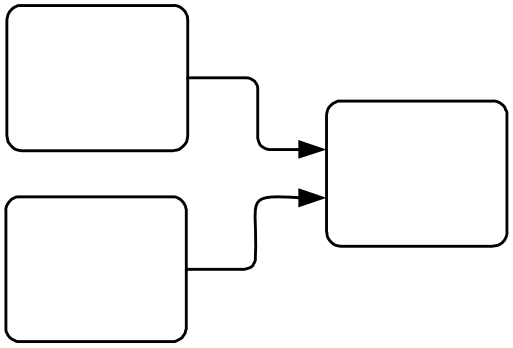
- Choix exclusif XOR : Exclusive data-based decision gateway
  - Point de fusion (Merge) XOR : point de branchement donnant lieu à des flux sortants alternatifs conditionnels, **mutuellement exclusifs**
  - 1 seul chemin sera choisi quand la condition qui lui est associée est validée (switch)



BPMN : Types de passerelles (2/7)

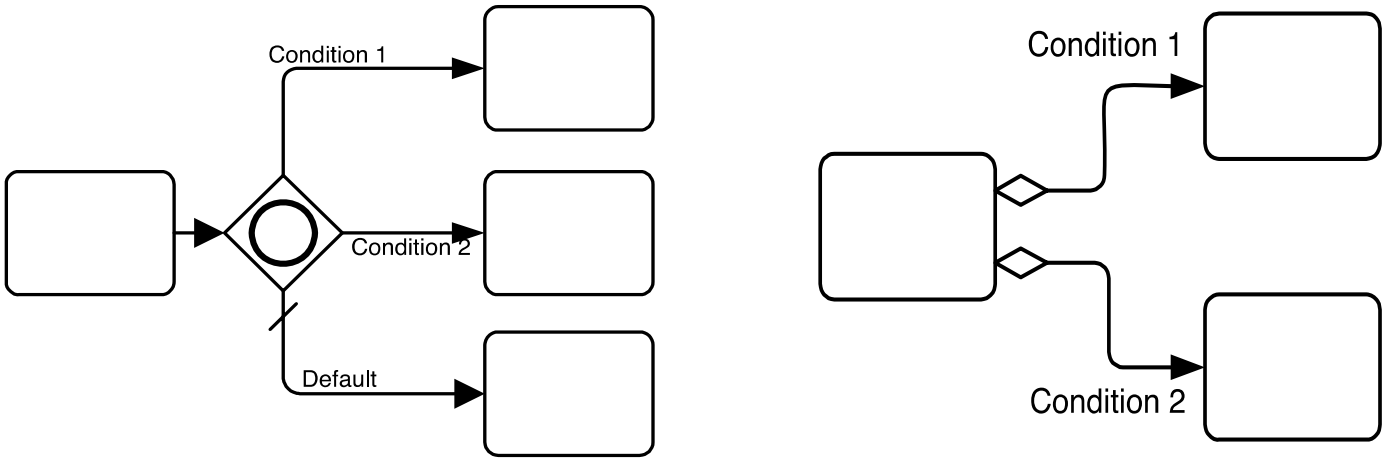
- Choix exclusif OR : Exclusive data-based decision gateway
  - Point de synchronisation (Join) OR : point de branchement donnant lieu à une **combinaison exclusive** de plusieurs flux entrants (1 ou +eurs cond. nécessaires pour déclencher le flux sortant)
  - Pas de gateway quand les **flux entrants sont alternatifs** (1 seul aura lieu pour déclencher le flux sortant)





BPMN : Types de passerelles (3/7)

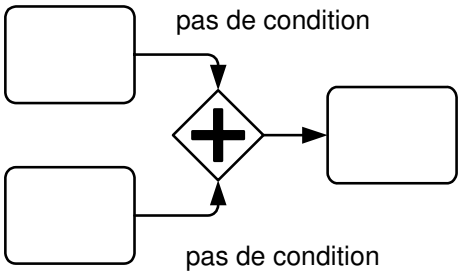
- Choix multiple : Inclusive decision gateway
  - Point de fusion (Merge) OR : point de branchement qui donne lieu à des flux conditionnels **parallèles** (concurrents)
  - Tous les flux dont les conditions associées sont validées seront choisis (Si)
  - Conditions portant sur des données
  - Une **exception** d'exécution se produit si aucune condition n'est validée



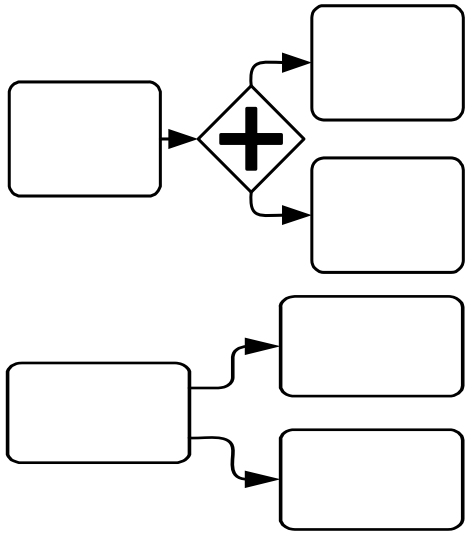
BPMN : Types de passerelles (4/7)

- Synchronisation (Join) et Branchement parallèle (Fork) : Parallel gateway
  - Synchronise (combine) ou crée des flux parallèles
  - Join : attend tous les flux parallèles entrants avant de déclencher le flux sortant
  - Fork : les flux sortants s'exécutent simultanément

Join

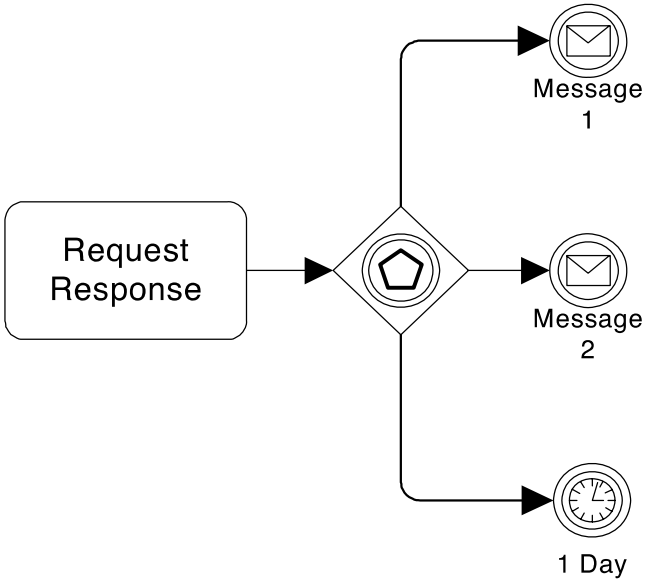


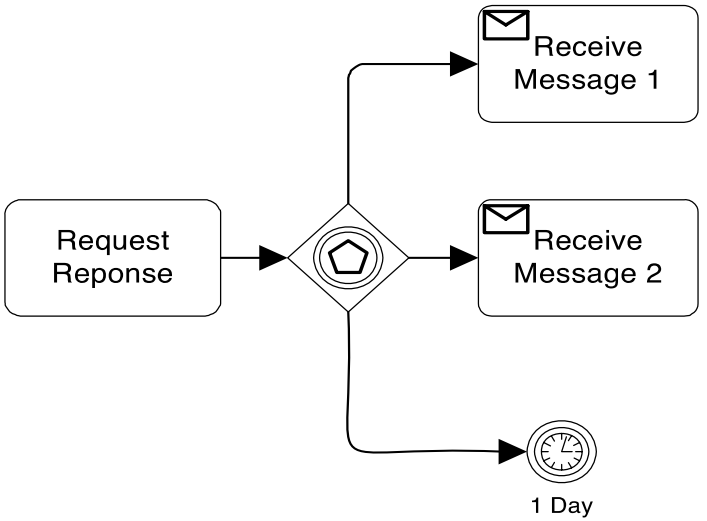
Fork



BPMN : types de passerelles (5/7)

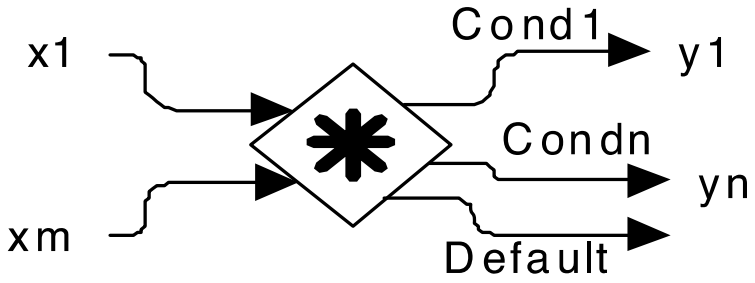
- Choix à base d'événements : Event-based gateway
  - Point de branchement où les flux alternatifs sortants sont exécutés quand des évènements associés surviennent
  - Le choix peut être exclusif ou parallèle





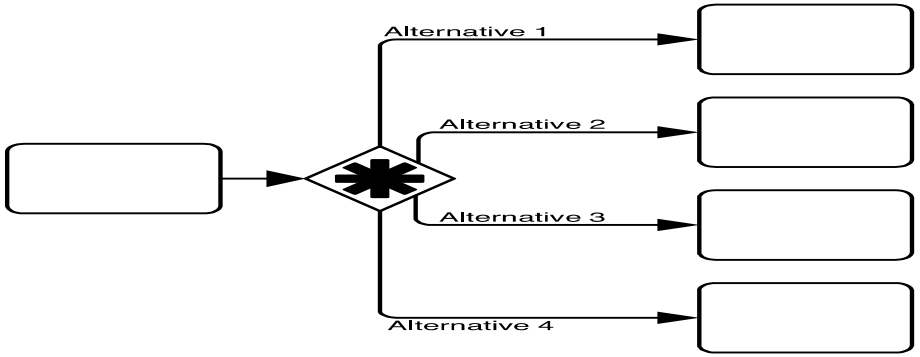
BPMN : Types de passerelles (6/7)

- Branchement complexe : Complex gateway
  - Modélise une **combinaison complexe** de synchronisation et de fusion
  - Indique que les flux sortants sont des flux concurrents, 1 ou +eurs parmi eux seront choisis si leurs conditions associées sont validées et que leur choix dépend uniquement de ces conditions et ne requiert pas obligatoirement d'attendre tous les flux entrants
  - Certains parmi les flux entrants sont nécessaires pour déclencher des flux sortants conditionnels concurrents

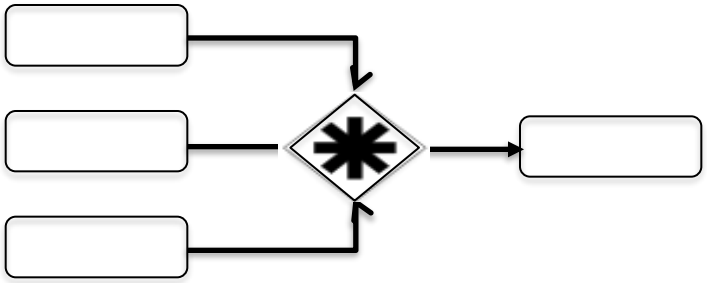


BPMN : Types de passerelles (7/7)

- Branchement complexe (suite) :
  - Pattern équivalent au choix multiple (Inclusive gateway)

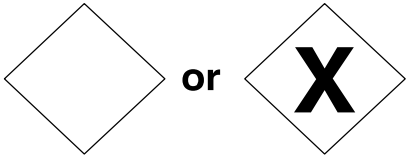


  - Demande de crédit à faire approuver par l'1 de 3 responsables

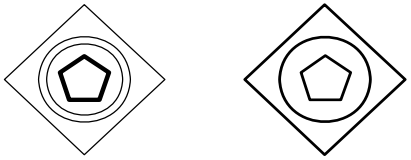


BPMN : Liste synthétique des passerelles

Exclusive Data-based : XOR Merge/OR Join

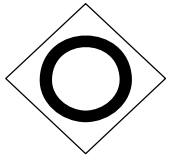


Exclusive Event-based : XOR Merge

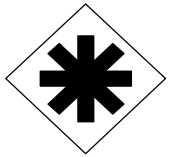


Parallel Event-Based

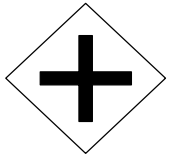
Inclusive : OR (Merge)



Complexe (Merge)



Parallel : AND (fork/Join)

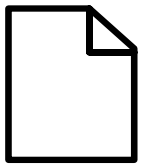


BPMN : Artifacts

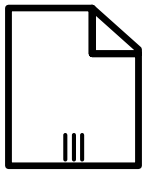
- Objet de données (Data object)
  - **Ressource** de données requises et/ou produites par le processus (ex : BD)
  - Peut être un simple objet ou une collection d'objets
- Data input et Data Output
  - Données entrées et sorties du processus
- Annotation
  - Fournit une information textuelle additionnelle pour décrire un diagramme BPMN
- Groupe
  - Regroupement d'éléments graphiques appartenant à 1 même catégorie qui constitue le label du groupe
  - Utilisé pour des fins de documentation ou d'analyse

BPMN : Liste synthétique des artifacts

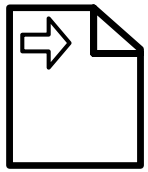
Data Object



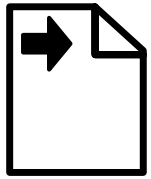
Data Collection



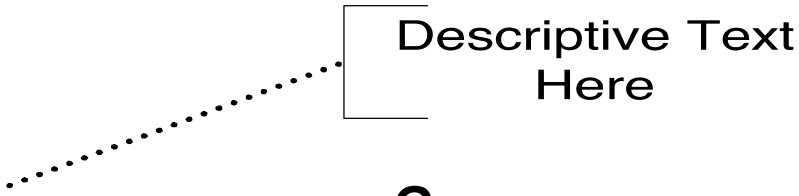
Data Input



Data Output

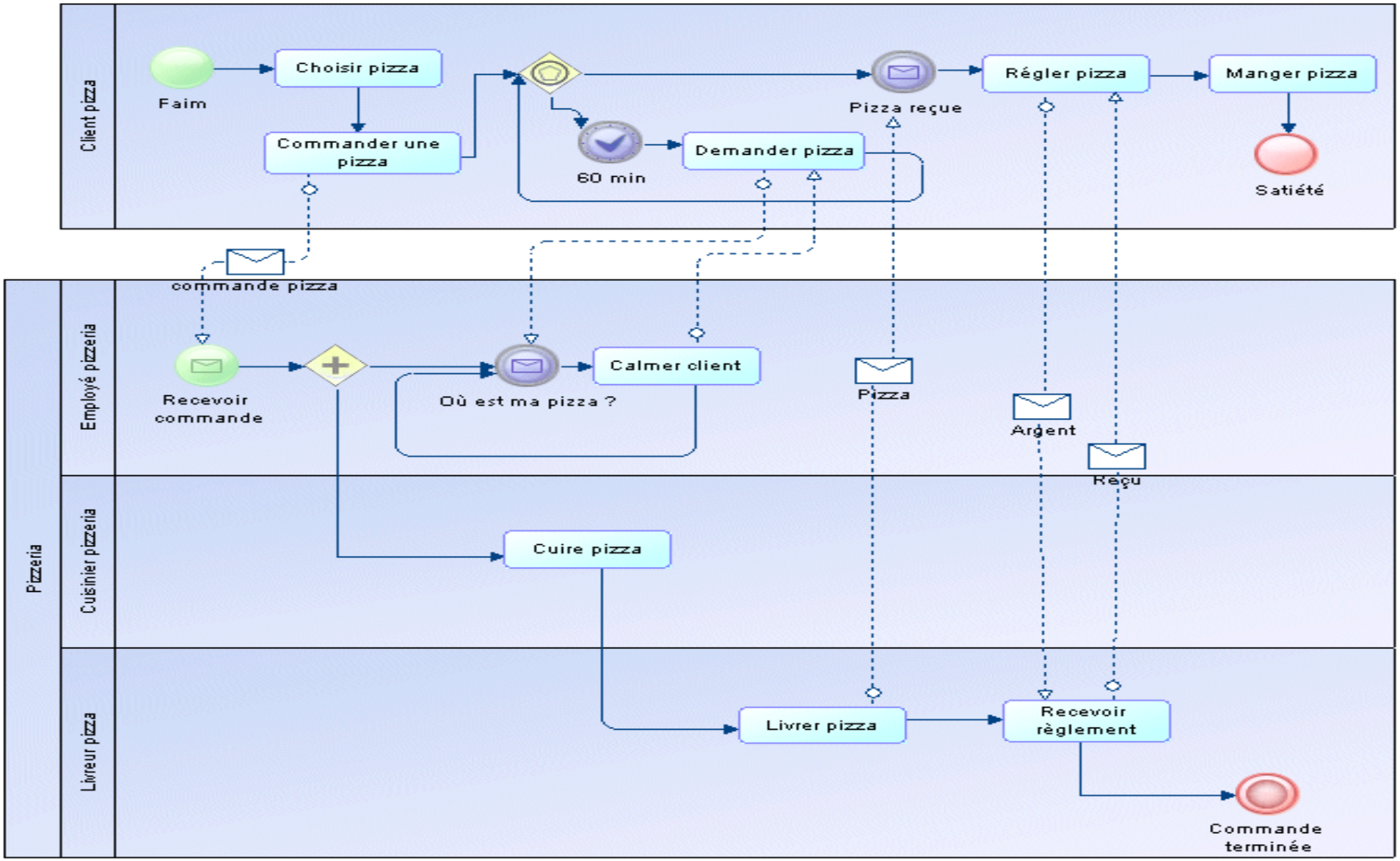


Annotation



Group

Exemple en BPMN



BPMN : Outils de modélisation*

- Free MicrosoftVisio BPMN
- BPMN Modeler Eclipse
- BPMN Studio Pro
- Intalio…

Partie 2 : SOA et Services Web

Partie 2: Plan

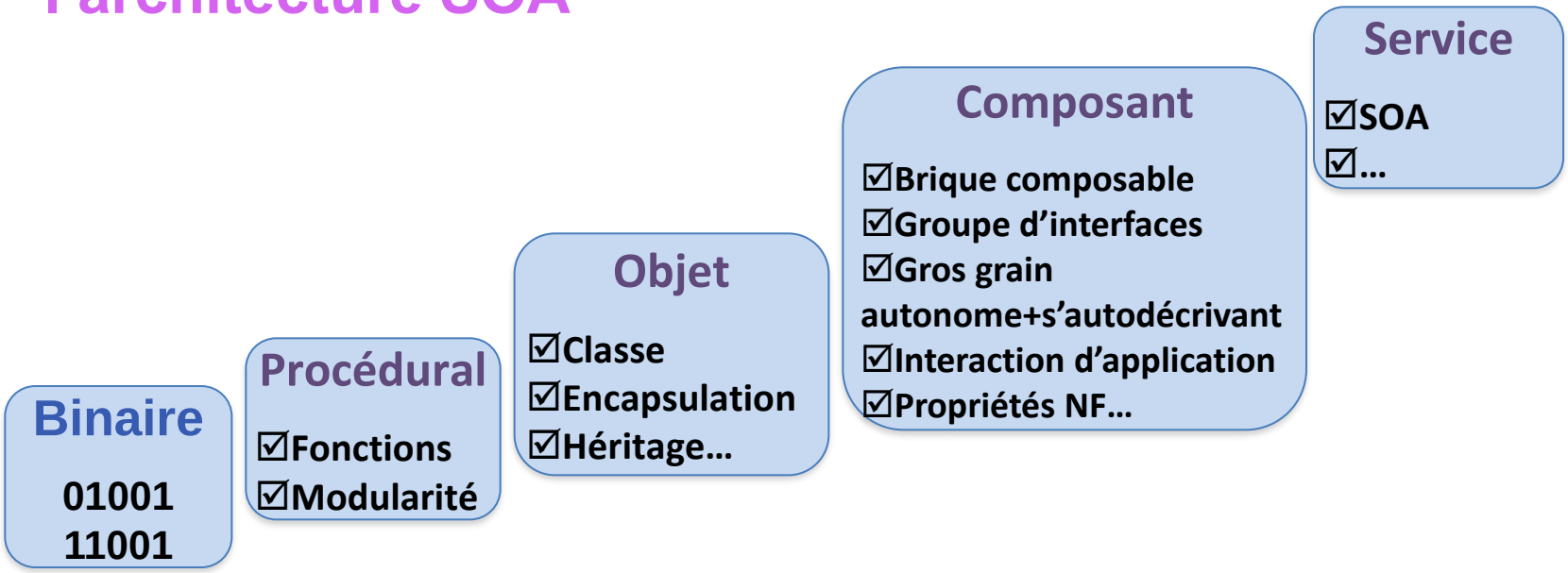
- Le concept Service
- L'architecture SOA et les Service Web
- Le standards des services Web
- L'API JAX-WS

Ch3. Le concept Service

- Evolution des paradigmes de développement
- Qu'est ce qu'un service?
- Orchestration des services
- Types de services
- Propriétés du service

Evolution des paradigmes de développement

- La conception d'un programme informatique s'effectue conformément à un paradigme de développement (PD)
- Un **PD définit un concept pour représenter** le monde **et des techniques pour traiter** ce concept
- Différents PD ont vu le jour et ont évolué du binaire, à différents **modèles de programmation** puis à **l'architecture SOA**



Concept Service

- **Composant logiciel** qui exécute une action pour le compte d'un client
- Il traduit le **niveau logique** d'accès aux traitements, plutôt que le niveau physique d'implémentation (EJB, Servlet…)

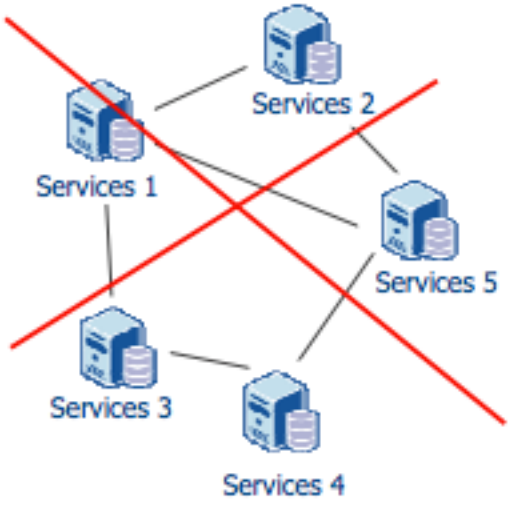
Définition du Service

- **Composant logiciel** :

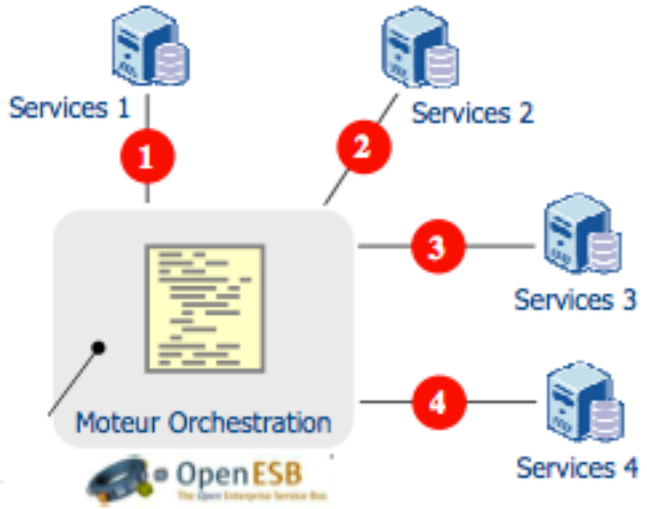
- **Mutualisé** (partagé puisqu'il est réutilisable et interopérable)
- **Référencé** dans un annuaire (où il est identifié)
- **Normalisé** (toutes ses fonctions sont appelées de la même façon via des paramètres, conformément à un **contrat**)
- Décrit par une **interface d'appel** (par un langage indépendant des technologies)
- Communicant avec le client par le biais de **messages (E/S)**
- **Neutre** (son utilisation est indépendante de son implémentation ou évolution tant que le contrat est respecté)
    - **Couplage faible :** interface isole le client du service
- **Déployé** (physiquement) sur un **serveur**

Orchestration des services

- Les services peuvent être composés (agrégés) dans le but de réaliser un processus donné
- L'orchestration leur permet de **communiquer sans avoir à se connaître** pour préserver leur couplage lâche (leur indépendance)
- **Un moteur d'orchestration se charge d'appeler les services** selon l'enchaînement désiré



Couplage fort



Couplage lâche

Types de services

- **Applicatif**
- **Fonctionnel**
- **Entité (CRUD) :** Create, Read, Update and Delete
- **Transverse (Infrastructure)**
- **Host**

Propriétés des services

- **Réutilisables** et possèdent des **contrats standardisés**
- Communiquent par **messages** à travers des **interfaces adressables**
- **Abstraits et prédictibles**
- **Modulaires** et de **large granularité**
- **Autonomes** et **sans état (stateless)**
- Moyens pour assurer une haute **Interopérabilité**
- **Faiblement Couplés**
- **Découvrables** (dynamiquement)
- **Composables**

Réutilisabilité par contrat

- Le service est réutilisable conformément à un contrat entre le fournisseur et le consommateur
- Le **contrat** décrit :

- La syntaxe du service : opération, input, output, format, protocole…
- La sémantique de son utilisation: pré-conditions, post-conditions…
- Sa QOS : temps de réponse attendu, temps de reprise après interruption…

- Le contrat est généralement décrit au moyen du standard WSDL
- Plusieurs contrats peuvent être définis pour répondre aux besoins différents des consommateurs (ex : service avec haute disponibilité/disponibilité normale)
- Le contrat est utilisé au design-time (génération de code) et au run-time (contrôle du respect du contrat)

Interface adressable et communication par message

- Chaque consommateur peut invoquer un service via son **adresse dans le réseau** à n'importe quel moment
- Le consommateur peut accéder localement au service pour augmenter la performance, s'ils sont hébergés dans la même machine

- Les services communiquent **uniquement par messages**
- Appels via le réseau vu que les services sont distribués en SOA

- Pour augmenter la performance, les concepteurs doivent penser à augmenter la granularité des interfaces de services pour diminuer le nombre d'appels réseau

Abstraction et Prédictibilité

- Le service fonctionne en « **boîte noire** »

- Seul le contrat du service (informations nécessaires pour l'invocation) est exposé au consommateur du service

- le fonctionnement interne du service (sa logique métier et son implémentation) ne sont pas visibles

- Il est **Prédictible**

- Son comportement et sa réponse lors de la réception d'une requête ne varient pas

Large granularité et modularité

- **Large granularité :** Le service est un **gros grain** qui regroupe un **ensemble d'interfaces cohérentes** se rapportant à un même module fonctionnel

- Principe à respecter lors de la conception

- **Modularité :** Il peut être **déployé de façon atomique** bien avant le développement ou déploiement d'applications consommatrices

- Principe différent du principe du paradigme OO où un programme OO est une unité indivisible

Autonomie et statelessness

- **Autonomie :**
  - Le service est **Indépendant des services externes** : son comportement est indépendant du contexte fonctionnel et technique dans lequel il a été invoqué
- **Statelessness :** Il est **sans état** (stateless) càd il n'intègre pas la gestion de contexte (puisqu'il est autonome)
  - But : Ne pas compliquer la maintenance, préserver la réutilisabilité et assurer la performance (minimiser la consommation de ressources systèmes, utilisées pour le stockage d'états)

Interopérabilité

- **Possibilité de communiquer avec un système hétérogène**

- Le service **précise un type de connecteur** (càd protocole et format de données) que ses clients potentiels doivent utiliser pour pouvoir invoquer l'interface qu'il fournit

- Une **spécification de médiation** permettra de réaliser le mapping au cas où le client adopte un format et types de données hétérogènes

- Mapping entre deux jeux de caractères comme l'ASCII et EBCDIC, et mapping de types de données
- Exemples de spécification de médiation : Les API JAX-RPC et JAXM pour le mapping des types de données Java aux types de données SOAP et XML dans le cas d'un service Web

Couplage faible (lâche)

- **Dépendance faible entre le consommateur et le service due à :**

- **Dépendance du contrat** et non pas de l'implémentation
- Echange à travers des **messages**
- **Orchestration** assure l'indépendance des services vu qu'elle leur permet de communiquer pour réaliser un processus, sans avoir à se connaître

- Avantage : **Maintenance facile**

- un changement dans le service suscite peu de changements dans ses consommateurs (juste ceux relatifs au respect du contrat)

Découvrabilité

- Il est **publié** par le fournisseur dans un annuaire : décrit par un ensemble de métadonnées qui permettent de l'identifier et qu'il est possible de màj
- Le consommateur peut **chercher** un service selon un ensemble de critères à partir de l'annuaire :

- L'annuaire renvoie au consommateur la liste des services (adresses, frais…) qui répondent à sa requête

- Tous les arguments nécessaires à l'exécution du service sélectionné (opérations, paramètres…) sont accessibles à partir de son contrat

Exemple : requête SOAP (1/2)

- Appel du service hel (opération : HelloWorld)

Message SOAP pour appeler l'opération makeHelloWorld avec un paramètre value

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:hel="http://helloworldwebservice.lisi.ensma.fr/">
    <soapenv:Header/>
    <soapenv:Body>
        <hel:makeHelloWorld>
            <value>Mickael BARON</value>
        </hel:makeHelloWorld>
   </soapenv:Body>
</soapenv:Envelope>
```

Exemple : requête SOAP (2/2)

- Appel du service hel (opération : HelloWorld)

Message SOAP pour appeler l'opération simpleHelloWorld, sans de paramètre

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:hel="http://helloworldwebservice.lisi.ensma.fr/">
    <soapenv:Header/>
    <soapenv:Body>
        <hel:simpleHelloWorld/>
    </soapenv:Body>
</soapenv:Envelope>
```

Exemple : Réponse SOAP

- Messages de Réponses aux appels du service HelloWorld

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body>
        <ns2:makeHelloWorldResponse xmlns:ns2="http://helloworldwebservice.lisi.ensma.fr/">
            <helloWorldResult>Hello World to Mickael BARON</helloWorldResult>
        </ns2:makeHelloWorldResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Les réponses sont sensiblement identiques

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body>
        <ns2:simpleHelloWorldResponse xmlns:ns2="http://helloworldwebservice.lisi.ensma.fr/">
            <helloWorldResult>Hello World to everybody</helloWorldResult>
        </ns2:simpleHelloWorldResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

```
<definitions
    name="Notebook"
    targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://notebookwebservice.lisi.ensma.fr/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
    <types>
        <xsd:schema targetNamespace="http://notebookwebservice.lisi.ensma.fr/">
            <xsd:complexType name="person">
                <xsd:sequence>
                    <xsd:element name="address" type="xs:string" minOccurs="0"/>
                    <xsd:element name="birthyear" type="xs:string" minOccurs="0"/>
                    <xsd:element name="name" type="xs:string" minOccurs="0"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="personArray" final="#all">
                <xsd:sequence>
                    <xsd:element name="item" type="tns:person" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:schema>
    </types>
    ...
</definitions>
```

Une personne est définie par une *adresse*, une *année de naissance* et un *nom*

Définition d'un type tableau de personne

```
<definitions
      targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
      name="Notebook"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      ...>
      <types>
            <xsd:schema>
                  <xsd:import namespace="http://notebookwebservice.lisi.ensma.fr/" schemaLocation="Notebook_schema1.xsd"/>
            </xsd:schema>
      </types>
      ...
</definitions>
```

Import le fichier XSD

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0"
      targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
      ...>
      <xs:complexType name="person">
            <xs:sequence>
                  <xs:element name="address" type="xs:string" minOccurs="0"/>
                  <xs:element name="birthyear" type="xs:string" minOccurs="0"/>
                  <xs:element name="name" type="xs:string" minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="personArray" final="#all">
            <xs:sequence>
                  <xs:element name="item" type="tns:person" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
            </xs:sequence>
      </xs:complexType>
</xs:schema>
```

Types d'échanges assurés par une opération

- **One-way** : Message input sans réponse
- **Notification** : Seul un message <output> est utilisé
- **Request/Response** : <input>, <output> et <fault>

➢ Le service reçoit un message du client et répond à sa requête

- **Solicit - response** : <input>, <output> et <fault>

➢ Le client reçoit un message du service et répond au service

Exemples des différents Types d'échanges

One-way :

```
<operation name="addPerson" parameterOrder="name address birthyear">
    <input message="tns:addPersonWithSimpleType"/>
</operation>
```

Notification :

```
<operation name="personStatus">
    <output message="trackingInformation"/>
</operation>
```

Request/Response :

```
<operation name="addPerson">
    <input message="tns:addPersonWithComplexType"/>
    <output message="tns:addPersonWithComplexTypeResponse"/>
</operation>
```

Solicit - response :

```
<operation name="clientQuery">
    <output message="bandWithRequest"/>
    <input message="bandwidthInfo" />
    <fault message="faultMessasge" />
</operation>
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions name="HelloWorld"
    targetNamespace="http://helloworldwebservice.lisi.ensma.fr/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://helloworldwebservice.lisi.ensma.fr/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types/>
  <message name="makeHelloWorld">
    <part name="value" type="xsd:string"/>
  </message>
  <message name="makeHelloWorldResponse">
    <part name="helloWorldResult" type="xsd:string"/>
  </message>
  <message name="simpleHelloWorld"/>
  <message name="simpleHelloWorldResponse">
    <part name="helloWorldResult" type="xsd:string"/>
  </message>
  <portType name="HelloWorld">
    <operation name="makeHelloWorld">
      <input message="tns:makeHelloWorld"/>
      <output message="tns:makeHelloWorldResponse"/>
    </operation>
    <operation name="simpleHelloWorld">
      <input message="tns:simpleHelloWorld"/>
      <output message="tns:simpleHelloWorldResponse"/>
    </operation>
  </portType>
```

```
<binding name="HelloWorldPortBinding" type="tns:HelloWorld">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
  <operation name="makeHelloWorld">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/"/>
    </input>
    <output>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/"/>
    </output>
  </operation>
  <operation name="simpleHelloWorld">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/"/>
    </input>
    <output>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/"/>
    </output>
  </operation>
 </binding>
 <service name="HelloWorld">
  <port name="HelloWorldPort" binding="tns:HelloWorldPortBinding">
    <soap:address location="TODO"/>
  </port>
 </service>
</definitions>
```