SOA – Services Web

BPEL : Orchestrer les Services Web

Mickaël BARON – 2010 (Rév. Mars 2011)
mailto:baron.mickael@gmail.com ou mailto:baron@ensma.fr

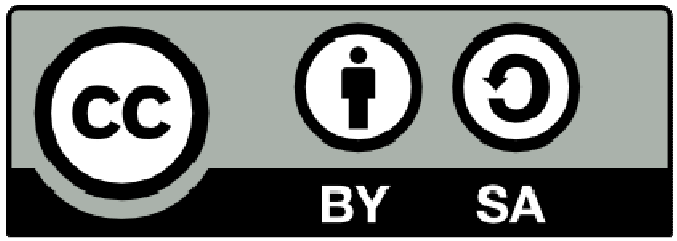
Licence

Creative Commons

Contrat Paternité

Partage des Conditions Initiales à l'Identique

2.0 France



http://creativecommons.org/licenses/by-sa/2.0/fr

Plan du cours

- Problématique
- Présentation BPEL
- BPEL par l'exemple : HelloWorld
- BPEL parties statique et dynamique
- Chorégraphie / Orchestration
- Structure d'un BPEL
- Partener Links
- Variables
- Activities

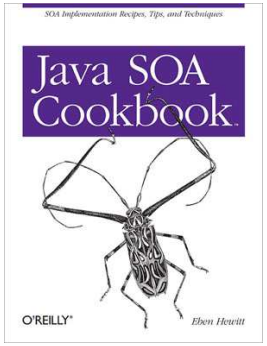
Déroulement du cours

- Pédagogie du cours
  - Illustration avec de nombreux exemples qui sont disponibles à l'adresse *http://mbaron.developpez.com/soa/bpel*
  - Des bulles d'aide tout au long du cours
  - Survol des principaux concepts en évitant une présentation exhaustive
  - Guidée par l'éditeur graphique Netbeans
- Logiciels utilisés    
  - Nav. Web, Netbeans 6.7.1, Glassfish 2.1.1 (OpenESB 2.2), SOAP UI
- Pré-requis
  - Ingénierie des données
  - Schema XML, WSDL, SOAP
- Remerciements
  - TODO

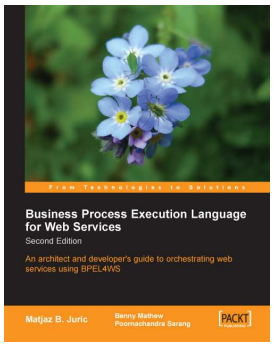
Ressources

- Billets issus de Blog
  - *jee-bpel-soa.blogspot.com/2008/09/apache-ode-first-bpel.html*
  - *blueprints.dev.java.net/bpcatalog/ee5/soa/*
- Articles
  - *docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.pdf*
  - *bpelsource.com*
  - *www.oracle.com/technology/pub/articles/matjaz_bpel2.html*
- Cours
  - *dlc.sun.com/pdf/820-6323/820-6323.pdf*
  - *www.javapassion.com/soaprogramming/BPELOverview.pdf*
  - *developers.sun.com/docs/javacaps/jbi/capsbpeldeseng.cnfg_bpel-se-blueprints_r.html*
  - *netbeans.org/kb/61/soa/understand-trs.html*

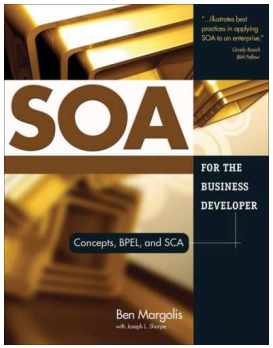
Ressources : Bibliothèque



- Java SOA Cookbook
  - Auteur : Eben Hewitt
  - Éditeur : O'Reilly
  - Edition : Mars 2009 - 752 pages - ISBN : 0596520727



- Business Process Execution Language for WS
  - Auteur : Benny Mathew, Matjaz Juric, Poornachandra …
  - Éditeur : Packt Publishing
  - Edition : January 2006 - 372 pages - ISBN : 1904811817



- SOA for the Business Developer : …, BPEL, …
  - Auteur : Benny Mathew, Matjaz Juric, Poornachandra …
  - Éditeur : Packt Publishing
  - Edition : January 2006 - 372 pages - ISBN : 1904811817

Problématique : Scénario

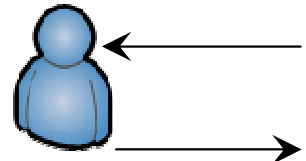
1. Inscription d'un étudiant à une école
   - Inscription à l'école (nom, adresse, …)
   - Payement des frais d'inscription (auprès des services d'une banque)
   - Attribution d'un numéro d'étudiant
2. Chercher un logement au CROUS
   - Inscription à partir du numéro d'étudiant
   - Choisir son type de logement
   - Payement des frais d'hébergement
   - Attribution d'un numéro CROUS
3. S'abonner au transport de bus
   - Sélectionner la ligne entre l'école et le logement
   - Choisir la réduction proposée en justifiant du numéro d'étudiant et du numéro CROUS
   - Inscription et payement

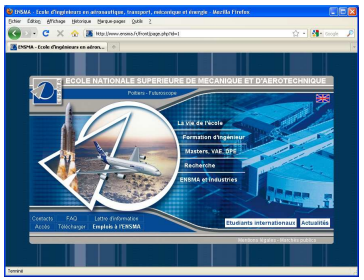
Problématique : Scénario

- Solution 1 : l'utilisateur doit se rendre sur chaque site Web des administrations

Se rendre sur le site de l'école

1



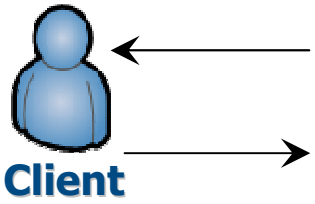


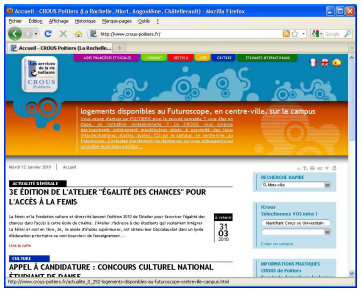
SI de l'Ecole

- Inscription
- Payement
- Générer numéro étudiant (INE)

Se rendre sur le site du CROUS

2



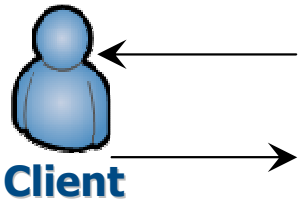


SI du CROUS

- Inscription à partir INE
- Payement
- Générer numéro CROUS

Se rendre sur le site de transport de BUS

3

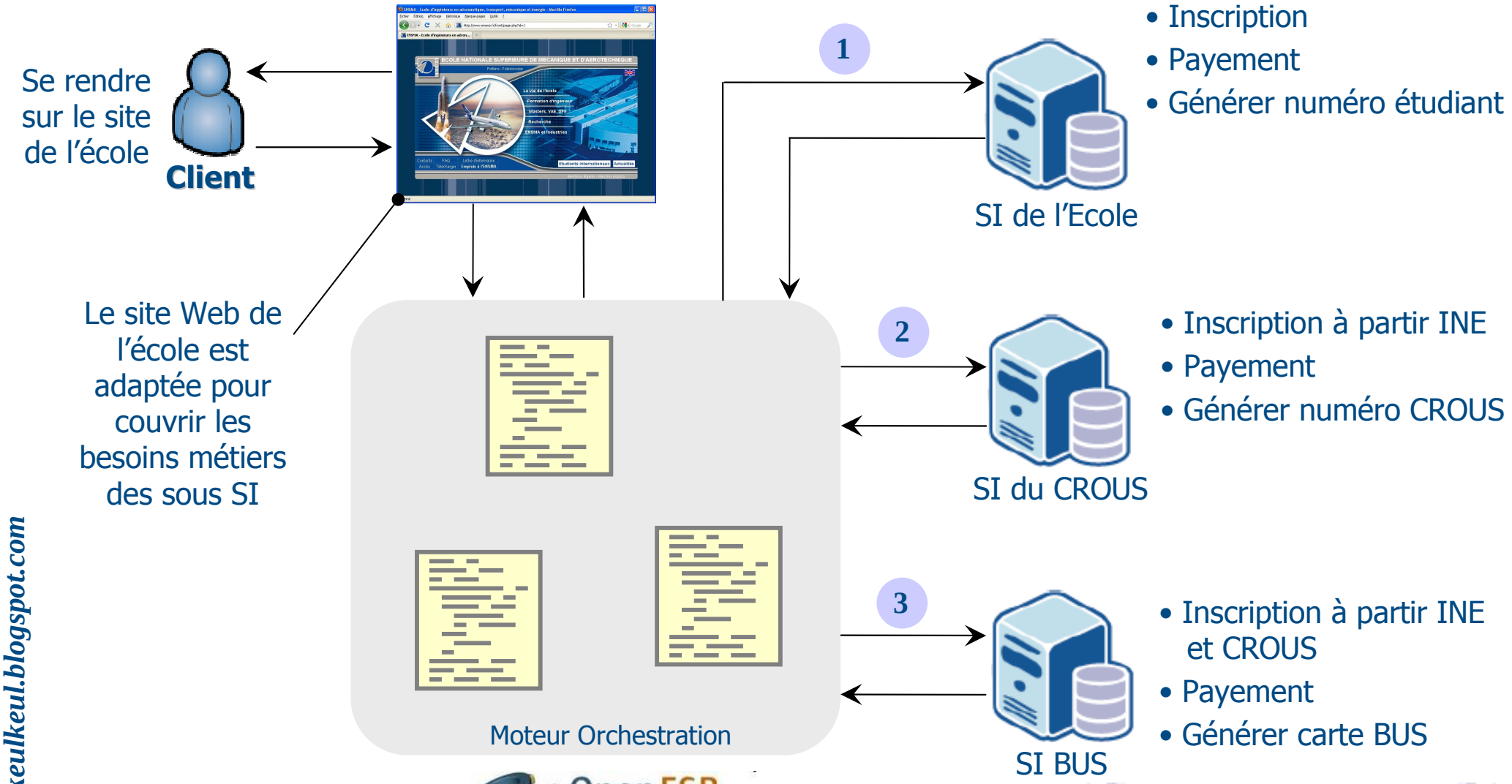


SI BUS

- Inscription à partir INE et CROUS
- Payement
- Générer carte BUS

Problématique : Scénario

- Solution 2 : l'utilisateur n'utilise que le site Web de l'école qui fournit un Workflow complet



Problématique

- Processus métiers sont de plus en plus complexes
  - Applications multiples à intégrer
  - Fort besoin de paralléliser les processus
  - Partenaires à intégrer sont nombreux
- Hétérogénéité des langages due aux plateformes de développement
  - Java 

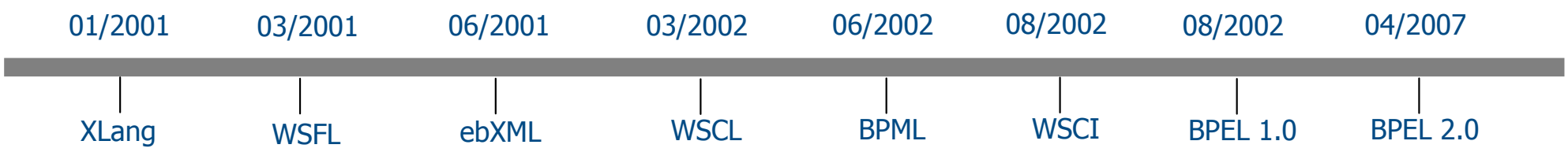
  - .NET 

  - PHP 

- Fort besoin d'évolution des processus
  - Partenaires peuvent évoluer
  - Intégration de nouveaux processus

Langages de processus métiers (Workflow)

- Différents langages permettent de modéliser les processus métiers



- Fondés sur des concepts proches
  - Activités, opérations, appels de services
  - Echanges des messages, modification des messages
  - Gestion des erreurs, événements
- Syntaxe basée sur le format XML
- Outillés
  - Editeurs graphiques
  - Moteur d'exécution

Présentation : WS-BPEL

- WS-BPEL (ou BPEL en abrégé) est l'acronyme de **B**usiness **P**rocess **E**xecution **L**anguage
- Son origine est issu des langages WSFL et XLANG
- Standard OASIS en V2.0
- Basé sur les standards du Services Web étendus (WSDL)
- Syntaxe basée sur XML
- Langage graphique
- Fournit les opérateurs classiques d'un langage de programmation (séquence, parallèle, boucle, …)
- Gestion des erreurs

Pour faire du BPEL : Outils

- Deux catégories d'outils sont à distinguer
  - Editeur graphique de processus BPEL
  - Moteur BPEL intégré dans la majorité des serveurs d'application
- A noter que les solutions BPEL proposées par les éditeurs de logiciels fournissent à la fois l'éditeur et le serveur
- Les éditeurs graphiques sont généralement intégrables dans les environnements de développement (Eclipse, Netbeans, Visual Studio, …)
- Dans ce cours, nous utiliserons une solution libre basée sur le moteur OpenESB et le module graphique fourni par Netbeans

Pour faire du BPEL : Outils … des solutions

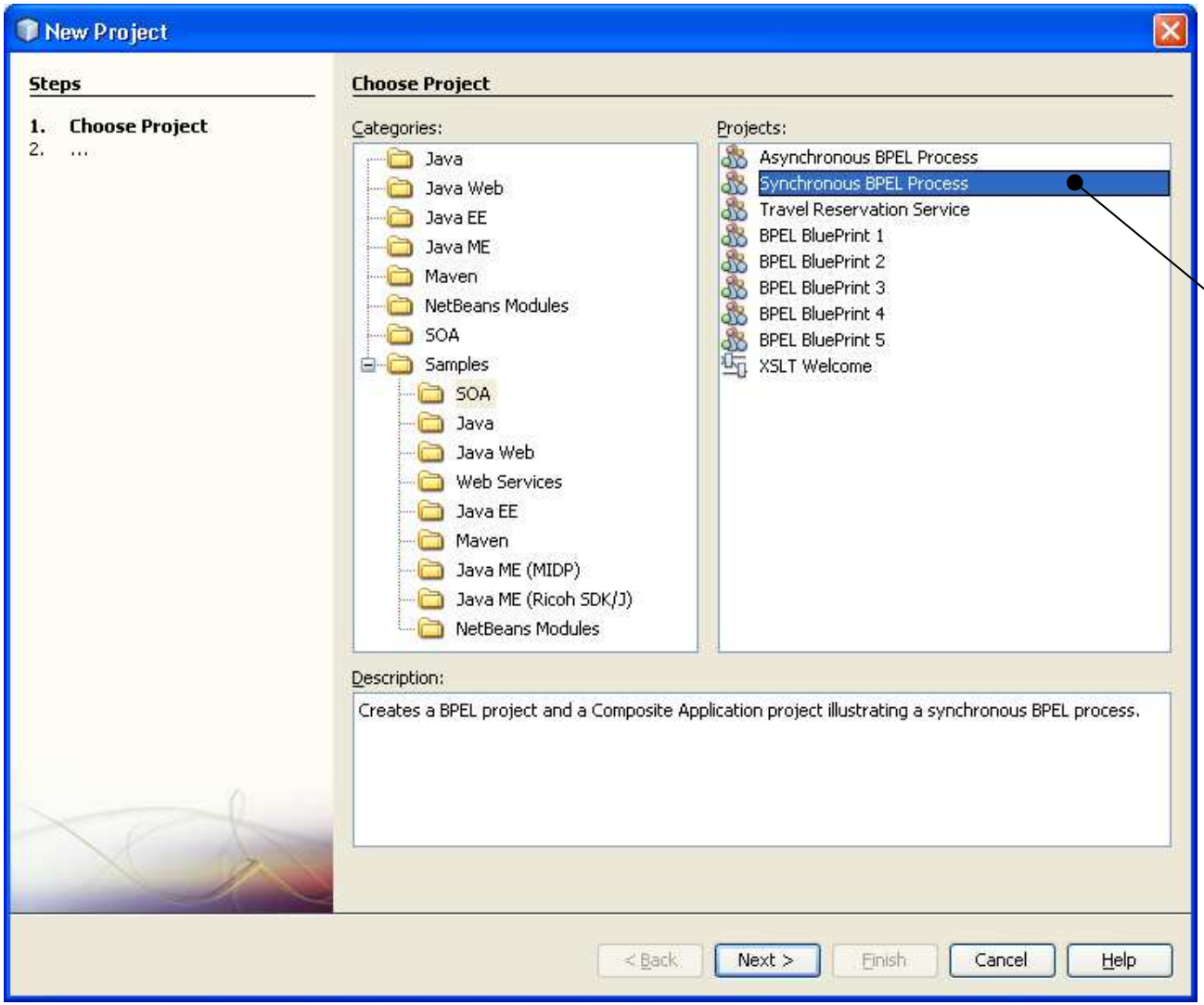
- OpenESB (via Glassfish 2.1) et Netbeans 6.7.1
  - *https://open-esb.dev.java.net/*
- Apache ODE et Eclipse BPEL Editor
  - *http://ode.apache.org/* et *http://www.eclipse.org/bpel/*
- JBoss jBPM (uniquement le moteur BPEL)
  - *http://www.jboss.com/products/jbpm/*
- Oracle BPEL Process Manager (Weblogic + JDeveloper)
  - *http://www.oracle.com/technology/bpel*
- Orchestra
  - *http://orchestra.ow2.org*
- IBM (Weblogic + Workshop)
- BPEL for Windows Workflow Foundation (Visual Studio et Biztalk)

BPEL par l'exemple : HelloWorld

- Pour introduire la présentation du langage BPEL nous construisons un processus *HelloWorld*
- Le processus BPEL est décrit par le Service Web suivant
  - Une opération *makeHello* qui prend en paramètre une chaîne de caractère et retourne une chaîne de caractère
- Le processus BPEL traite le Workflow suivant
  - Récupération du message envoyé par le client (chaîne de caractères)
  - Transformation du message en ajoutant le texte *HelloWorld*
  - Retourner le nouveau message au client

BPEL par l'exemple : HelloWorld

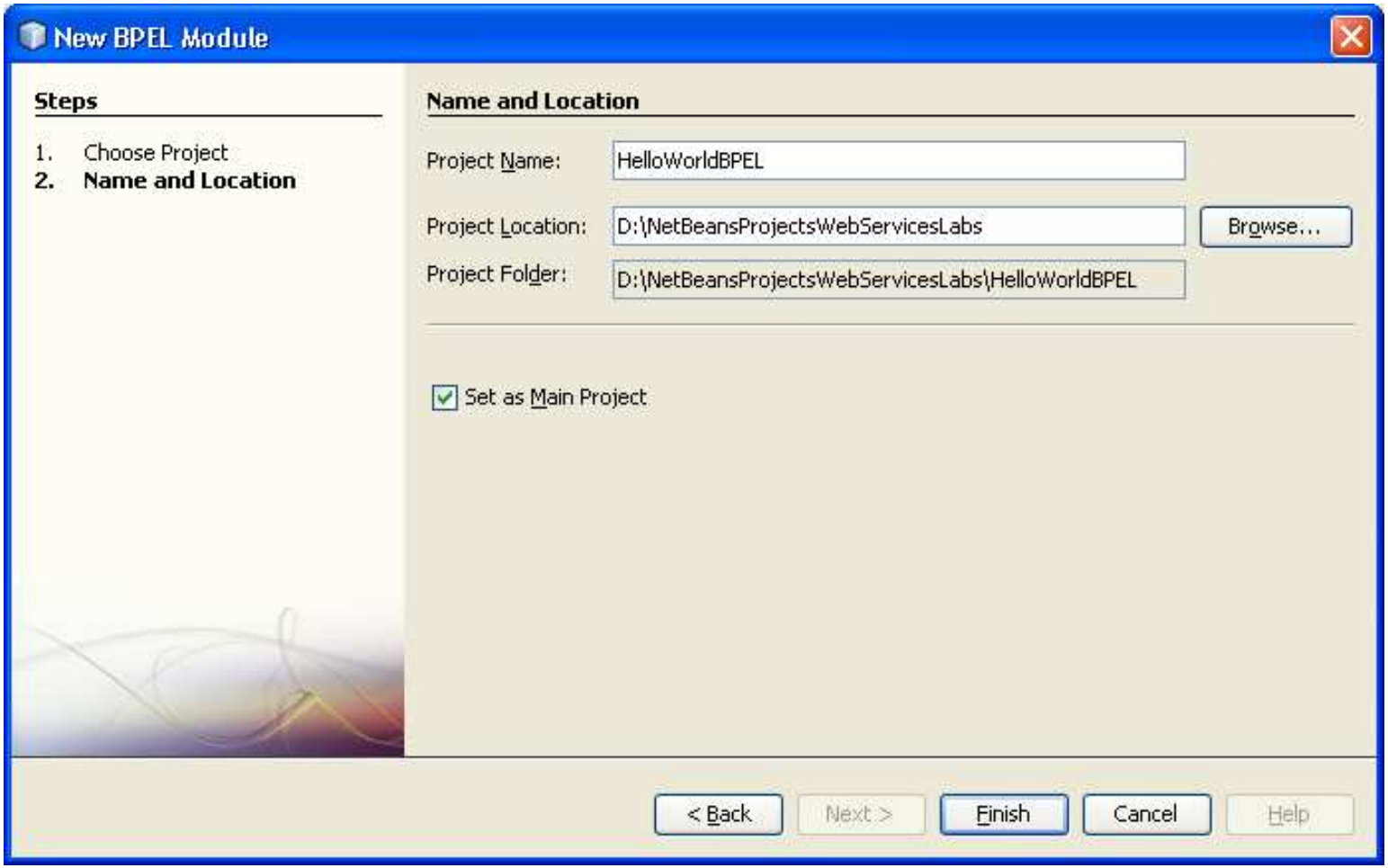
- Création d'un nouveau projet BPEL (*File -> New Project*) à partir des exemples (Samples)



Choisir cet exemple « Synchronous BPEL Process »

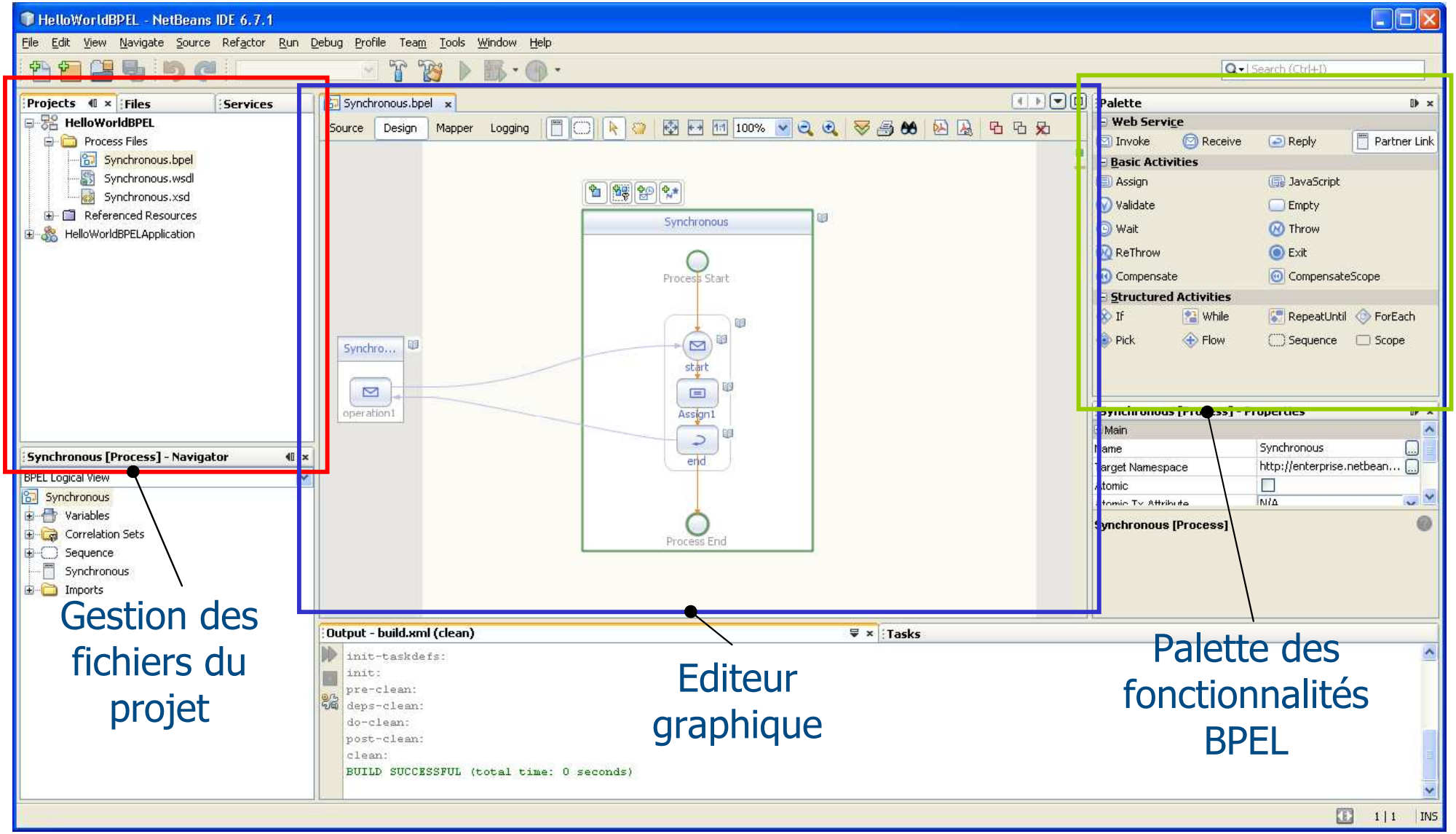
BPEL par l'exemple : HelloWorld

- Préciser les caractéristiques physiques du projet BPEL



BPEL par l'exemple : HelloWorld

Génération des fichiers et initialisation des outils BPEL



BPEL par l'exemple : HelloWorld

Edition du BPEL en mode « texte »

```
<process name="Synchronous">
     <import namespace="http://localhost/Synchronous/Synchronous"
            location="Synchronous.wsdl"
            importType="http://schemas.xmlsoap.org/wsdl/"/>
     <partnerLinks>
          <partnerLink name="Synchronous" partnerLinkType="ns1:partnerlinktype1"
               myRole="partnerlinktyperole1"></partnerLink>
     </partnerLinks>
     <variables>
        <variable name="outputVar" messageType="ns1:responseMessage"></variable>
        <variable name="inputVar" messageType="ns1:requestMessage"></variable>
     </variables>

     <sequence>
          <receive name="start" partnerLink="Synchronous" operation="operation1"
               portType="ns1:portType1" variable="inputVar" createInstance="yes">
          </receive>
          <assign name="Assign1">
               <copy>
                    <from>$inputVar.inputType/ns2:paramA</from>
                    <to>$outputVar.resultType/ns2:paramA</to>
               </copy>
          </assign>
          <reply name="end" partnerLink="Synchronous" operation="operation1"
               portType="ns1:portType1" variable="outputVar">
        </reply>
    </sequence>
</process>
```

Une relation de partenaire est définie

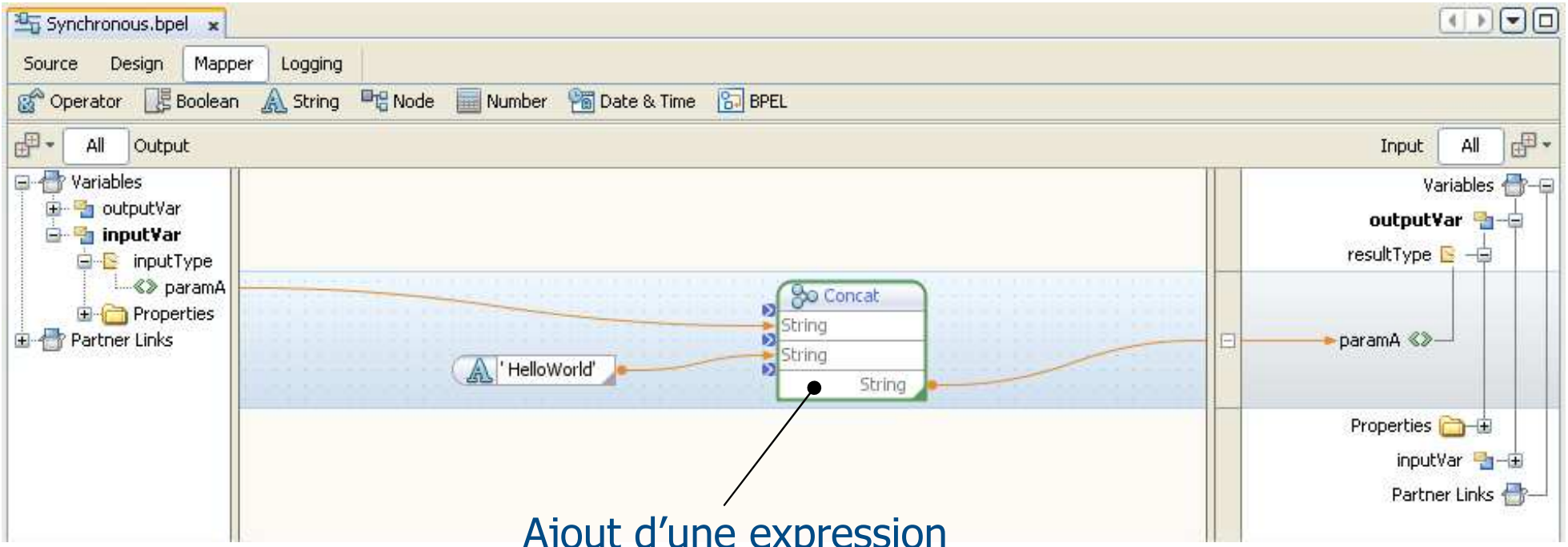
Deux variables sont définies

Trois activités déclenchées en séquence

Synchronous.bpel du projet **HelloWorldBPEL**

BPEL par l'exemple : HelloWorld

Extension de l'activité **Assign**



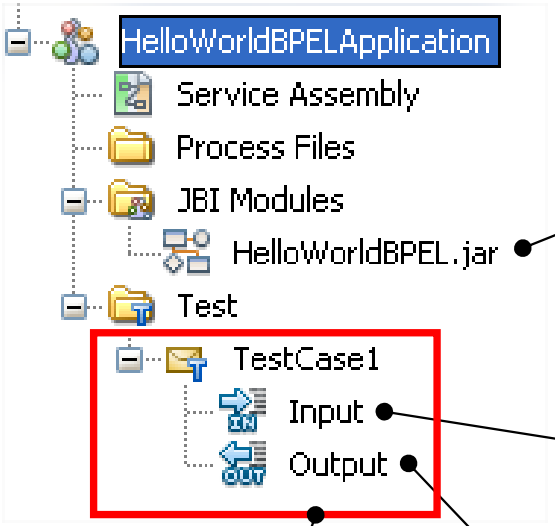
Ajout d'une expression

```
<assign name="Assign1">
     <copy>
          <from>concat($inputVar.inputType/ns2:paramA, ' HelloWorld')</from>
          <to>$outputVar.resultType/ns2:paramA</to>
     </copy>
</assign>
```

Le contenu de l'activité *assign* a été mis à jour

BPEL par l'exemple : HelloWorld

- Utilisation du projet **HelloWorldBPELApplication** pour le déploiement (type *Composite Application*)



Tous les projets BPEL peuvent être déployés comme module JBI

```
<soapenv:Envelope ...>
      <soapenv:Body>
            <syn:typeA>
                  <syn:paramA>Mickael BARON</syn:paramA>
            </syn:typeA>
      </soapenv:Body>
</soapenv:Envelope>
```

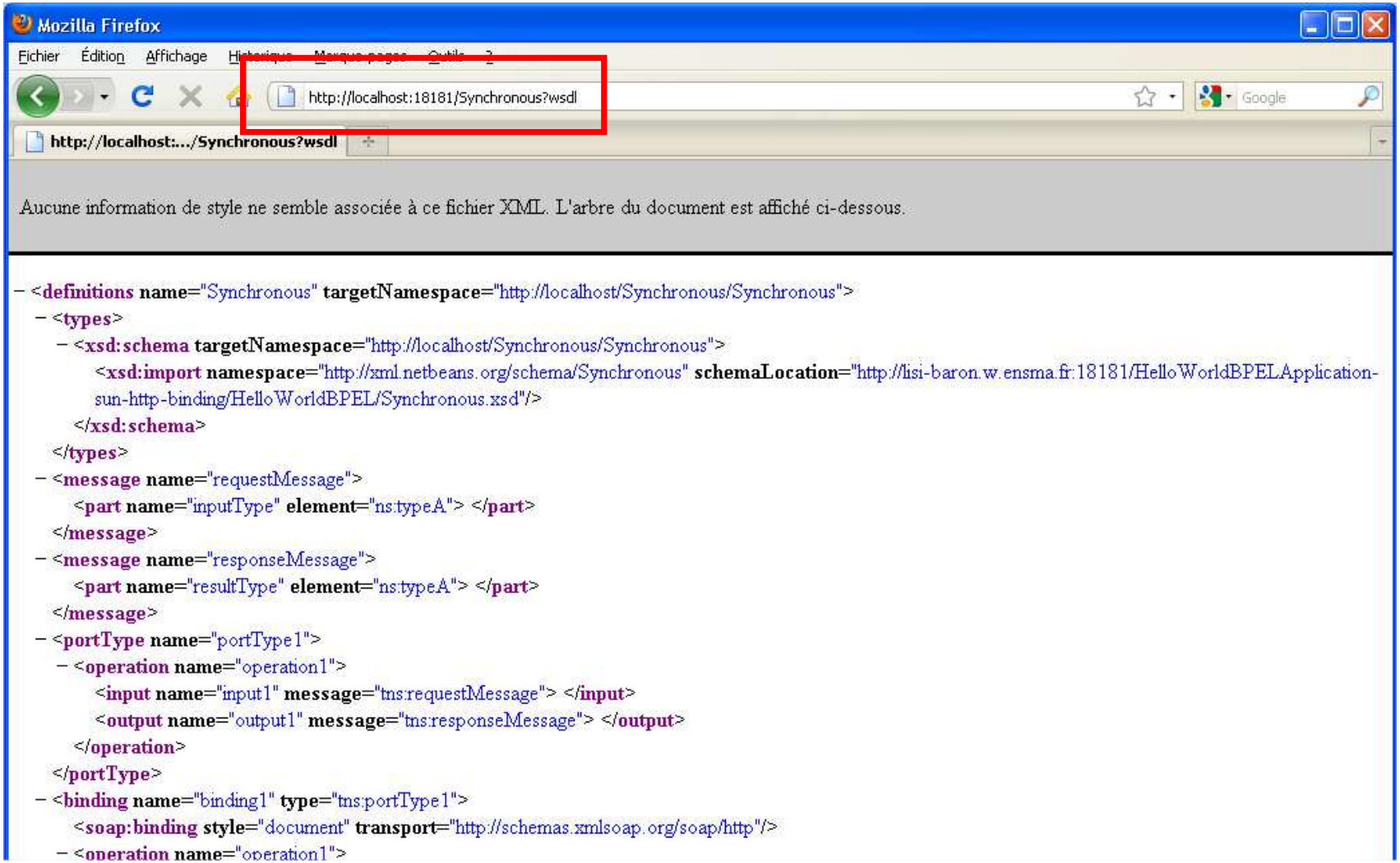
```
<SOAP-ENV:Envelope ...>
      <SOAP-ENV:Header/>
      <SOAP-ENV:Body>
            <typeA xmlns="http://xml.netbeans.org/schema/Synchronous">
                  <paramA>Mickael BARON HelloWorld</paramA>
            </typeA>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Des tests unitaires peuvent être définis

Le message d'entrée a été étendu

BPEL par l'exemple : HelloWorld

- Le processus BPEL est décrit par un WSDL et est accessible par un client Service Web (voir cours JAX-WS)

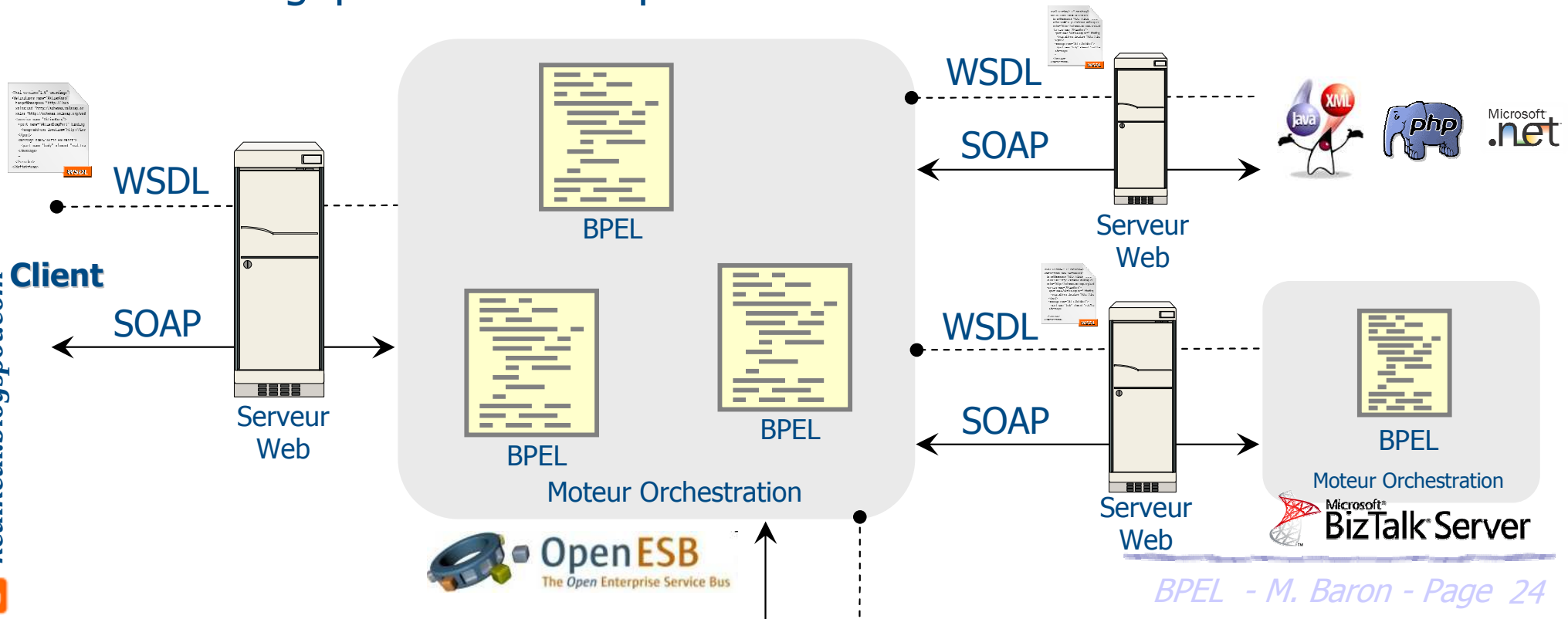


BPEL : partie statique

- Comme vu précédemment la partie **statique** d'un processus est définie par un document WSDL
- Le document WSDL permet de décrire
  - Les points d'entrées et de sorties du processus
  - Définir les types des données (XML Schema) et les messages utilisés pour décrire l'état du processus
  - Les opérations qui sont autorisées et qui permettent d'invoquer le processus
- Par conséquent tous les concepts introduits dans les précédents cours WSDL, SOAP et JAX-WS sont ré-utilisables
  - Création d'un client du processus à partir du WSDL (ws-import)
  - Envoie de messages SOAP

BPEL : partie dynamique

- La partie **dynamique** du processus est décrite par le fichier BPEL
- Ce document BPEL permet de décrire
  - L'ordonnancement des différentes sous étapes du processus
  - L'invocation vers les opérations des Services Web partenaires
  - La logique et l'état du processus

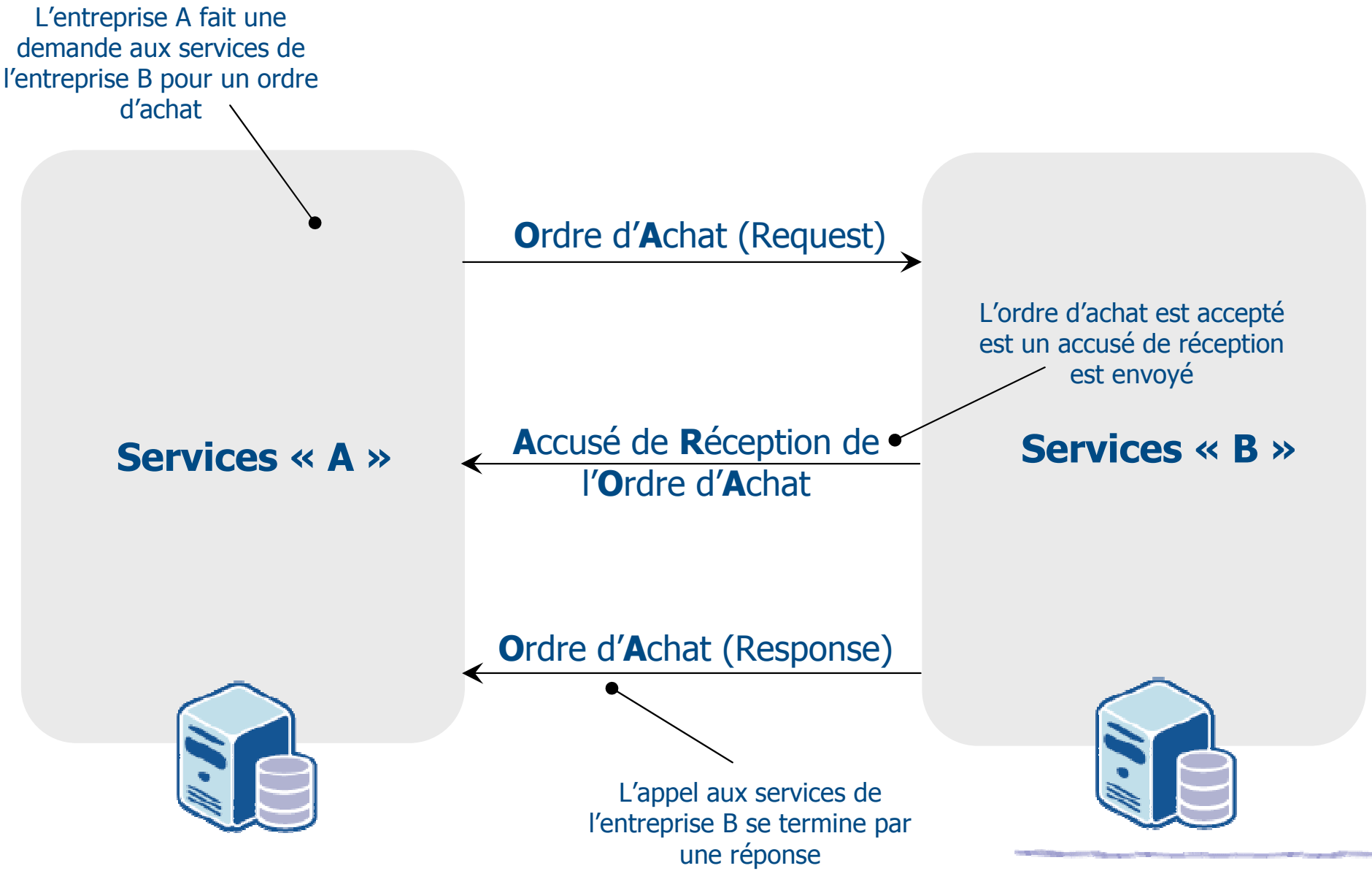


Chorégraphie / Orchestration

- **Chorégraphie**
  - Vue globale
  - Description des interactions entre plusieurs partenaires
  - Description des messages échangés entre les participants et leurs ordres
  - WSDL s'inscrit dans la description de la chorégraphie
  - La chorégraphie peut être décrite par le langage WS-CDL
- **Orchestration**
  - Vue locale à un processus
  - Description de la logique d'exécution des services partenaires et des messages échangés
  - BPEL s'inscrit dans la description de l'orchestration

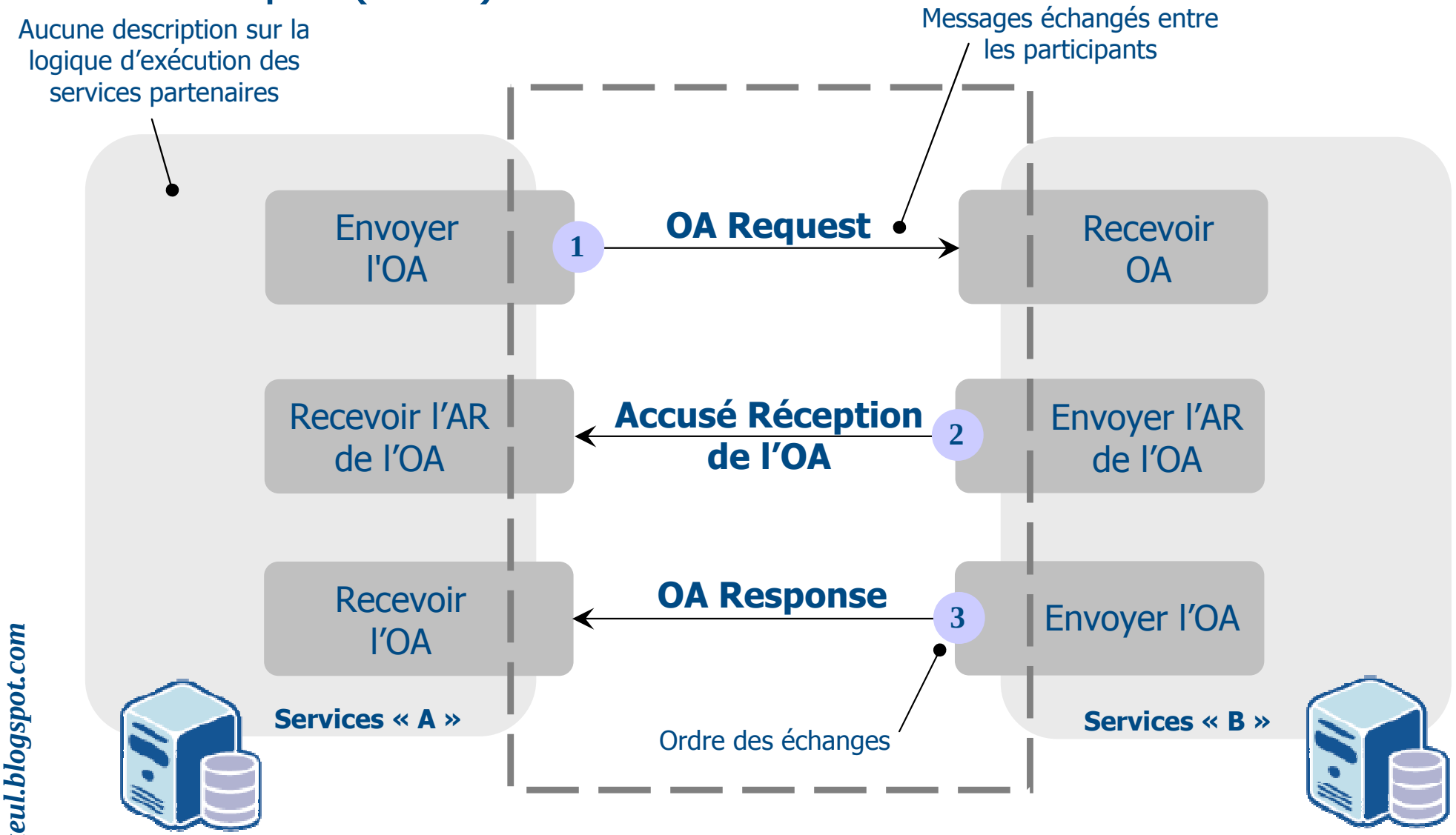
Chorégraphie / Orchestration

➤ Exemple : Ordre d'achat



Chorégraphie / Orchestration

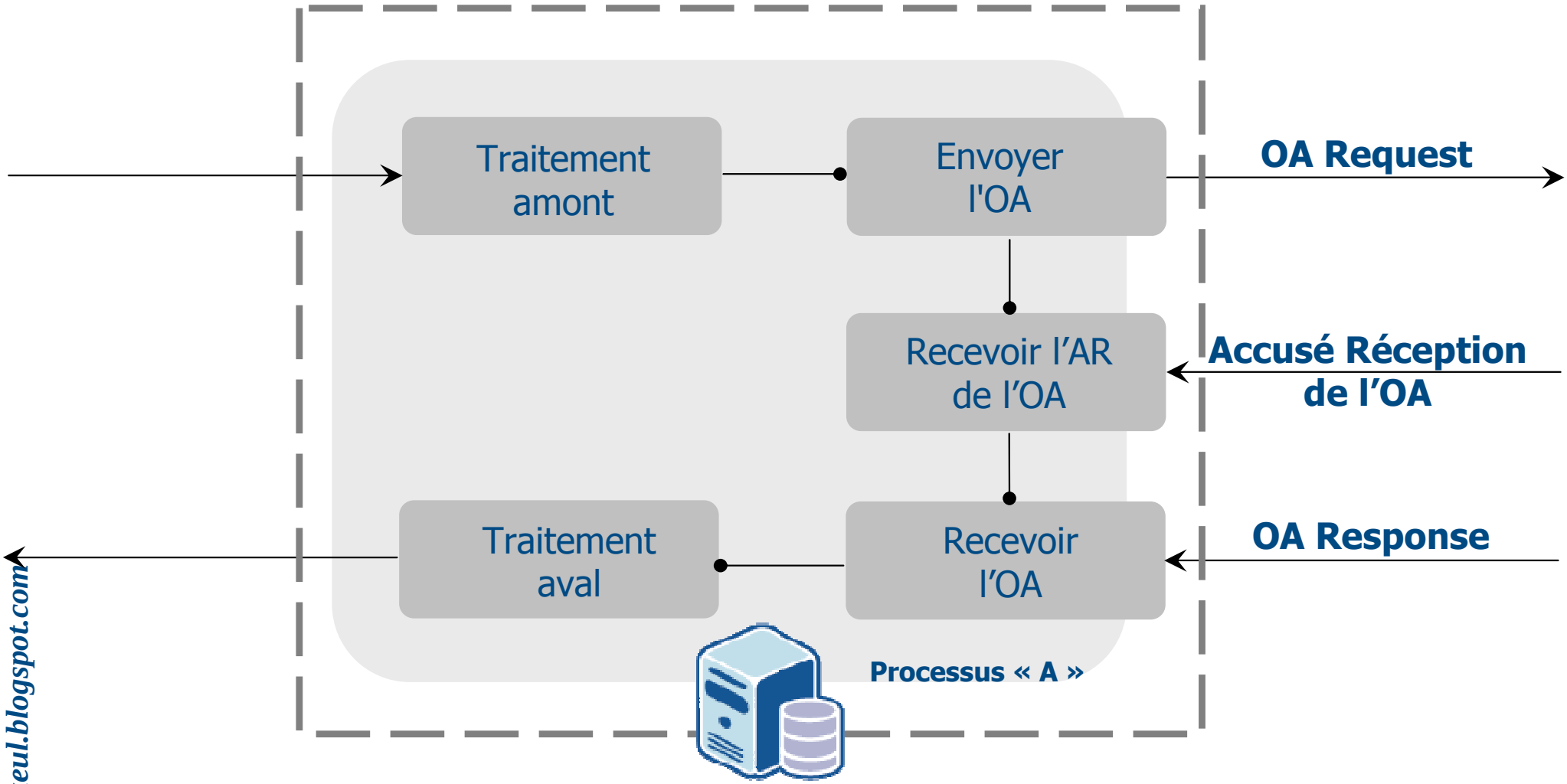
➤ Exemple (suite) : Ordre d'achat



Chorégraphie : Description des messages échangés entre les participants et leurs ordres

Chorégraphie / Orchestration

Exemple (suite) : Ordre d'achat



Orchestration : Description de la logique d'exécution des services partenaires et des messages échangés

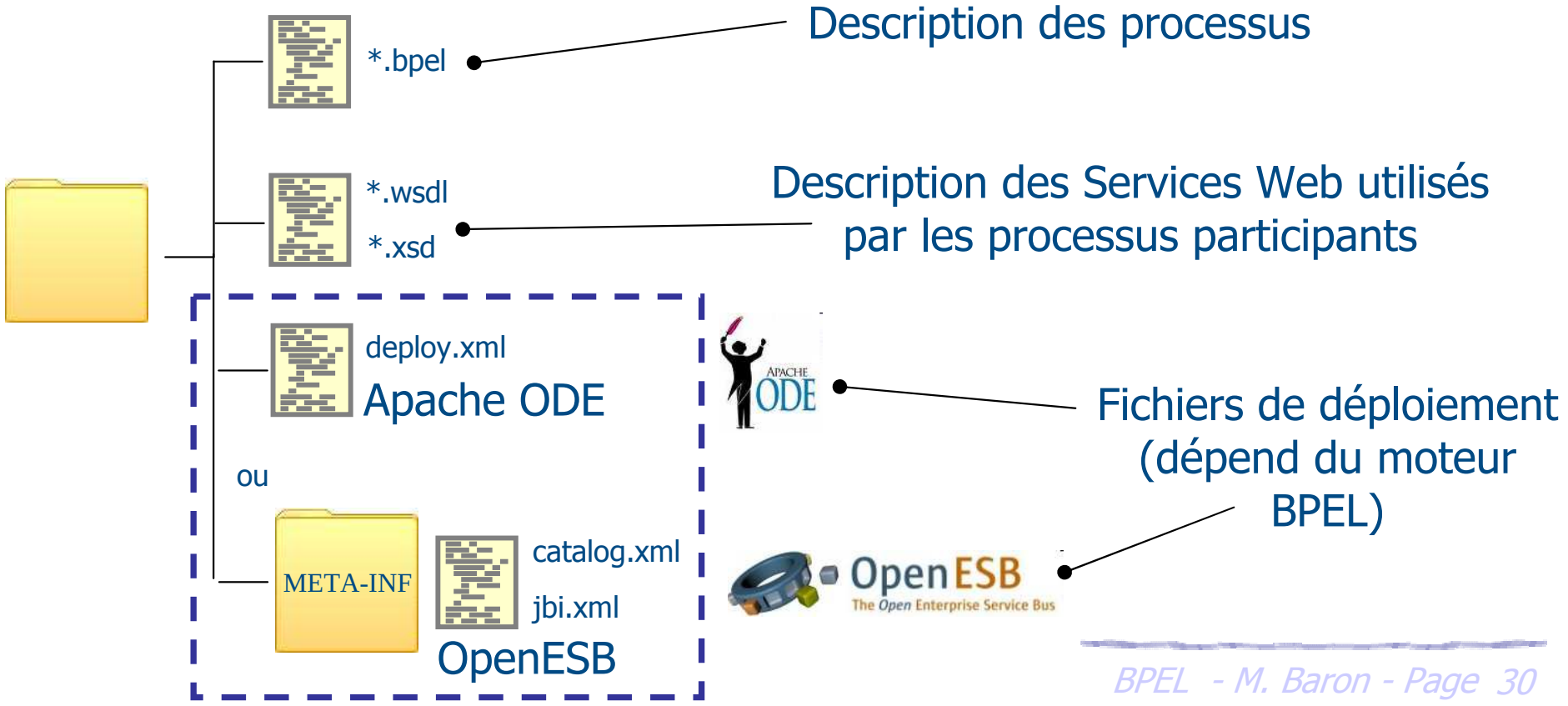
Structure d'un fichier BPEL

- **<process>** (un seul autorisé)
  - **<partnerLinks>** (un seul autorisé)
    - Définition et rôles des processus participants
  - **<variables>** (un seul autorisé)
    - Variable d'état utilisée dans le processus
  - **<correlationSets>** (un seul autorisé)
    - Propriétés liées au déroulement du processus
  - **<faultHandlers>** (un seul autorisé)
    - Traitement des erreurs déclenchées par l e processus
  - **<compensationHandlers>** (un seul autorisé)
    - Traitements des contextes arrières (undo)
  - **<eventHandlers>** (un seul autorisé)
    - Traitement des événements
  - **<Activities>** (plusieurs autorisés)
    - Description du processus

Etudiées dans ce cours d'introduction

Structure d'un projet BPEL

- Un projet BPEL se compose de trois catégories de fichiers
  - Les fichiers de description des processus (**.bpel*)
  - Les fichiers de description des Services Web (**.wsdl* et **.xsd*)
  - Les fichiers de déploiement (**.xml*) dont la structure dépend fortement du moteur BPEL utilisé



BPEL par l'exemple : HelloWorld 2

- Nous étendons le processus *HelloWorld* précédent en ajoutant une invocation à un Service Web externe
- Ce processus BPEL est décrit par le Service Web suivant
  - Une opération *makeHello* qui prend en paramètre une chaîne de caractères et retourne une chaîne de caractères
- Le processus BPEL traite le Workflow suivant
  - Récupération du message envoyé par le client (chaîne de caractères)
  - Invocation d'un Service Web externe (partenaire) avec comme paramètre la chaîne de caractères reçue par le processus
  - Retourner au client le message récupéré de l'invocation précédente

Partner Links

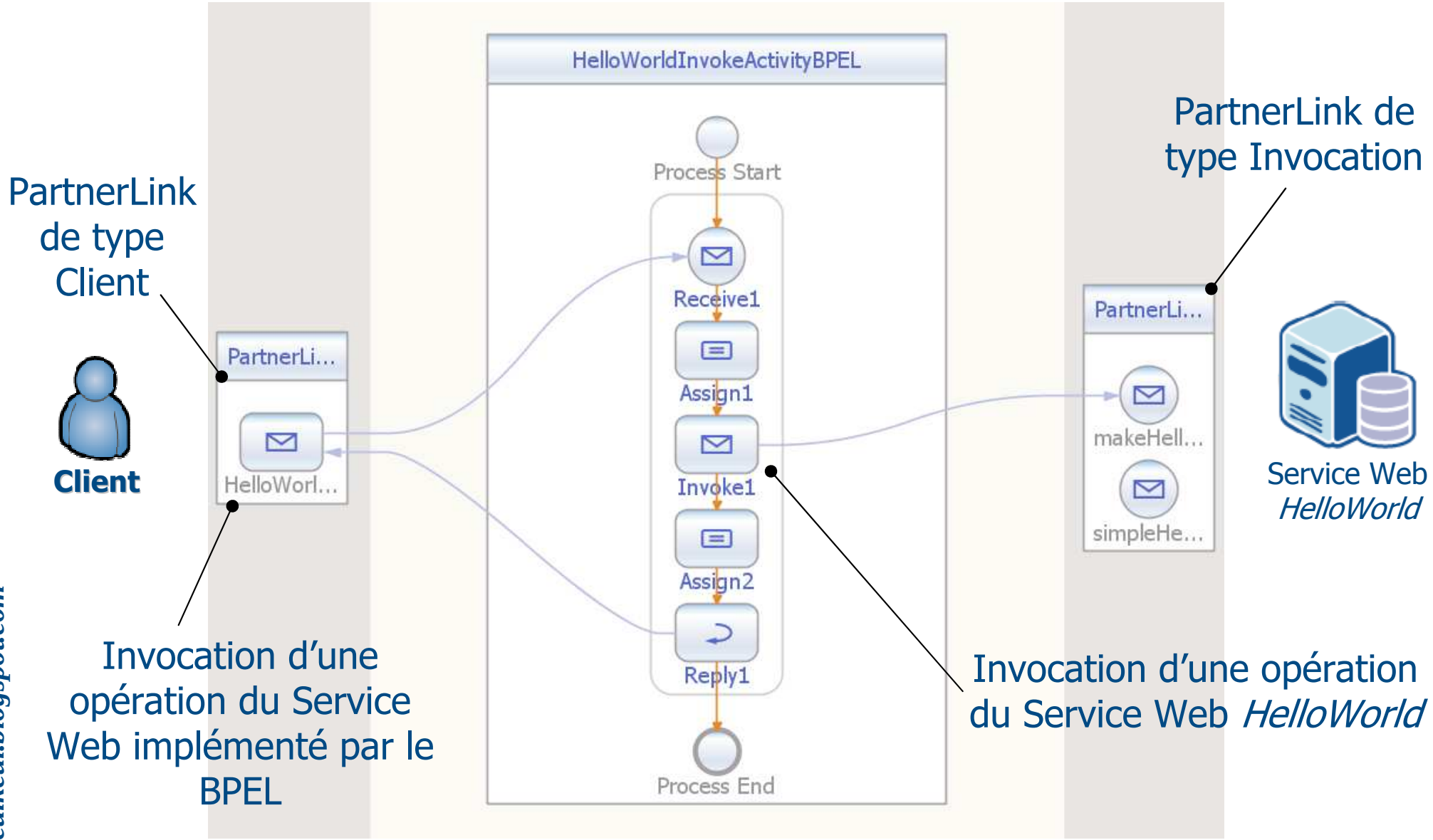
- Pour rappel, du point de vue des clients le processus BPEL est un Service Web
- Deux façons d'interagir entre un processus BPEL et des Web Services externes
  - Un processus BPEL invoque des opérations issues d'autres Web Services (dit Partenaires) : Lien de partenaire (*PartenerLink*) de type invocation
  - Un processus BPEL reçoit des invocations issues de clients : Lien de partenaire (*PartenerLink*) de type client
- Pour résumer, un lien de partenaire désigne les relations entre des partenaires / clients et le processus BPEL

Partner Links

- Un lien de partenaire est représenté dans le fichier BPEL par un élément *partnerLink*
- Il contient les attributs suivants
  - *name* : nom donné au *partnerLink*
  - *myRole* : spécifie le rôle du processus BPEL
  - *partnerRole* : spécifie le rôle du partenaire ou du client
  - *partnerLinkType* : type de *partnerLink* défini dans la description WSDL
- Si l'attribut *myRole* est uniquement utilisé (sans *partnerRole*) seules les interactions vers le processus sont autorisées
- Si l'attribut *partnerRole* est uniquement utilisé (sans *myRole*) seules les interactions vers les partenaires et les clients sont autorisés
- Les deux rôles peuvent être utilisés en même temps

Partner Links

- Exemple : Deux liens de partenaire (*client* et *invocation*)



Partner Links

- Exemple (suite) : Deux liens de partenaire (*client* et *invocation*)

Description du lien de partenaire lié à l'interaction entre le client et le processus

```
<partnerLinks>
   <partnerLink
        name="PartnerLinkClient"
        partnerLinkType="tns:HelloWorldInvokeActivityBPEL"
        myRole="HelloWorldInvokeActivityBPELPortTypeRole"/>


   <partnerLink
        name="PartnerLinkInvocation"
        partnerLinkType="tns:HelloWorldLinkType"
        partnerRole="HelloWorldRole"/>
</partnerLinks>
```

Description du lien de partenaire lié à l'interaction entre le processus et le Service Web *HelloWorld*

HelloWorldInvokeActiviyBPEL.bpel du projet **HelloWorldInvokeActivityBPEL**

Partner Links Types

- Un *PartnerLinkType* décrit la relation entre deux services en détaillant le rôle que chaque service implémente
- Chaque rôle spécifie un *PortType* (issu du WSDL) qui doit être implémenté par le service qui implémente ce rôle

```
<partnerLink
     name="PartnerLinkInvocation"
     partnerLinkType="tns:HelloWorldLinkType"
     partnerRole="HelloWorldRole"/>
...
```

Le lien de partenaire (PartnerLink) décrit dans le fichier BPEL

Le PartnerLinkType est décrit dans le fichier WSDL du Service Web

```
<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>
<plnk:partnerLinkType name="HelloWorldLinkType">
     <plnk:role name="HelloWorldRole" portType="ns:HelloWorld"/>
</plnk:partnerLinkType>
```

Variables

- Les messages sont envoyés et reçus vers et de partenaires
  - Nécessite de sauvegarder ces messages pour être utilisés par le processus BPEL
- BPEL définit la notion de variables qui permet de manipuler les messages des interactions entre les partenaires
- Une variable est définie par des types et des messages déclarés dans un WSDL
- Une variable est définie par les attributs suivant
  - *name* : nom de la variable
  - *type* : typée via un type XML Schema par exemple

    Ou
  - *messageType* : typée via un message

Variables

Exemple : Définition de variables dans un BPEL

```
<variables>
     <variable name="tmpValue" type="xs:string"/>
     <variable name="MakeHelloWorldOutput" messageType="tns:makeHelloWorldResponse"/>
     <variable name="MakeHelloWorldInput" messageType="tns:makeHelloWorld"/>
     <variable name="output" messageType="tns:HelloWorldInvokeActivityBPELOperationResponse"/>
     <variable name="input" messageType="tns:HelloWorldInvokeActivityBPELOperationRequest"/>
</variables>
```

Variables décrites dans le fichier BPEL

```
<message name="HelloWorldInvokeActivityBPELOperationRequest">
     <part name="inputmessage" type="xsd:string"/>
</message>
<message name="HelloWorldInvokeActivityBPELOperationResponse">
     <part name="outputmessage" type="xsd:string"/>
</message>
```

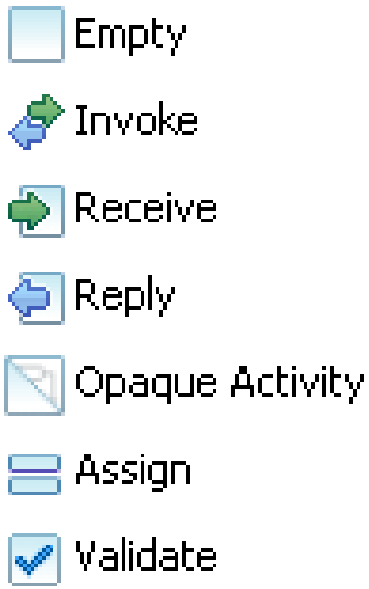
Messages décrits dans le fichier WSDL du processus

```
<message name="makeHelloWorld">
     <part name="value" type="xsd:string" />
</message>
<message name="makeHelloWorldResponse">
     <part name="helloWorldResult" type="xsd:string" />
</message>
```

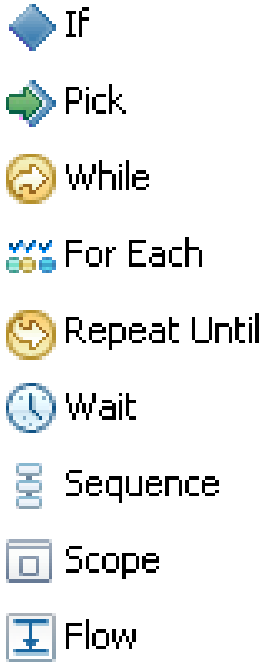
Messages décrits dans le fichier WSDL du Service Web

Activités

- Un processus BPEL est décrit par un ensemble d'étapes
- Chaque étape est appelée une **activité** (*activity*)
- Deux catégories d'activité sont à distinguer



Activité Simple



Activité Structurée

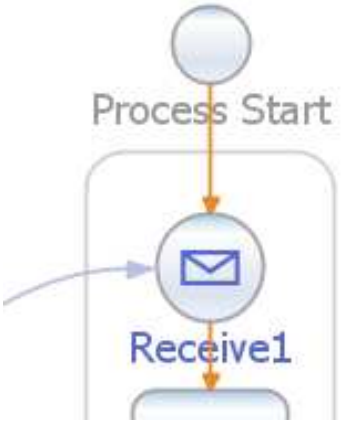
Activités : Receive

- L'activité **Receive** est utilisée pour la mise en attente du processus tant qu'un message n'est pas envoyé par un partenaire
- Elle est utilisée généralement pour instancier le processus BPEL (première activité du processus)
- Elle est définie par la balise *<receive>* dont les principaux attributs de cette activité sont
  - *name* : nom de l'activité
  - *partnerLink* : lien partenaire utilisé pour identifier le partenaire qui doit déclencher le processus
  - *operation* : identifiant de l'opération que le processus doit implémenter
  - *variable* : où stocker le message envoyé par le partenaire

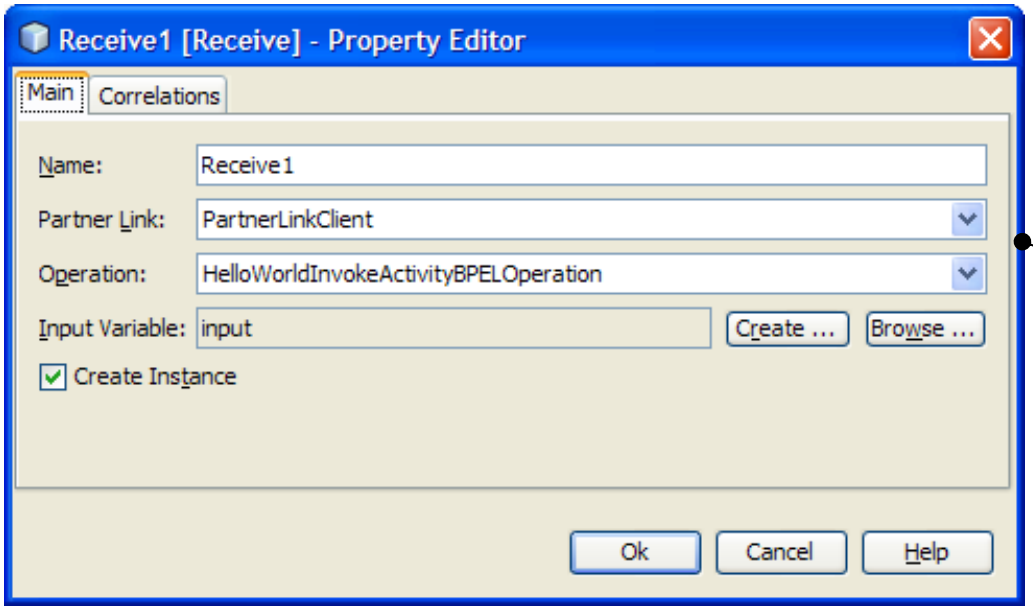
Activités : Receive

Exemple : Mise en place de l'activité **Receive**

```
<receive
      name="Receive1"
      createInstance="yes"
      partnerLink="PartnerLinkClient"
      operation="HelloWorldInvokeActivityBPELOperation"
      portType="tns:HelloWorldInvokeActivityBPELPortType"
      variable="input"/>
```



HelloWorldInvokeActiviyBPEL.bpel du projet **HelloWorldInvokeActivityBPEL**



Des outils simplifient l'édition d'une activité **Receive**

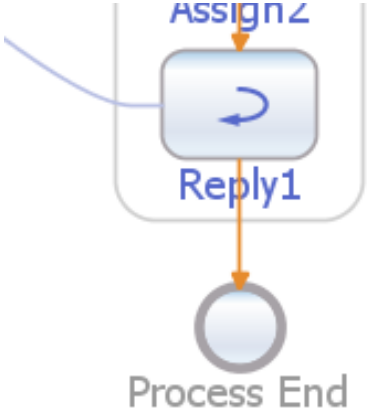
Activités : Reply

- L'activité **Reply** permet d'envoyer une réponse au message envoyé à l'activité **Receive**
- Le couple **Reply** / **Receive** décrit une opération de type requête / réponse
- Elle est définie par la balise *<reply>* dont les principaux attributs de cette activité sont
  - *name* : nom de l'activité
  - *partnerLink* : identifiant du lien partenaire utilisé pour identifier le partenaire qui doit recevoir le message de réponse
  - *operation* : identifiant de l'opération que le processus a implémenté
  - *variable* : message contenant le message à retourner

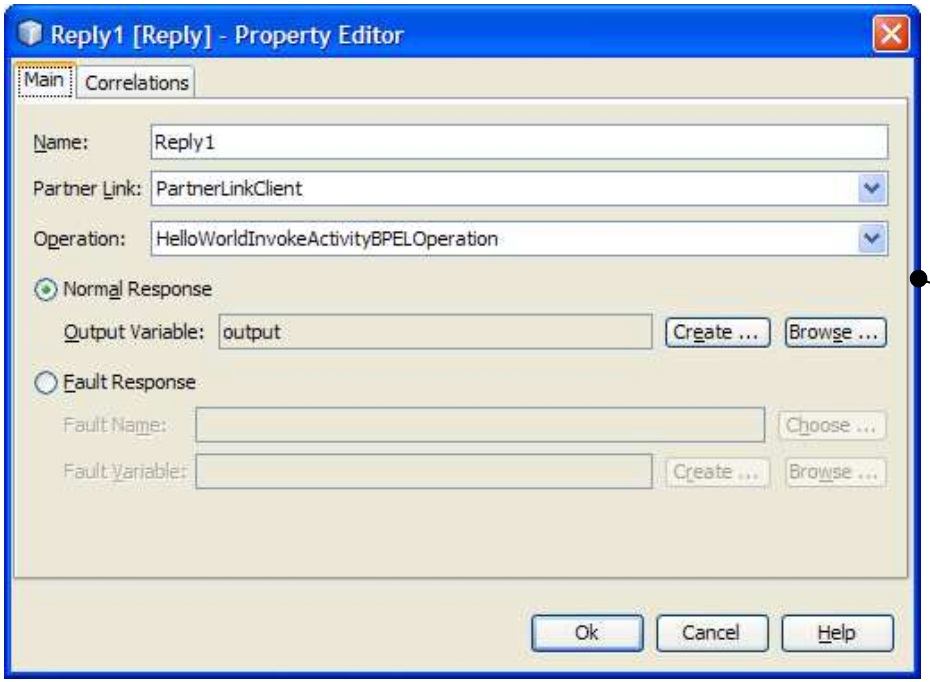
Activités : Reply

Exemple : Mise en place de l'activité **Reply**

```
<reply
      name="Reply1"
      partnerLink="PartnerLinkClient"
      operation="HelloWorldInvokeActivityBPELOperation"
      portType="tns:HelloWorldInvokeActivityBPELPortType"
      variable="output"/>
```



HelloWorldInvokeActiviyBPEL.bpel du projet **HelloWorldInvokeActivityBPEL**



Des outils simplifient l'édition d'une activité **Reply**

Activités : Assign

- L'activité **Assign** peut être utilisée pour copier des données d'une variable vers une autre
- Possibilité d'utiliser des expressions complexes pour modifier le contenu des variables (XPath, transformations XSL)
- Elle est définie par la balise *<assign>* qui peut contenir un ensemble de sous balises <copy>
- La balise *<copy>* contient à son tour des sous balises *<from>* et *<to>* pour exprimer la copie d'un contenu vers un autre
- Structure XML de la balise *<assign>*

```
<assign>
    <copy>
        <from ...>
        <to ...>
    </copy>
</assign>
```

Activités : Assign

➤ Exemple : Mise en place de l'activité **Assign**

```
<assign name="Assign1">
     <copy>
          <from variable="input" part="inputmessage"/>
          <to variable="MakeHelloWorldInput" part="value"/>
     </copy>
</assign>
```

HelloWorldInvokeActiviyBPEL.bpel du projet **HelloWorldInvokeActivityBPEL**

Copie le contenu de la variable *input* vers celui de la variable *MakeHelloWorldInput*

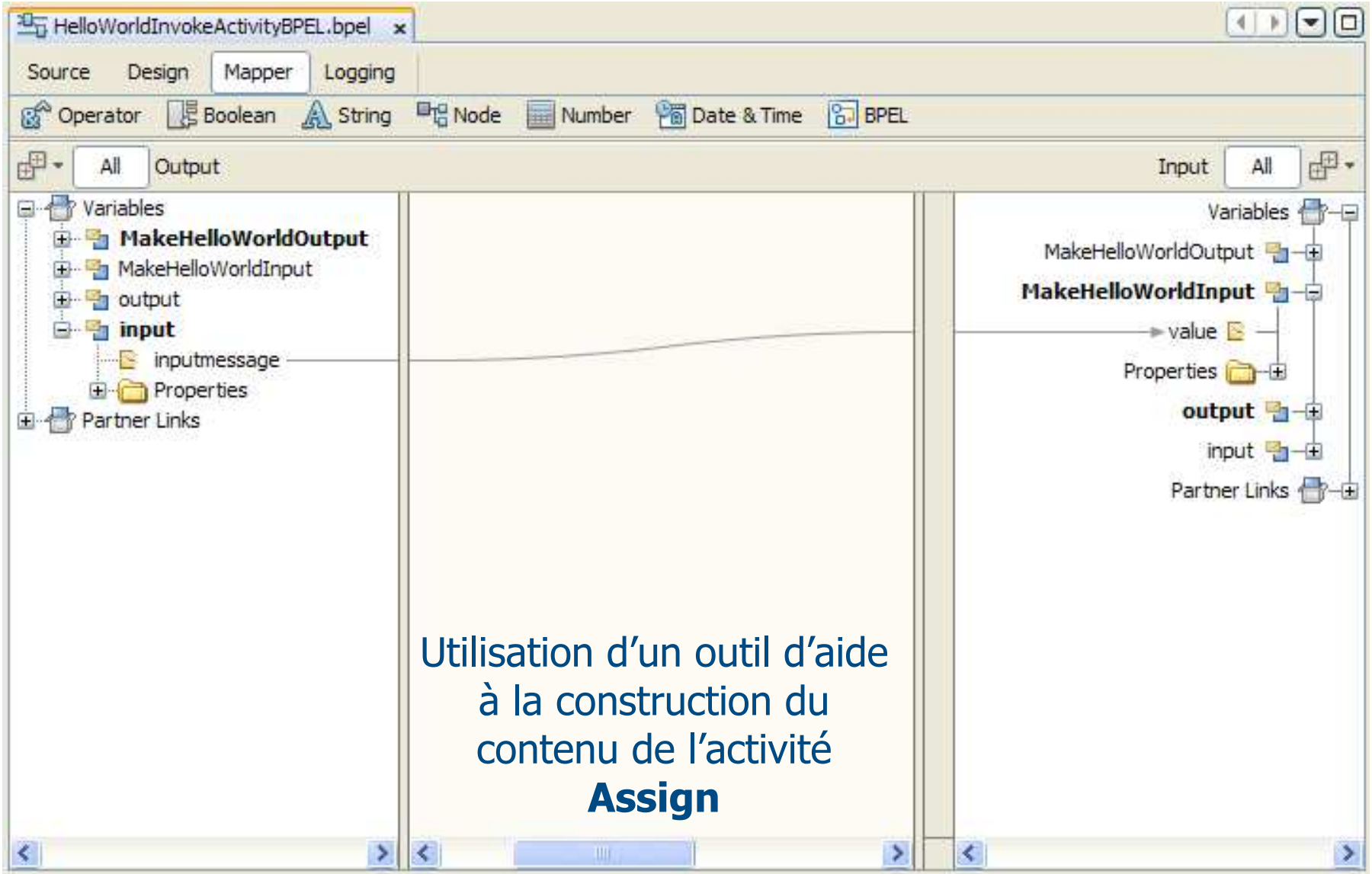
```
<message name="HelloWorldInvokeActivityBPELOperationRequest">
     <part name="inputmessage" type="xsd:string"/>
</message>
```

```
<message name="makeHelloWorld">
     <part name="value" type="xsd:string" />
</message>
```

Respect des typages puisque les parts des messages sont de type identique (*xsd:string*)

Activités : Assign

- Exemple (suite) : Mise en place de l'activité **Assign**



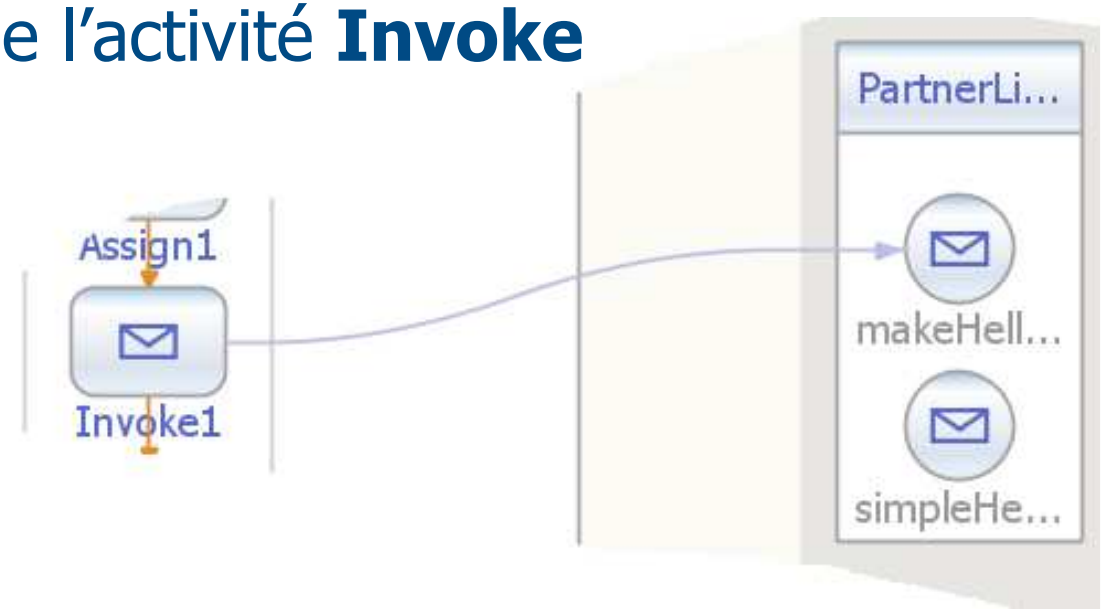
Activities : Invoke

- L'activité **Invoke** est utilisée pour déclencher l'appel à une opération sur un *portType* défini par un lien de partenaire
- Seules les opérations suivant le modèle **One-way** et **Request/Response** peuvent être invoquées par un BPEL
- L'invocation d'opération nécessite l'utilisation de variables en entrée et en sortie pour initialiser la requête et la réponse
- Elle est définie par la balise *<invoke>*
  - *name* : nom de l'activité
  - *partnerLink* : identifiant du lien partenaire
  - *operation* : l'opération à invoquer
  - *portType* : le *portType* pour de l'opération
  - *inputVariable* : informations à transmettre à la requête
  - *ouputVariable* : utilisées pour récupérer les données de la réponse

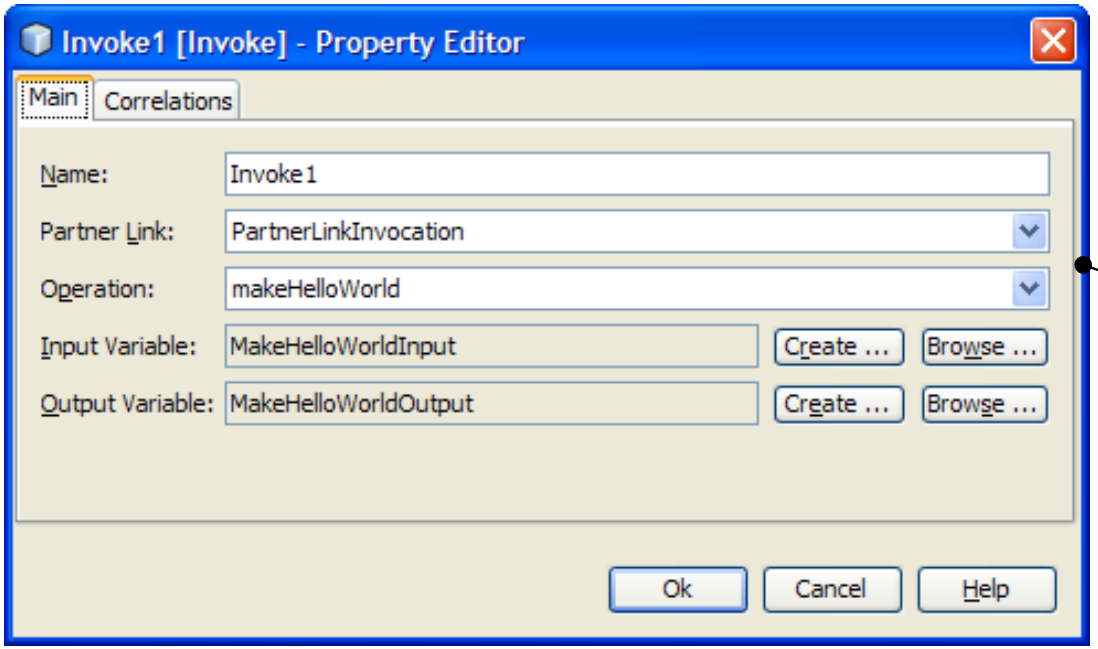
Activities : Invoke

➤ Exemple : Mise en place de l'activité **Invoke**

```
<invoke
      name="Invoke1"
      partnerLink="PartnerLinkInvocation"
      operation="makeHelloWorld"
      portType="tns:HelloWorld"
      inputVariable="MakeHelloWorldInput"
      outputVariable="MakeHelloWorldOutput"/>
```



HelloWorldInvokeActiviyBPEL.bpel du projet **HelloWorldInvokeActivityBPEL**



Des outils simplifient l'édition d'une activité **Invoke**

Activities : Invoke

- L'activité **Invoke** ne permet de déclencher que des opérations issues d'un Service Web étendu (via WSDL)
- Les travaux de standardisation s'intéressent actuellement à la possibilité d'invoquer différents services de technologies différentes (Service Web de type REST)
- Toutefois, il est actuellement possible d'invoquer des services sous condition de définir un *adaptateur* WSDL
- Démarche
  - Définir un Service Web « Adaptateur » respectant le contrat du service (défini par un EJB) à invoquer
  - Implémenter le mapping entre le Service Web « Adaptateur » et le service à invoquer
  - Ajouter comme lien de partenaire le Service Web Adaptateur
  - Invoquer les opérations

Activities : Sequence / Flow / While / Pick / ...

- L'ordonnancement des activités est défini par des activités complexes
- L'activité **Sequence** permet d'exprimer que des activités soient déclenchées dans un ordre donné
- L'activité **Flow** permet de définir qu'une ou plusieurs activités soient déclenchées de manière concurrente
- L'activité **While** permet de définir qu'une activité peut être répétée plusieurs fois
- L'activité **Pick** permet de mettre en attente une activité jusqu'à l'arrivée d'un message
- ...

Activities : Sequence / Flow / While / Pick / …

➤ Exemple : Mise en place de l'activité **Sequence**

```
<sequence>
    <receive name="Receive1" createInstance="yes" partnerLink="PartnerLinkClient" operation="makeHello"
1       portType="tns:HelloWorldInvokeActivityBPELPortType" variable="input"/>

    <assign name="Assign1">
        <copy>
2           <from variable="input" part="inputmessage"/>
            <to variable="MakeHelloWorldInput" part="value"/>
        </copy>
    </assign>

    <invoke name="Invoke1" partnerLink="PartnerLinkInvocation" operation="makeHelloWorld“
3       inputVariable="MakeHelloWorldInput" outputVariable="MakeHelloWorldOutput"/>

    <assign name="Assign2">
        <copy>
4           <from variable="MakeHelloWorldOutput" part="helloWorldResult"/>
            <to variable="output" part="outputmessage"/>
        </copy>
    </assign>

    <reply name="Reply1" partnerLink="PartnerLinkClient" operation="makeHello“
5       portType="tns:HelloWorldInvokeActivityBPELPortType" variable="output"/>
</sequence>
```

HelloWorldInvokeActiviyBPEL.bpel du projet **HelloWorldInvokeActivityBPEL**

Bilan

- De nombreux concepts sur BPEL restent encore à étudier
  - Activités simples (*throw*, *wait*, *empty*)
  - Activités complexes (*scope*, *compensate*, *switch*, *link*)
- Processus BPEL en mode Synchrone et asynchrone
- Gestion des erreurs
- Gestion transactionnelle
- Journalisation et alertes
  - Connaître l'état du processus BPEL en cours d'exécution
- Moteur BPEL via Open ESB
  - Configuration du moteur