Une introduction aux services *Web*

Guy Tremblay

Université du Québec à Montréal (UQAM)
http://www.info2.uqam.ca/~tremblay

LATECE
(LAbo. sur les TEchnologies du Commerce Electronique)
http://www.latece.uqam.ca

2 février 2007 / LIRMM

Aperçu

Définition des services *Web*

Description des services de base : WSDL

Description des processus : WS-BPEL

Services *Web* vs. composants logiciels

Travaux en cours au LATECE sur les WS

Quelques pistes à explorer

Aperçu

Définition des services *Web* (W3C)

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL).

Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

[Web Services Architecture, Feb. 2004]

Définition des services *Web* (Kadima et Monfort)

Les service web sont des applications autodescriptives, modulaires et faiblement couplées qui fournissent un modéle simple de programmation et de déploiement d'applications, basé sur des normes, et s'exécutant au travers de l'infrastructure web.

[. . .]

Services web

= *HTTP + SOAP + WSDL + Composants logiciels*

[Les Web Services, Dunod, 2001]

SOA vs. WS

"Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains."

[OASIS, 2006]

SOA vs. WS

"Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains."

[OASIS, 2006]

"SOA describes an architectural style that is independent of using a particular technology. [SOA] involves advertisement of services in some form of a registry that clients can use to introspect, discover, hook up to, and invoke services of their choosing. "

[Bell, 2007]

SOA vs. WS

"Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains."

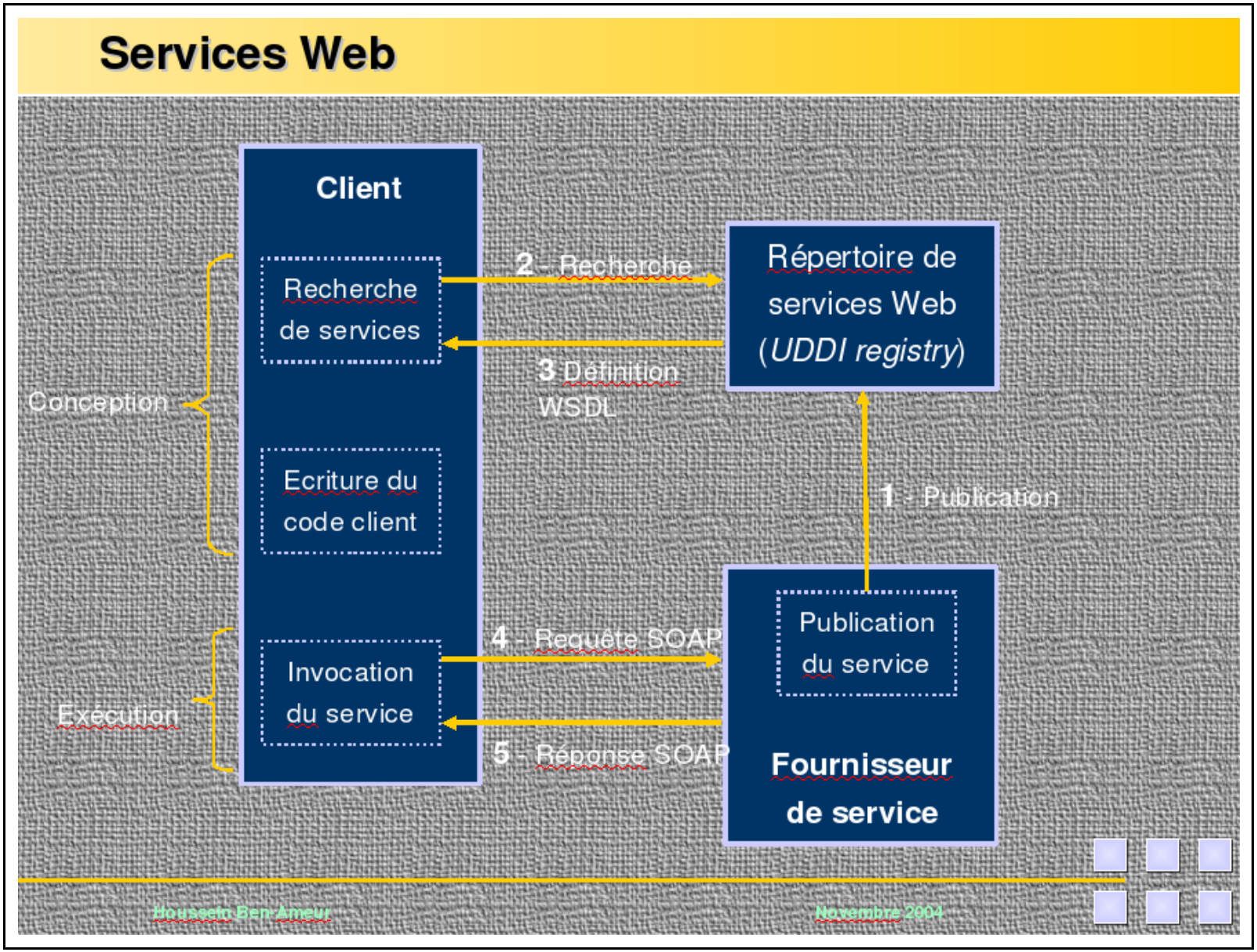
[OASIS, 2006]

"SOA describes an architectural style that is independent of using a particular technology. [SOA] involves advertisement of services in some form of a registry that clients can use to introspect, discover, hook up to, and invoke services of their choosing. "

[Bell, 2007]

WS comme technologie pour SOA

UDDI = *Universal Description, Discovery and Integration*



[H. Ben-Ameur, Séminaire Latece, UQAM, Nov. 2004]

Niveaux de description des WS

- Service de base : Quelles sont les opérations exportées par le service?

  WSDL = *Web Services Description Language*

Niveaux de description des WS

- Service de base : Quelles sont les opérations exportées par le service?

  WSDL = *Web Services Description Language*
- Processus d'affaire = service composite : Quelles sont les interactions du processus avec d'autres WS = Comment le processus réalise-t-il sa tâche?

  WS-BPEL = *Web Services Business Process Execution Language* = Orchestration

Niveaux de description des WS

- Service de base : Quelles sont les opérations exportées par le service?

  WSDL = *Web Services Description Language*
- Processus d'affaire = service composite : Quelles sont les interactions du processus avec d'autres WS = Comment le processus réalise-t-il sa tâche?

  WS-BPEL = *Web Services Business Process Execution Language* = Orchestration
- Groupe de services : Comment doivent-ils collaborer pour réaliser un objectif commun (tâche complexe)?

  WS-CDL = *Web Services Choreography Description Language* = Chorégraphie

Niveaux de description des WS

- Service de base : Quelles sont les opérations exportées par le service?

  WSDL = *Web Services Description Language*
- Processus d'affaire = service composite : Quelles sont les interactions du processus avec d'autres WS = Comment le processus réalise-t-il sa tâche?

  WS-BPEL = *Web Services Business Process Execution Language* = Orchestration
- Groupe de services : Comment doivent-ils collaborer pour réaliser un objectif commun (tâche complexe)?

  WS-CDL = *Web Services Choreography Description Language* = Chorégraphie . . . une autre fois

Aperçu

Description WSDL d'un service

- WSDL = *Web Service Description Language*
- Un service est caractérisé par un ou plusieurs ports ; chaque port est typé et lié à un protocole spécifique
- Un WS est décrit par un fichier WSDL (XML)

Description WSDL d'un service

- WSDL = *Web Service Description Language*
- Un service est caractérisé par un ou plusieurs ports ; chaque port est typé et lié à un protocole spécifique
- Un WS est décrit par un fichier WSDL (XML)
  - Spécification abstraite (interface)
    - Définitions de types de données (XSD)
    - Définitions de messages (opérations)
    - Définitions de types de port (collection d'opérations)
  - Spécification concrète
    - Définitions de liaisons (protocoles et format de données)
    - Définitions de ports (adresses)
    - Définitions de services

Spécification abstraite : Message

- Un message comporte une ou plusieurs parties (champs)
- Chaque partie est caractérisée par un type
- Exemple :

```
<message name="GetTradePriceInput">
  <part name="tickerSymbol" element="xsd:string"/>
  <part name="time"element="xsd:timeInstant"/>
</message>

<message name="GetTradePriceOutput">
  <part name="result" type="xsd:float"/>
</message>
```

Spécification abstraite : Message

- Un message comporte une ou plusieurs parties (champs)
- Chaque partie est caractérisée par un type
- Exemple :

```
message      GetTradePriceInput
  part       tickerSymbol              :string
  part       time                :timeInstant


message      GetTradePriceOutput
  part       result              :float
```

Spécification abstraite : Type de port

- Un type de port permet de décrire les opérations offertes par un service
- Quatre types d'opérations :
  - Unidirectionnelle : le service reçoit un message et ne retourne aucune réponse
  - Requête/réponse : le services reçoit un message et retourne un message en réponse, de façon synchrone
  - Sollicitation/réponse : le service sollicite un autre service et reçoit une réponse en retour
  - Notification : le service sollicite un autre service, sans attendre de réponse

Exemple : Une opération pour obtenir le prix d'un item

```
<portType name="StockQuotePortType">
    <operation name="GetTradePrice">
       <input  message="GetTradePriceInput"/>
       <output message="GetTradePriceOutput"/>
       <fault  message="GetTradePriceWrongInput"/>
     </operation>

     [... Autres opérations du port ...]
</portType>
```

Exemple : Une opération pour obtenir le prix d'un item

```
portType        StockQuotePortType
    operation       GetTradePrice
       input            GetTradePriceInput
       output           GetTradePriceOutput
       fault            GetTradePriceWrongInput

    [... Autres opérations du port ...]
```

Spécification concrète : Liaison (*binding*)

- Spécifie le protocole utilisé ainsi que le format des messages pour les opérations d'un type de port
- Protocoles standards : `SOAP`, `HTTP GET/POST`, `MIME`

Spécification concrète : Liaison (*binding*)

- Spécifie le protocole utilisé ainsi que le format des messages pour les opérations d'un type de port
- Protocoles standards : `SOAP`, `HTTP GET/POST`, `MIME`
- Exemple :

```
<binding name="StockQuoteSoapBinding"
         type="StockQuotePortType">
  <binding style="document"
           transport="http://schemas.xmlsoap.org/soap/http
    <operation name="GetLastTradePrice">
      <operation soapAction="http://example.com/GetLastTra
        <input>  <body use="literal"/> </input>
        <output> <body use="literal"/> </output>
    </operation>
</binding>
```

Spécification concrète : Liaison (*binding*)

- Spécifie le protocole utilisé ainsi que le format des messages pour les opérations d'un type de port
- Protocoles standards : `SOAP`, `HTTP GET/POST`, `MIME`
- Exemple :

```
binding
                StockQuotePortType
 <binding style="document"
          transport="..."
    operation       GetLastTradePrice
      operation soapAction="..."
        input          use="literal"
        output         use="literal"
```

Spécification concrète : Port et service

- Un port spécifie l'adresse d'une liaison
- Un service est un ensemble de ports
  - Plusieurs ports peuvent être associés à un même type de port, avec des liaisons ou adresses différentes

Spécification concrète : Port et service

- Un port spécifie l'adresse d'une liaison
- Un service est un ensemble de ports
  - Plusieurs ports peuvent être associés à un même type de port, avec des liaisons ou adresses différentes
- Exemple :

```
<service name="StockQuoteService">
   <documentation>My first service</documentation>

   <port name="StockQuotePort"
         binding="StockQuoteBinding">
      <address
         location="http://example.com/stockquote"/>
   </port>
</service>
```

Spécification concrète : Port et service

- Un port spécifie l'adresse d'une liaison
- Un service est un ensemble de ports
  - Plusieurs ports peuvent être associés à un même type de port, avec des liaisons ou adresses différentes
- Exemple :

```
service       StockQuoteService


   port       StockQuotePort
        binding="StockQuoteBinding"
      address
        location="http://example.com/stockquote"
```

Aperçu

WS-BPEL

- WS-BPEL = *Web Services – Business Process Execution Language*
- Rôle = Modéliser des processus d'affaires complexes, e.g., avec séquences complexes d'interactions

WS-BPEL

- WS-BPEL = *Web Services – Business Process Execution Language*
- Rôle = Modéliser des processus d'affaires complexes, e.g., avec séquences complexes d'interactions
- Ancêtres (BEA, IBM, Microsoft) :
  - BPEL4WS (*Business Process Execution Language for Web Services*)
  - XLANG (*XML Business Process Language*)
  - WSFL (*Web Services Flow Language*)

WS-BPEL

- WS-BPEL = *Web Services – Business Process Execution Language*
- Rôle = Modéliser des processus d'affaires complexes, e.g., avec séquences complexes d'interactions
- Ancêtres (BEA, IBM, Microsoft) :
  - BPEL4WS (*Business Process Execution Language for Web Services*)
  - XLANG (*XML Business Process Language*)
  - WSFL (*Web Services Flow Language*)
- Deux utilisations prévues :
  - Processus exécutables $\Rightarrow$ comportement effectif d'un processus d'affaire
  - Processus abstraits = "protocoles d'affaire" $\Rightarrow$ protocole d'utilisation d'un processus d'affaire

Principaux éléments d'une description BPEL

- Les opérations fournies aux/requises des partenaires (décrites avec WSDL)
  $\Rightarrow$ `partnerLinkTypes`, `partnerLinks`, `partners`
- L'état interne requis pour modéliser l'interaction et le comportement
  $\Rightarrow$ `variables`, `correlationSets`
- Le comportement du processus (décrit de façon opérationnelle)
  $\Rightarrow$ `activity`

- Exemple : `ordersProcess` [`http://www.it.uc3m.es/jaf/verbus`]

Principaux éléments d'une description BPEL

- Les opérations fournies aux/requises des partenaires (décrites avec WSDL)
  $\Rightarrow$ `partnerLinkTypes`, `partnerLinks`, `partners`
- L'état interne requis pour modéliser l'interaction et le comportement
  $\Rightarrow$ `variables`, `correlationSets`
- Le comportement du processus (décrit de façon opérationnelle)
  $\Rightarrow$ `activity`

- Exemple : `ordersProcess` [`http://www.it.uc3m.es/jaf/verbus`]

Principaux éléments d'une description BPEL

- Les opérations fournies aux/requises des partenaires (décrites avec WSDL)
  $\Rightarrow$ `partnerLinkTypes`, `partnerLinks`, `partners`
- L'état interne requis pour modéliser l'interaction et le comportement
  $\Rightarrow$ `variables`, `correlationSets`
- Le comportement du processus (décrit de façon opérationnelle)
  $\Rightarrow$ `activity`

- Exemple : `ordersProcess` [`http://www.it.uc3m.es/jaf/verbus`]

Exemple `ordersProcess`: les `portTypes` WSDL

```
<portType    name="orderService">
  <operation name="order">
    <input   message="OrderMessage"/>
    <output  message="InvoiceMessage"/>
  </operation>
</portType>

<portType    name="warehouse">
  <operation name="order">
    <input   message="OrderMessage"/>
    <output  message="InvoiceMessage"/>
  </operation>
  <operation name="schedule">
    <input   message="OrderMessage"/>
  </operation>
</portType>

<portType    name="warehouseCallback">
  <operation name="receive_not">
    <input   message="InvoiceMessage"/>
  </operation>
</portType>
```

Exemple `ordersProcess`: les `portTypes` WSDL

```
portType          orderService
  operation       order
    input            OrderMessage
    output           InvoiceMessage


portType          warehouse
  operation       order
    input            OrderMessage
    output           InvoiceMessage

  operation       schedule
    input            OrderMessage


portType          warehouseCallback
  operation       receive_not
    input            InvoiceMessage
```

Exemple `ordersProcess`: les `partnerLinkTypes`

```
<plnk:partnerLinkType name="orderLnk">
  <plnk:role name="orderService">
    <plnk:portType name="tns:orderService"/>
  </plnk:role>
</plnk:partnerLinkType>

<plnk:partnerLinkType name="warehouseLnk">
  <plnk:role name="orderService">
    <plnk:portType name="tns:warehouseCallback"/>
  </plnk:role>
  <plnk:role name="warehouseService">
    <plnk:portType name="tns:warehouse"/>
  </plnk:role>
</plnk:partnerLinkType>
```

Exemple `ordersProcess`: les `partnerLinkTypes`

```
partnerLinkType        orderLnk
  role          orderService
    portType              orderService


partnerLinkType        warehouseLnk
  role          orderService
    portType              warehouseCallback

  role          warehouseService
    portType              warehouse
```

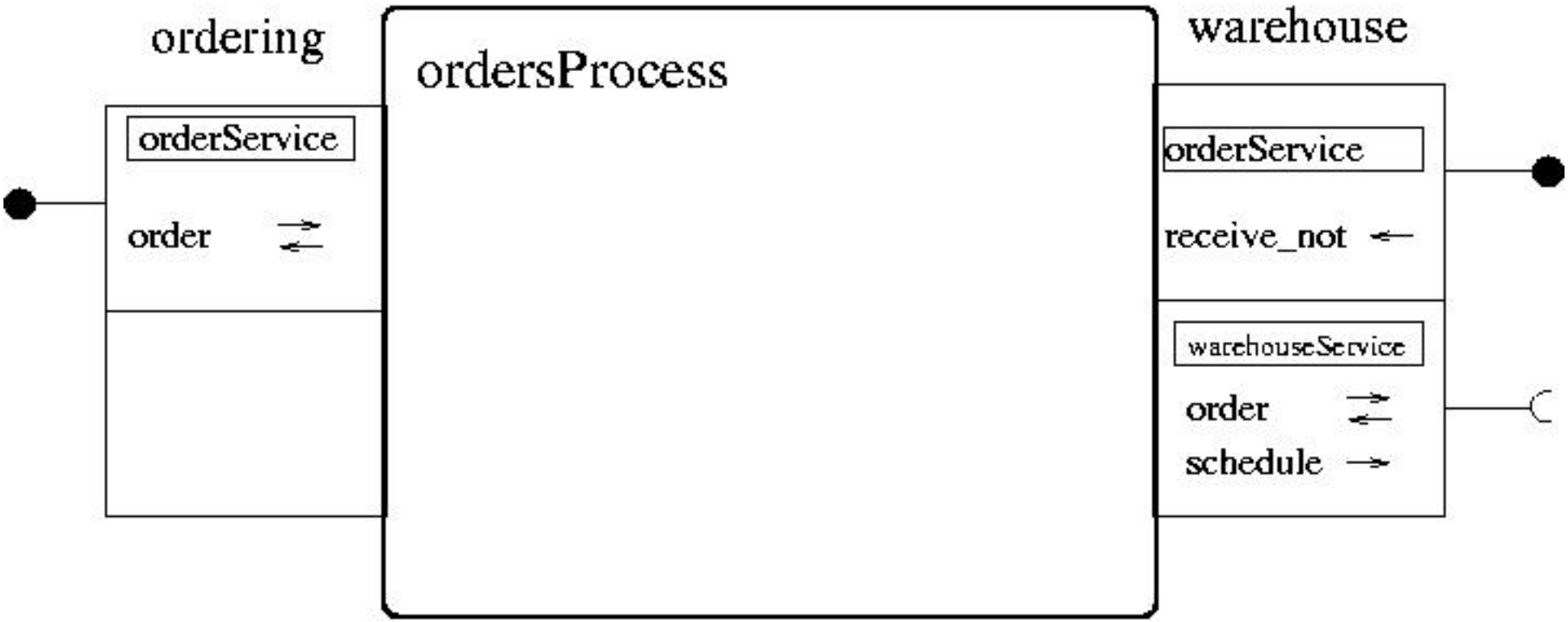
Exemple `ordersProcess` : les `partnerLinks`

```
<partnerLinks>
  <partnerLink name="ordering"
               partnerLinkType="orderLnk"
               myRole="orderService" />

  <partnerLink name="warehouse"
               partnerLinkType="warehouseLnk"
               myRole="orderService"
               partnerRole="warehouseService"/>
</partnerLinks>
```

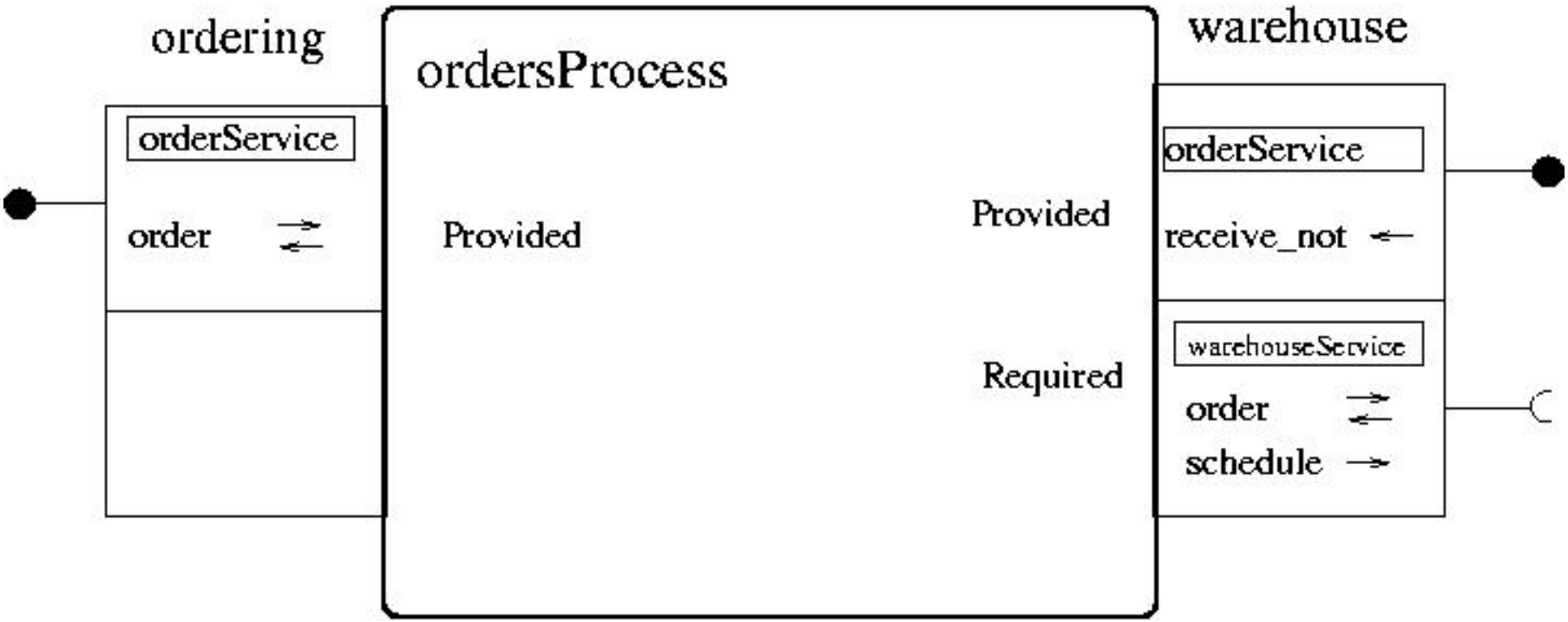
Exemple `ordersProcess` : les `partnerLinks`

```
partnerLink        ordering
                   partnerLinkType="orderLnk"
                   myRole="orderService"

partnerLink        warehouse
                   partnerLinkType="warehouseLnk"
                   myRole="orderService"
                   partnerRole="warehouseService
```

Exemple `ordersProcess`



Exemple `ordersProcess`



Description des activités

Opérations impératives typiques pour processus séquentiels communiquants (à la CSP, CCS, Lotos, etc.)

- `<assign>`
- `<sequence>`, `<flow>`, `<switch>`, `<while>`
- `<receive>`, `<reply>`, `<invoke>`
- `<pick>`, `<wait>`
- `<throw>`, `<compensate>`, `<terminate>`
- `<scope>`

Description des activités

Opérations impératives typiques pour processus séquentiels communiquants (à la CSP, CCS, Lotos, etc.)

- `<assign>`
- `<sequence>`, `<flow>`, `<switch>`, `<while>`
- `<receive>`, `<reply>`, `<invoke>`
- `<pick>`, `<wait>`
- `<throw>`, `<compensate>`, `<terminate>`
- `<scope>`

Particularité : Possible, avec les `flow`, de ne spécifier qu'un ordre partiel d'exécution

Exemple : activités séquentielles

```
<sequence>
  <receive portType="orderService" partnerLink="ordering"
           operation="order" variable="order"
           createInstance="yes"/>

  <switch>
    <case condition="...">
      <invoke portType="warehouse"
              partnerLink="warehouseLnk"
              operation="order"
              inputVariable="order"
              outputVariable="invoice"/>
    </case>

    <otherwise> ... </otherwise>
      </switch>
  ...
</sequence>
```

Exemple : activités séquentielles

```
<sequence>
  <receive portType="orderService" partnerLink="ordering"
           operation="order" variable="order"
           createInstance="yes"/>

  <switch>
    <case condition="...">
      <invoke portType="warehouse"
              partnerLink="warehouseLnk"
              operation="order"
              inputVariable="order"
              outputVariable="invoice"/>
    </case>

    <otherwise> ... </otherwise>
      </switch>
  ...
</sequence>
```

Exemple : activités séquentielles

```
<sequence>
  <receive portType="orderService" partnerLink="ordering"
           operation="order" variable="order"
           createInstance="yes"/>

  <switch>
    <case condition="...">
      <invoke portType="warehouse"
              partnerLink="warehouseLnk"
              operation="order"
              inputVariable="order"
              outputVariable="invoice"/>
    </case>

    <otherwise> ... </otherwise>
      </switch>
  ...
</sequence>
```

Exemple : activités séquentielles

```
<sequence>
  <receive portType="orderService" partnerLink="ordering"
           operation="order" variable="order"
           createInstance="yes"/>

  <switch>
    <case condition="...">
      <invoke portType="warehouse"
              partnerLink="warehouseLnk"
              operation="order"
              inputVariable="order"
              outputVariable="invoice"/>
    </case>

    <otherwise> ... </otherwise>
      </switch>
  ...
</sequence>
```

Exemple : activités séquentielles

```
<sequence>
  <receive portType="orderService" partnerLink="ordering"
           operation="order" variable="order"
           createInstance="yes"/>

  <switch>
    <case condition="...">
      <invoke portType="warehouse"
              partnerLink="warehouseLnk"
              operation="order"
              inputVariable="order"
              outputVariable="invoice"/>
    </case>

    <otherwise> ... </otherwise>
      </switch>
  ...
</sequence>
```

Exemple : sélection non-déterministe

```
<pick>
  <onMessage portType="pt1"
             partnerLink="pl1"
             operation="op1" variable="v1">
      ... activite1 ...
  </onMessage>

  <onMessage portType="p2"
             partnerLink="pl2"
             operation="op1" variable="v2">
      ... activite2 ...
  </onMessage>

  ...

  <onAlarm name="timeout" for="P5D">
    ... activite ...
  </onAlarm>
</pick>
```

Aperçu

Services *Web* vs. composants logiciels (à la SCL)

WS (WSDL)	SCL
▸ `message`	▸ Signature

Services *Web* vs. composants logiciels (à la SCL)

WS (WSDL)

- `operation`
  - `message` (1 ou 2)

SCL

- Service
  - Signature

Services *Web* vs. composants logiciels (à la SCL)

WS (WSDL)

- `portType`
  - `operation` (1 ou +)
    - `message` (1 ou 2)

SCL

- Port
  - Service (1 ou +)
    - Signature

Services *Web* vs. composants logiciels (suite)

WS (BPEL)

- `portType` (fourni)
- `portType` (requis)

SCL

- Port (fourni)
- Port (requis)

Services *Web* vs. composants logiciels (suite)

WS (BPEL)

- `partnerLink`
  - `portType` (fourni)
    (0, 1 ou +)
  - `portType` (requis)
    (0, 1 ou +)

SCL

- Port (fourni)
- Port (requis)

Services *Web* vs. composants logiciels (suite)

WS (BPEL)

- `process`
  - `partnerLink` (1 ou +)
    - `portType` (fourni) (0, 1 ou +)
    - `portType` (requis) (0, 1 ou +)

SCL

- Composant
  - Ports fournis
    - Port (fourni) (0, 1 ou +)
  - Ports requis
    - Port (requis) (0, 1 ou +)

Aperçu

Travaux en cours au LATECE sur les WS

- Vérification de processus exécutable WS-BPEL :
  - Entrées = processus BPEL + *expression d'interface*
  - Sortie = Le comportement du processus satisfait-il l'expression d'interface?
  - Comment :
    - Le processus BPEL est traduit en Promela
    - L'expression d'interface est traduite en assertions de `trace`
    - SPIN est utilisé pour effectuer la vérification

Travaux en cours au LATECE sur les WS

- Vérification de processus exécutable WS-BPEL :
  - Entrées = processus BPEL + *expression d'interface*
  - Sortie = Le comportement du processus satisfait-il l'expression d'interface?
  - Comment :
    - Le processus BPEL est traduit en Promela
    - L'expression d'interface est traduite en assertions de `trace`
    - SPIN est utilisé pour effectuer la vérification
- Conception d'*adapateurs* (processus et messages) pour les processus d'affaires en tourisme (*Open Travel Alliance*)

Travaux en cours au LATECE sur les WS

- Vérification de processus exécutable WS-BPEL :
  - Entrées = processus BPEL + *expression d'interface*
  - Sortie = Le comportement du processus satisfait-il l'expression d'interface?
  - Comment :
    - Le processus BPEL est traduit en Promela
    - L'expression d'interface est traduite en assertions de `trace`
    - SPIN est utilisé pour effectuer la vérification
- Conception d'*adapateurs* (processus et messages) pour les processus d'affaires en tourisme (*Open Travel Alliance*)
- Recherche de services basée sur la génération automatique de composition de processus

Travaux en cours au LATECE sur les WS

- Vérification de processus exécutable WS-BPEL :
  - Entrées = processus BPEL + *expression d'interface*
  - Sortie = Le comportement du processus satisfait-il l'expression d'interface?
  - Comment :
    - Le processus BPEL est traduit en Promela
    - L'expression d'interface est traduite en assertions de `trace`
    - SPIN est utilisé pour effectuer la vérification
- Conception d'*adapateurs* (processus et messages) pour les processus d'affaires en tourisme (*Open Travel Alliance*)
- Recherche de services basée sur la génération automatique de composition de processus
- Heuristique pour la génération des *compensations* dans des processus BPEL

Aperçu

Pistes possibles à explorer

- De quelle façon les stratégies et techniques proposées pour les composants peuvent-elles être utilisées pour les services Web?
- Comment les notions d'orchestration vs. chorégraphie des services Web s'appliquent-elles aux composants logiciels?
- Quels sont les liens entre composants logiciels et architecture/design (Eden et Kazman)?

Pistes possibles à explorer

- De quelle façon les stratégies et techniques proposées pour les composants peuvent-elles être utilisées pour les services Web?
- Comment les notions d'orchestration vs. chorégraphie des services Web s'appliquent-elles aux composants logiciels?
- Quels sont les liens entre composants logiciels et architecture/design (Eden et Kazman)?

	Local	**Non-local**
Extensional	Implementation	
Intensional	Design	Architecture

Questions? Commentaires? Remarques?