Introduction aux Web Services

Orientation Réseaux et Services, Unité PDA

Olivier Liechti

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

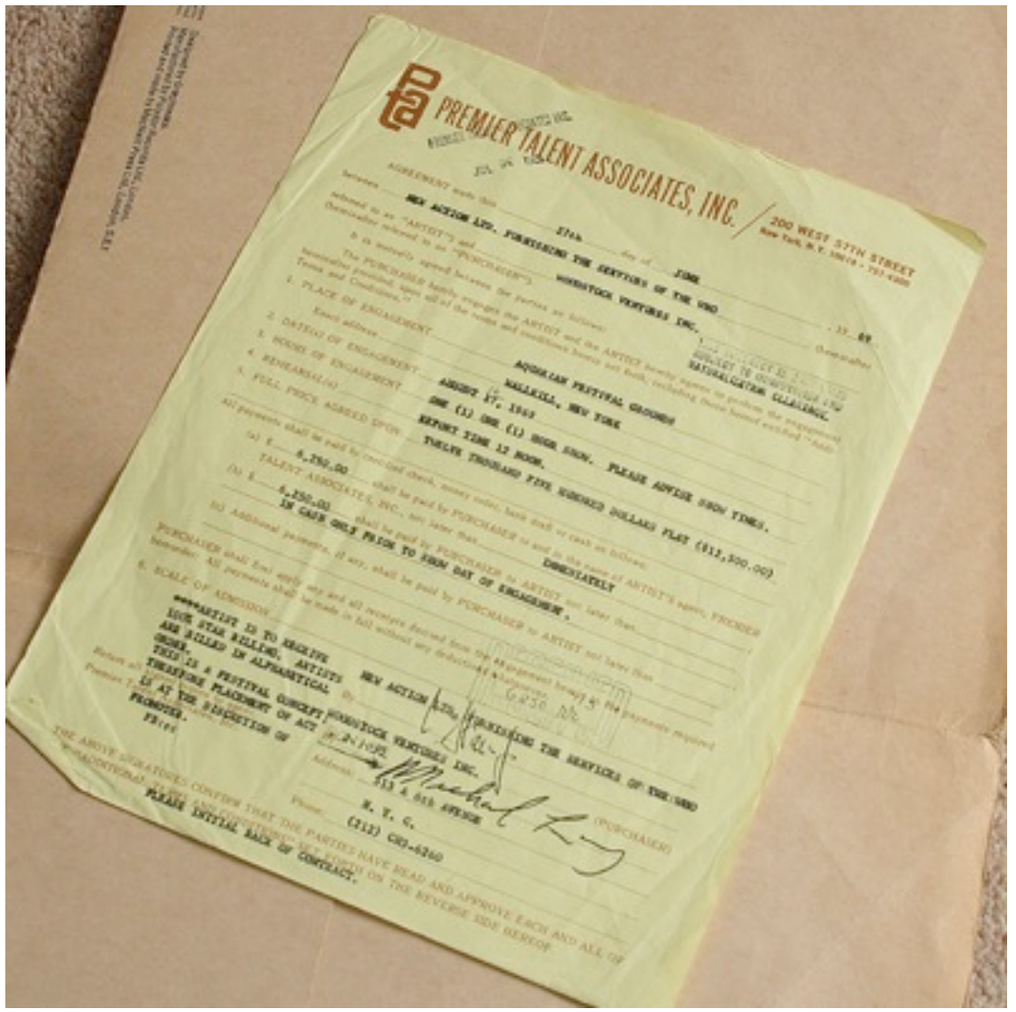
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

La notion de service

La notion de service

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- La notion de service est très générale:
  - On l'utilise pour modéliser la structure de systèmes.
  - C'est vrai pour des "petits" systèmes (une application).
  - C'est vrai pour des "grands" systèmes (tout un système d'information).
  - C'est vrai pour des systèmes "distribués".
- Qu'est-ce qu'un service?
  - Un service est "quelque chose" qui donne accès à une certaine fonctionnalité.
  - La fonctionnalité du service est décrite dans une interface, qui définit le contrat entre le client et le fournisseur du service.

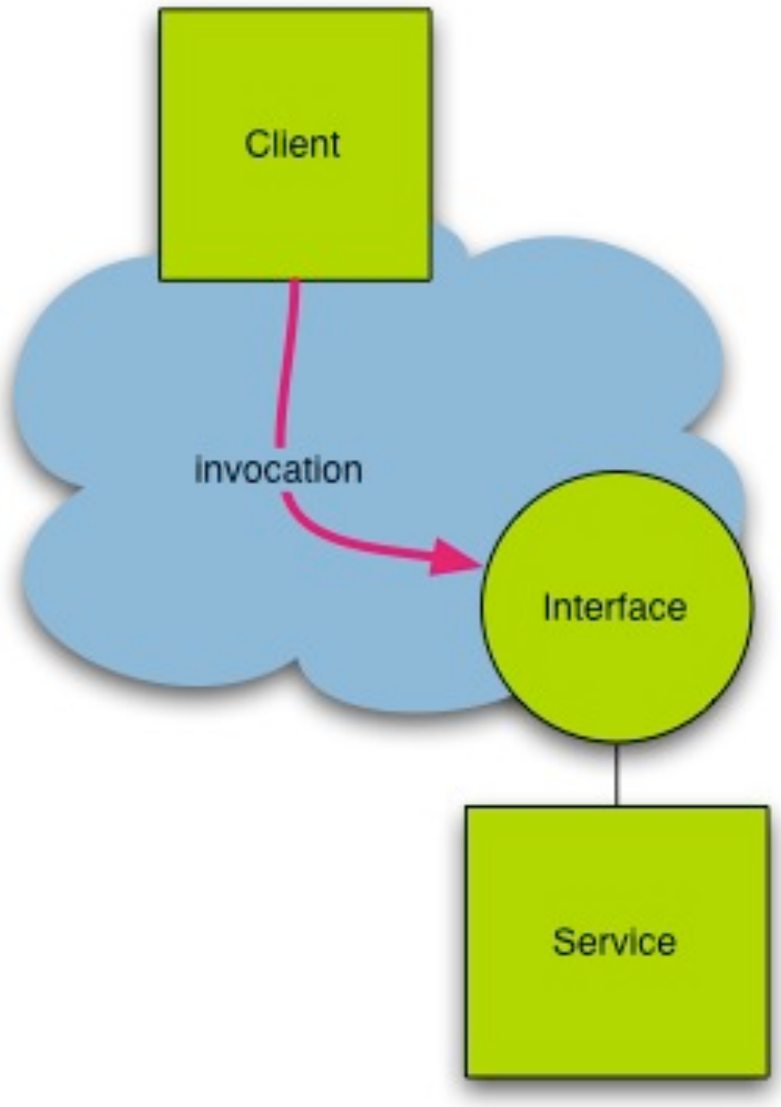


http://flickr.com/photos/ian-s/2655970731/sizes/l/#cc_license

Comment implémenter un service?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- On peut utiliser différents types de technologies pour implémenter un "service".
- Dans une application Java:
  - Une classe qui implémente une interface est un exemple de service.
  - Dans ce cas, les clients utilisent le service au travers d'appels de méthodes (dans la même machine virtuelle).
- Dans une application Java distribuée:
  - La technologie Remote Method Invocation (RMI) permet d'appeler une méthode sur un objet géré par une autre machine virtuelle (au travers du réseau).
  - Ce dernier exemple met en évidence la notion de RPC (Remote Procedure Call)



La notion de "registre de services"

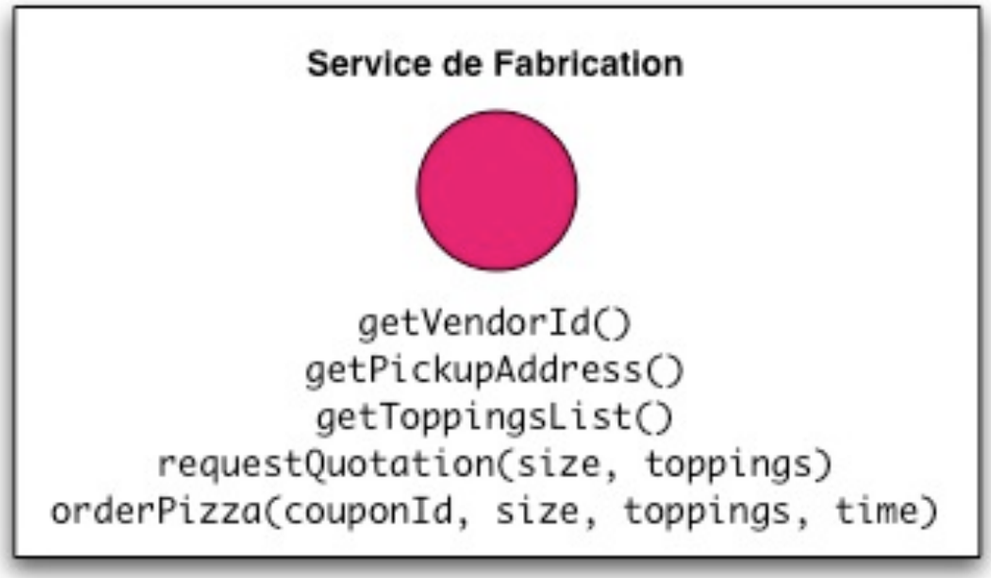
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

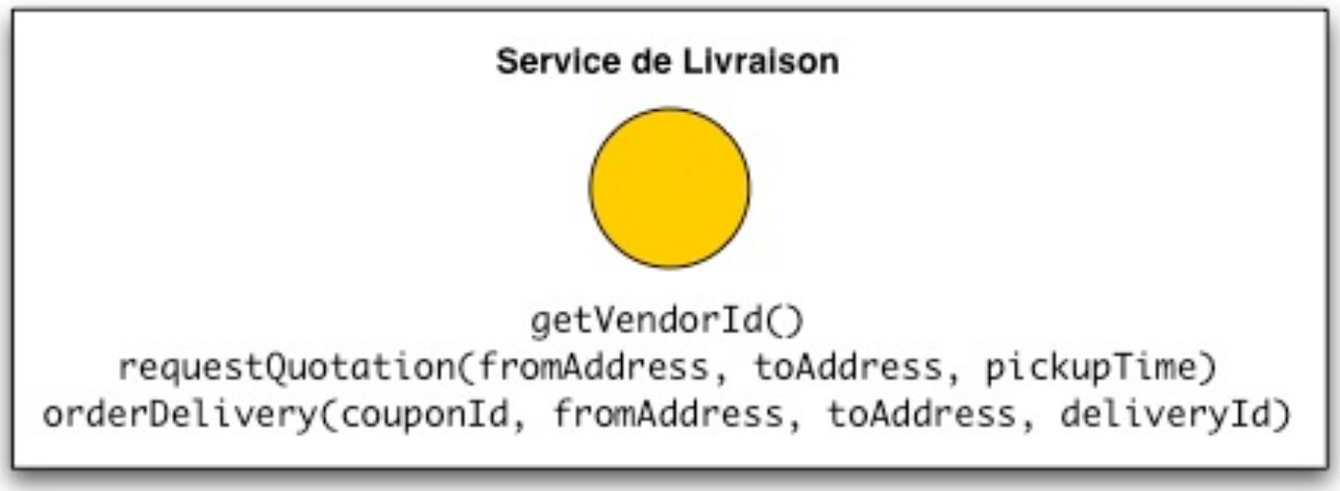
- La vision des "services" va au-delà d'un simple mécanisme d'invocation:
  - De plus en plus de fournisseurs vont créer et déployer des services.
  - De plus en plus d'applications vont vouloir combiner et assembler ces services pour implémenter des processus plus ou moins compliqués.
- Exemple:
  - Pour manger une pizza, j'ai besoin 1) d'un service de fabrication de pizza, 2) d'un service de livraison, 3) d'un service de paiement.
  - Plusieurs entreprises fournissent ces services (pas forcément en permanence).
  - Questions: comment identifier les fournisseurs? Comment choisir un fournisseur?

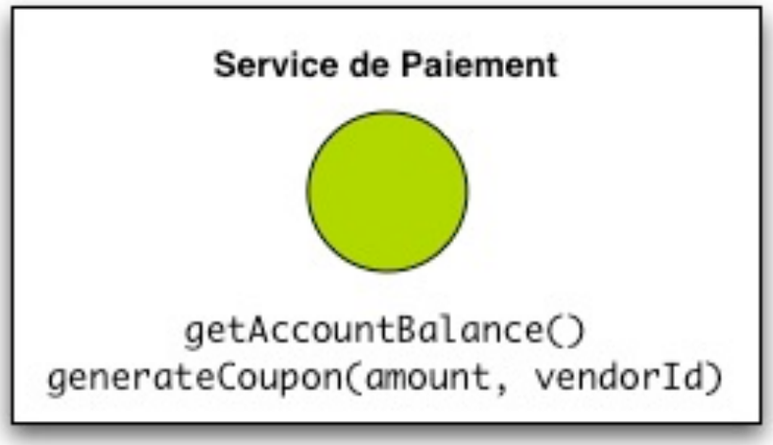
http://flickr.com/photos/qchristopher/1822166721/sizes/m/#cc_license

Exemple: interfaces des différents services

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

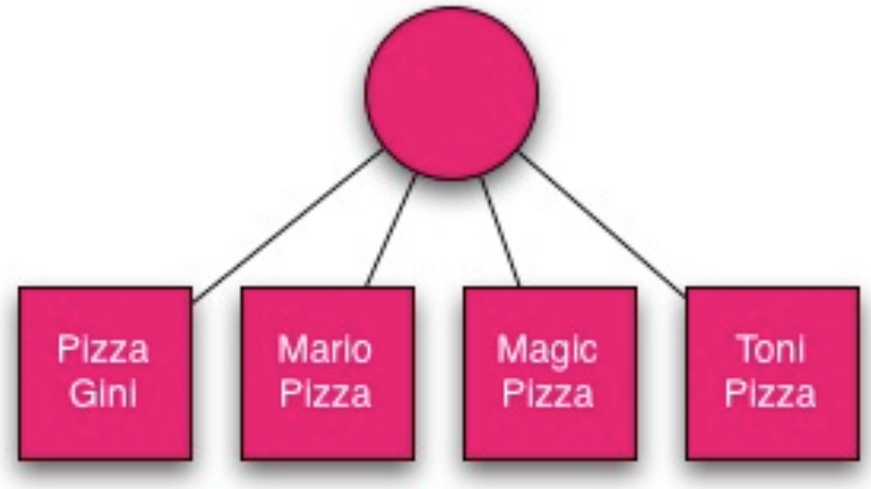


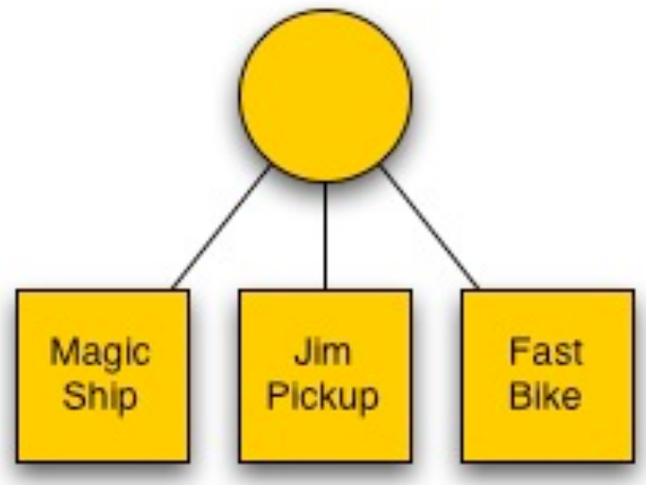


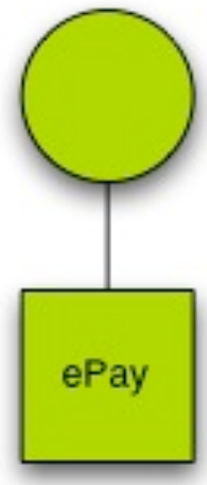


Exemple: plusieurs fournisseurs de service

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

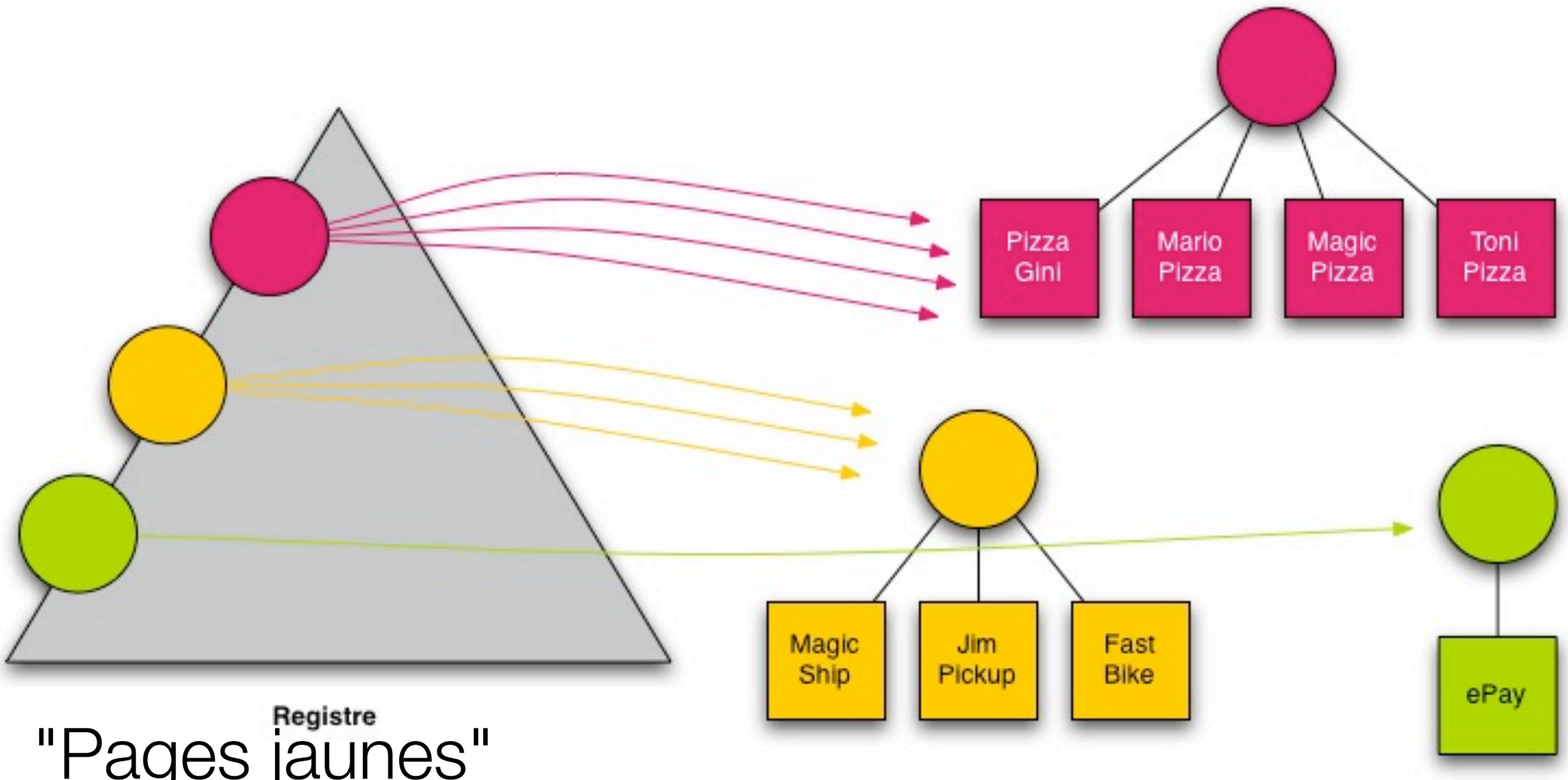






Exemple: enregistrement dans un "registre"

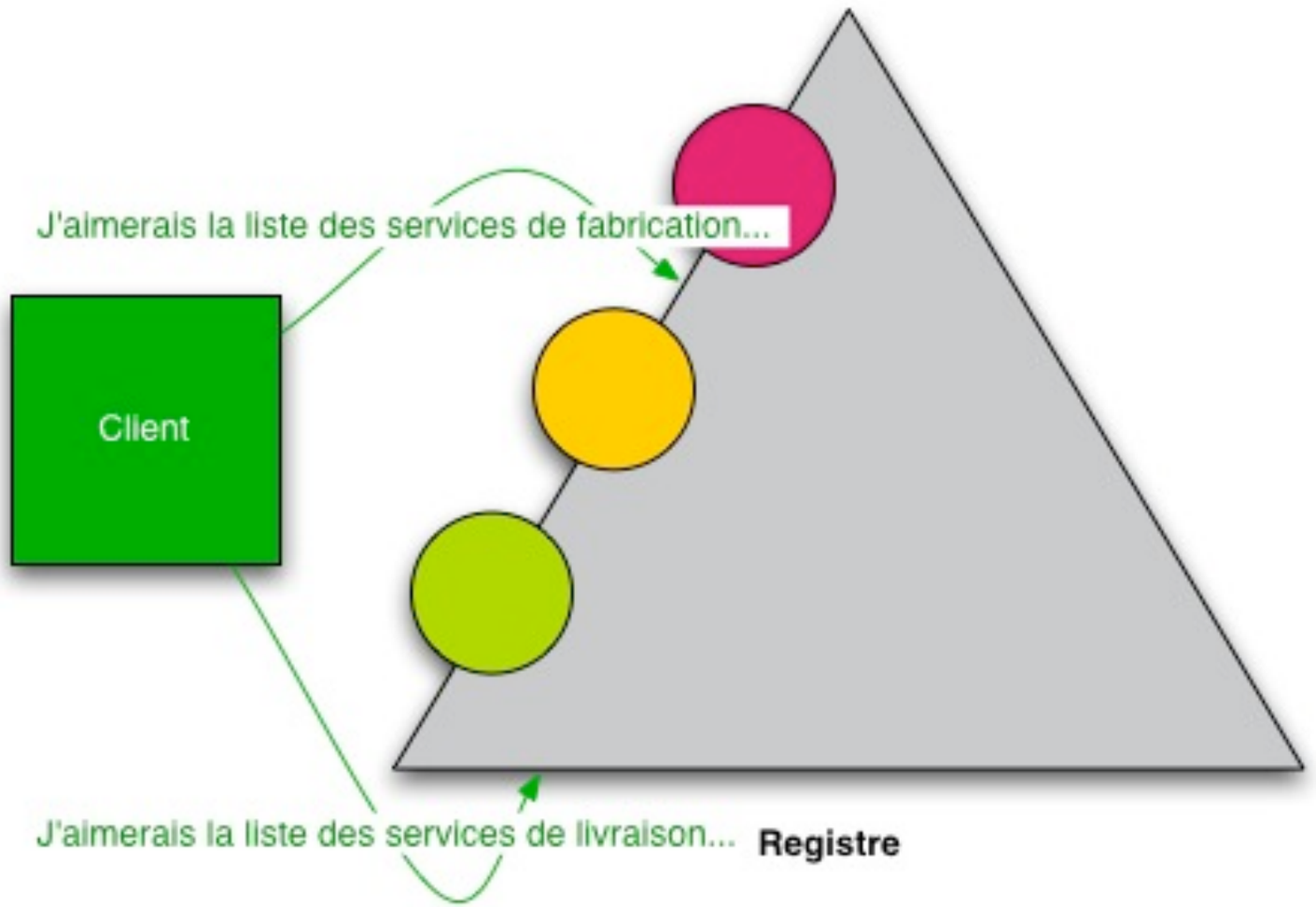
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud



"Pages jaunes"

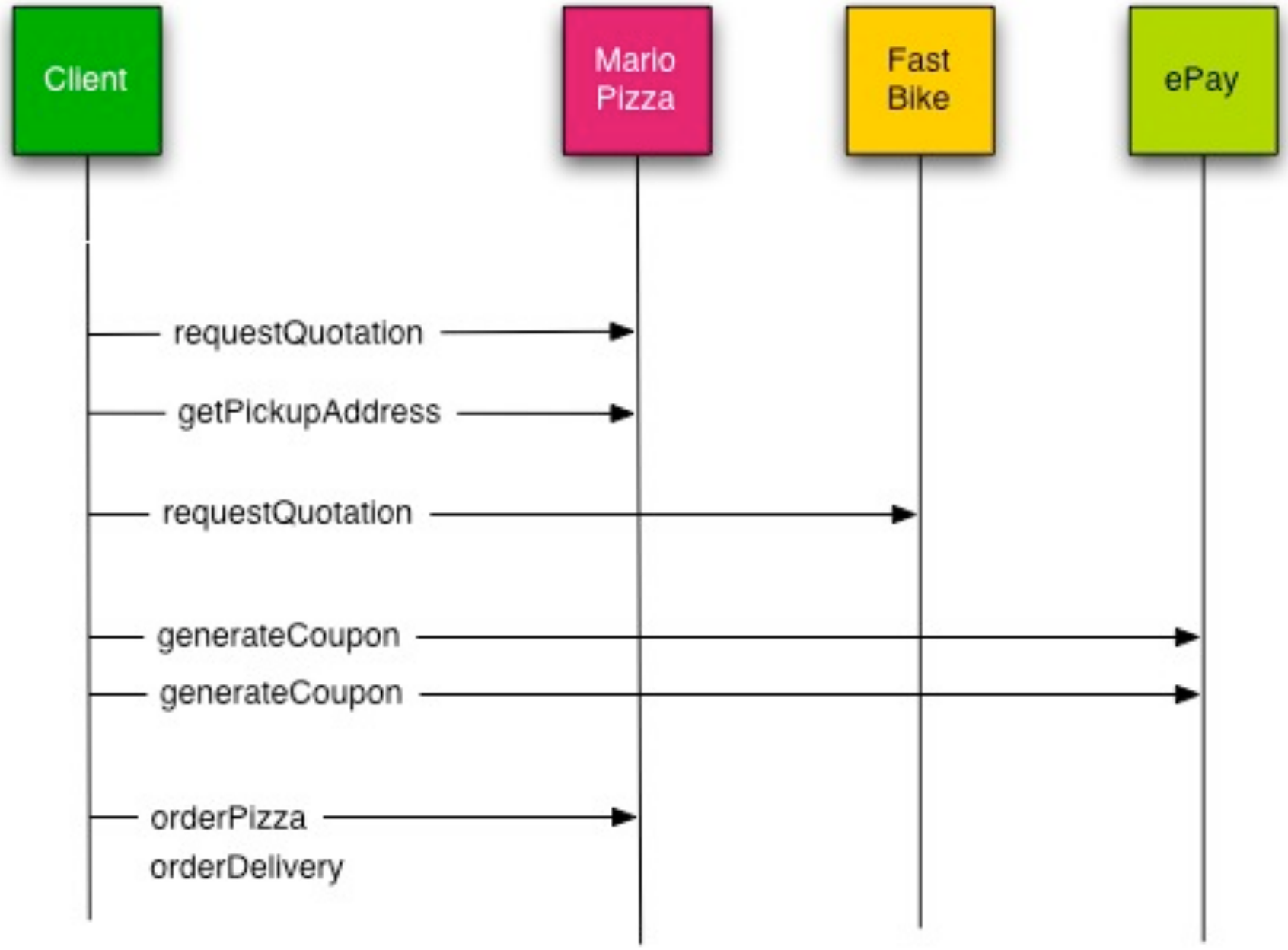
Exemple: utilisation du registre par un client

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



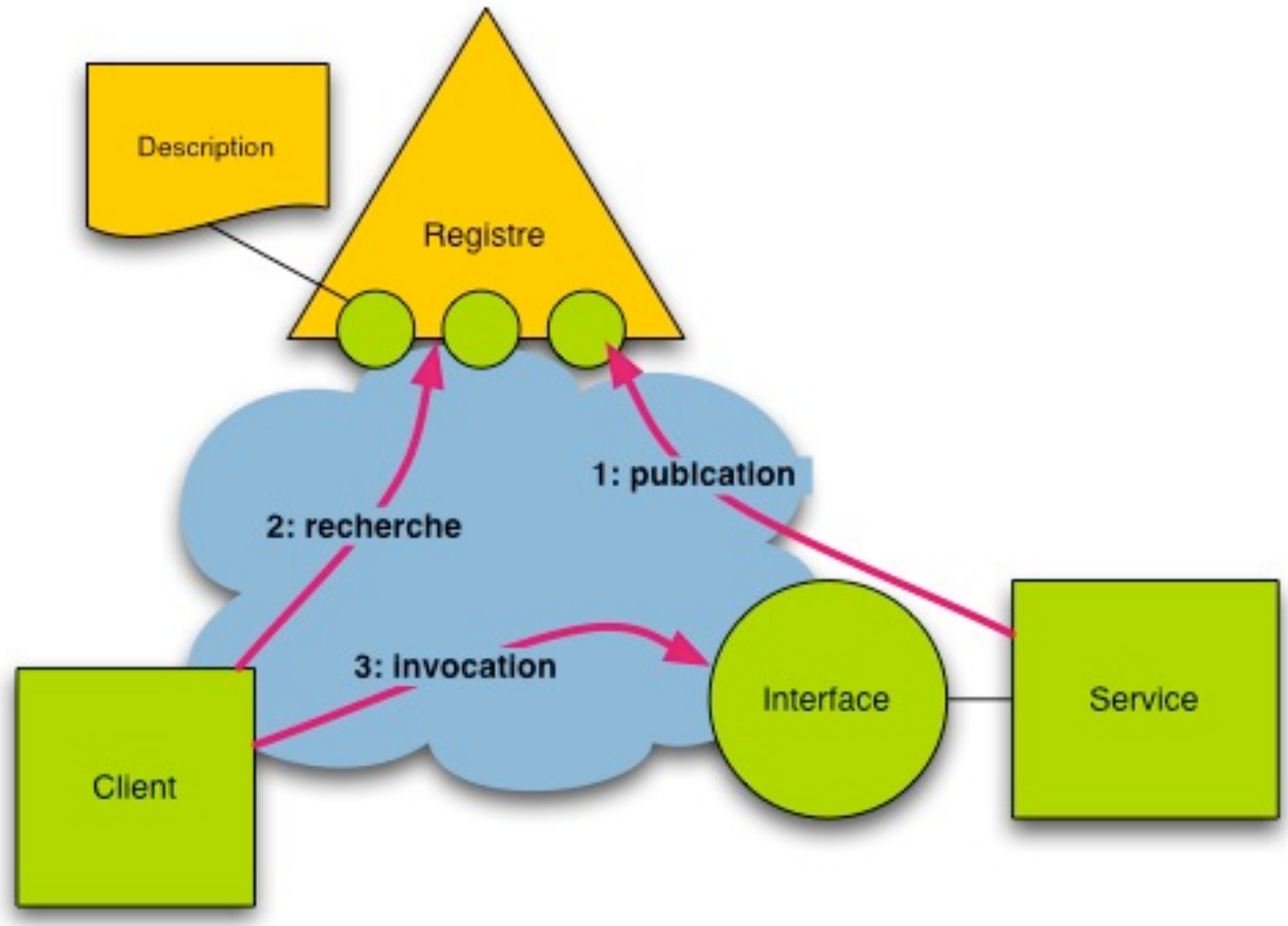
Exemple: processus complet

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud



Architecture générale

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud



Pour utiliser des services, on a besoin...

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

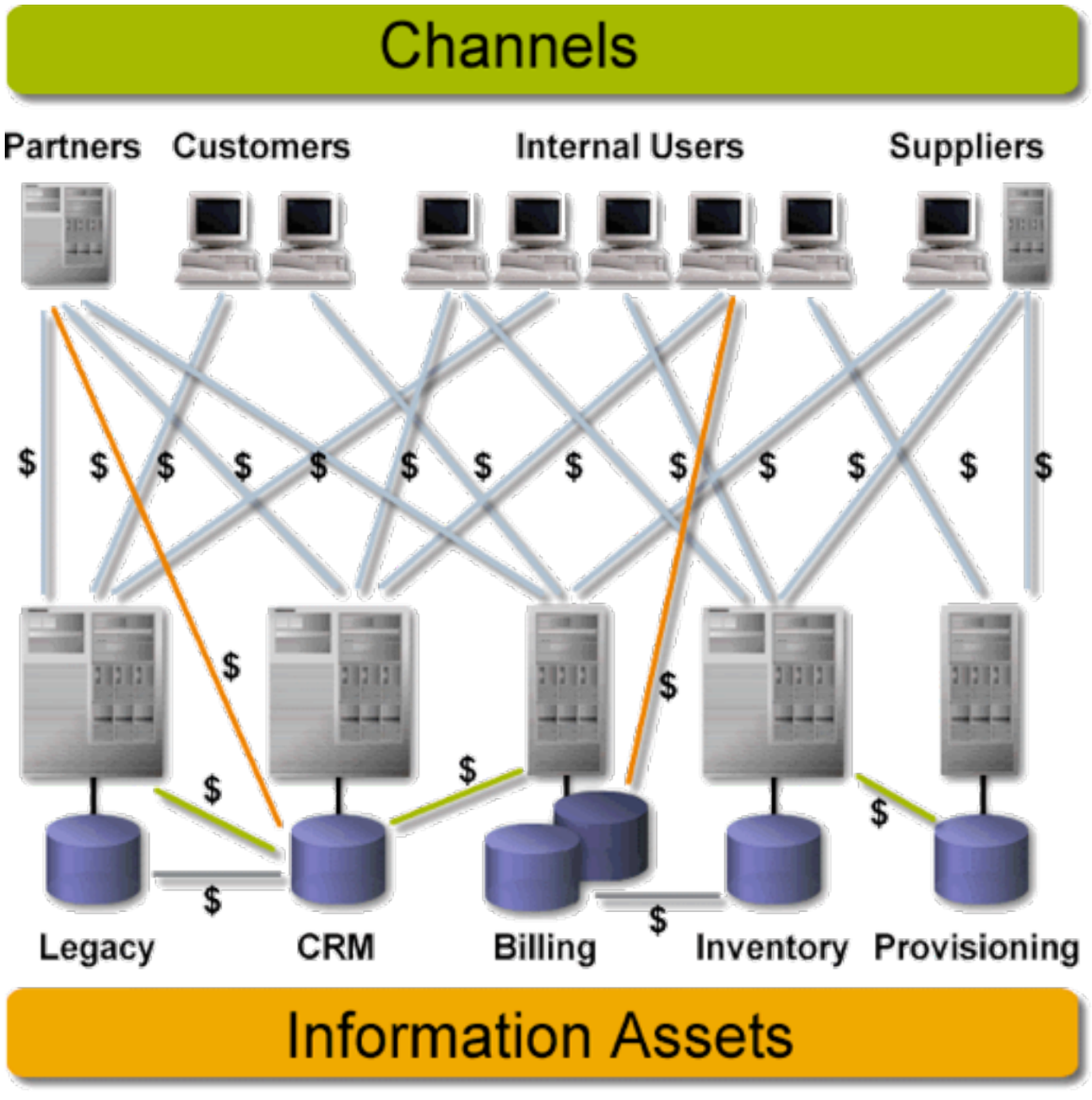
- D'un format standardisé pour décrire l'interface des services
  - Nous parlerons plus tard de WSDL, qui en est un exemple.
- D'un protocole standardisé pour invoquer un service
  - Nous parlerons plus tard de SOAP, qui en est un exemple.
- D'un mécanisme standardisé pour répertorier des services
  - Nous parlerons plus tard de UDDI, qui en est un exemple.

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Architecture Orientée Services (SOA)

Architecture "accidentelle"

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

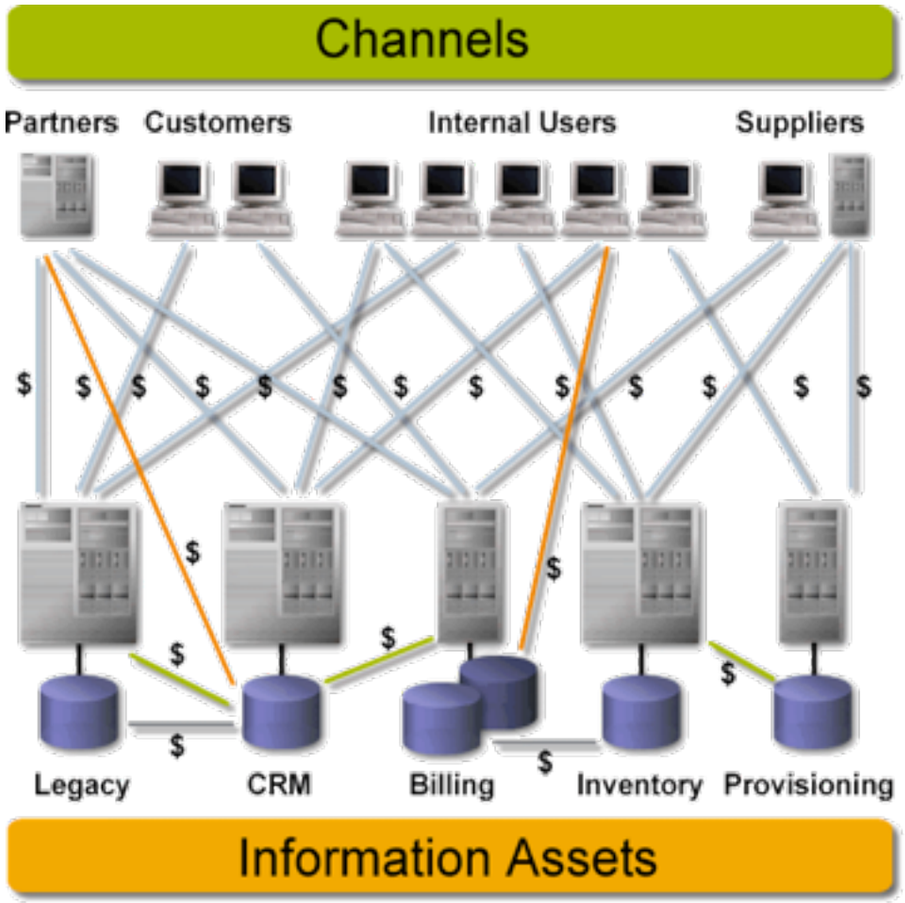


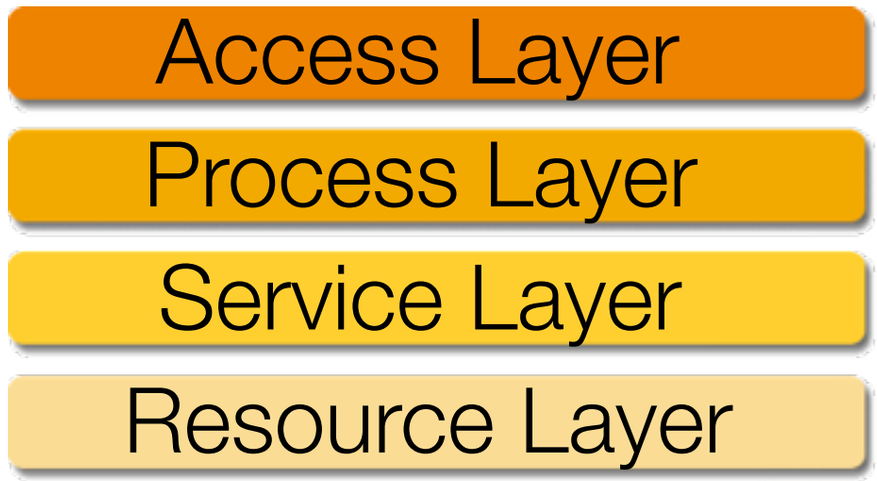
© *Sun Microsystems*

Rigide
Complexe
Monolithique
Difficile à adapter
Difficile à intégrer
Coûteuse

Vers une architecture en couches

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

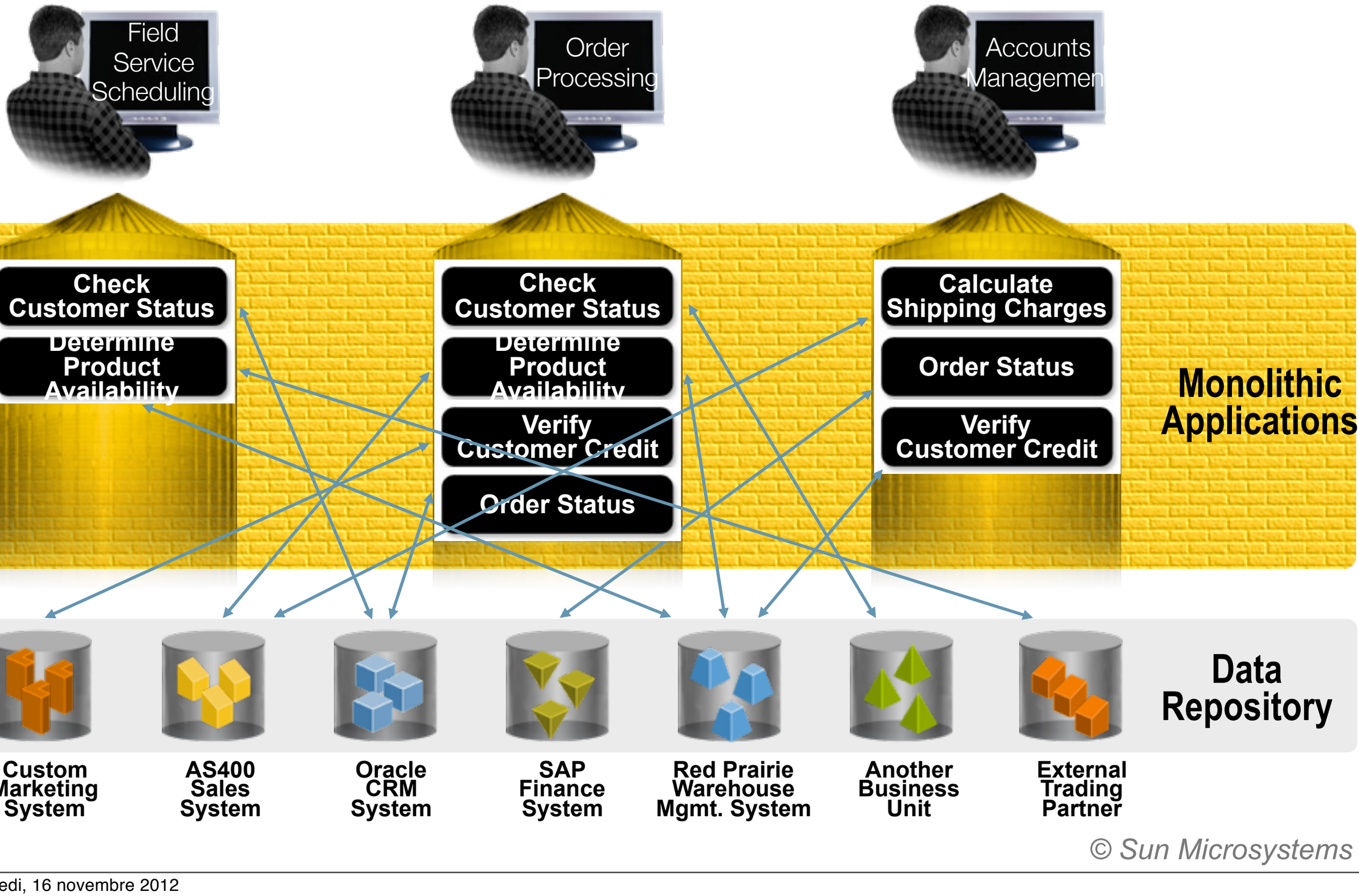




© Sun Microsystems

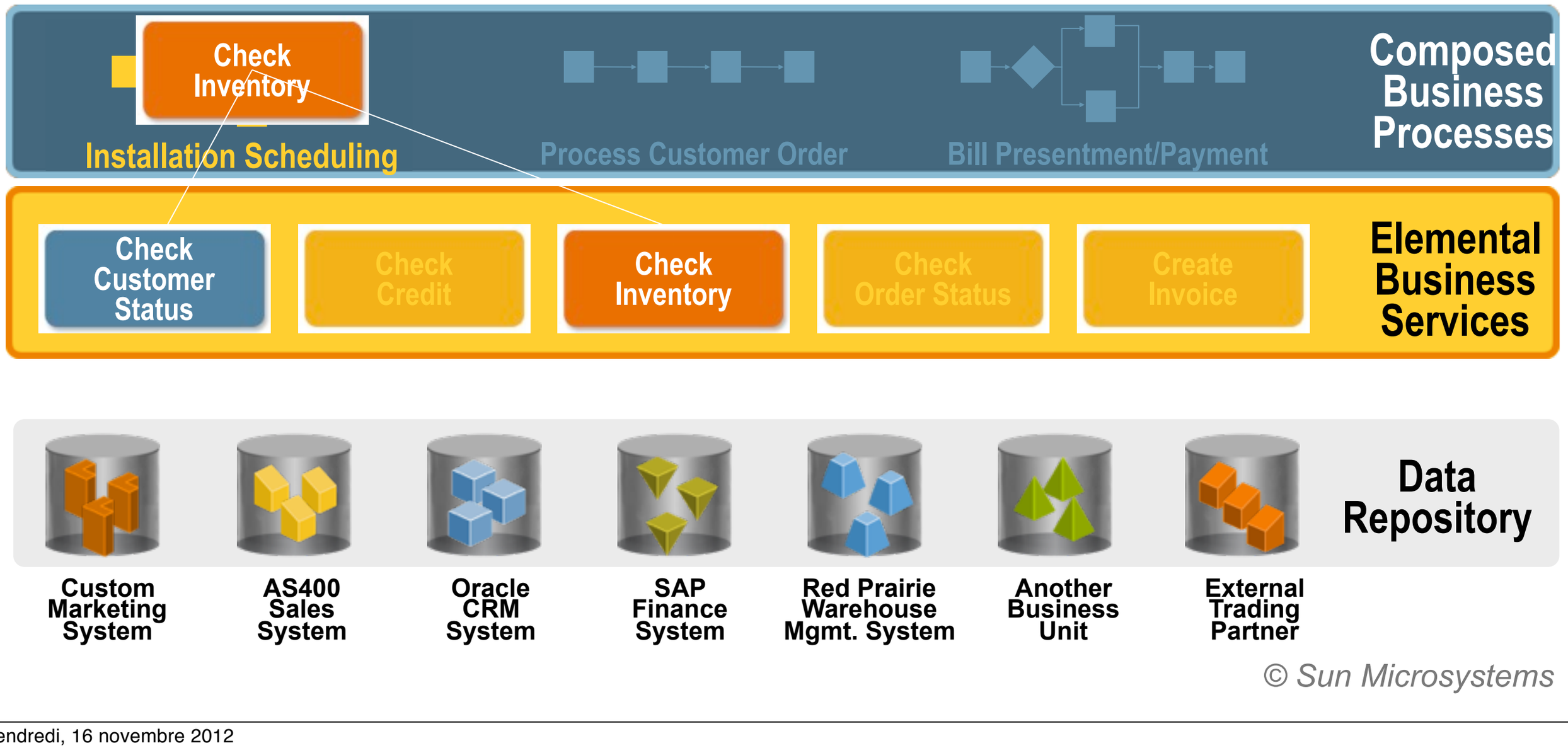
Architecture: les silos applicatifs

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



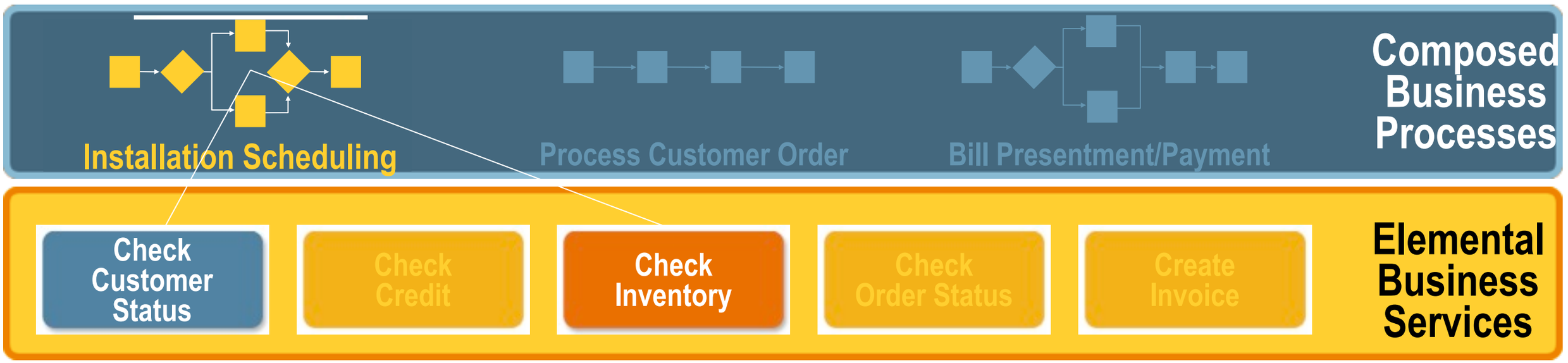
Composition de services partagés

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



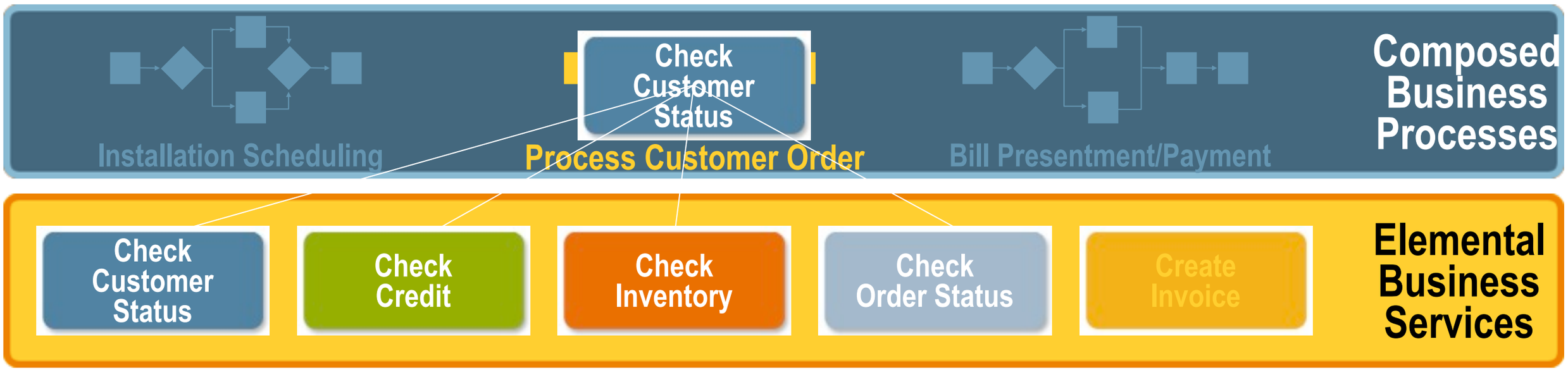
Composition de services partagés

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



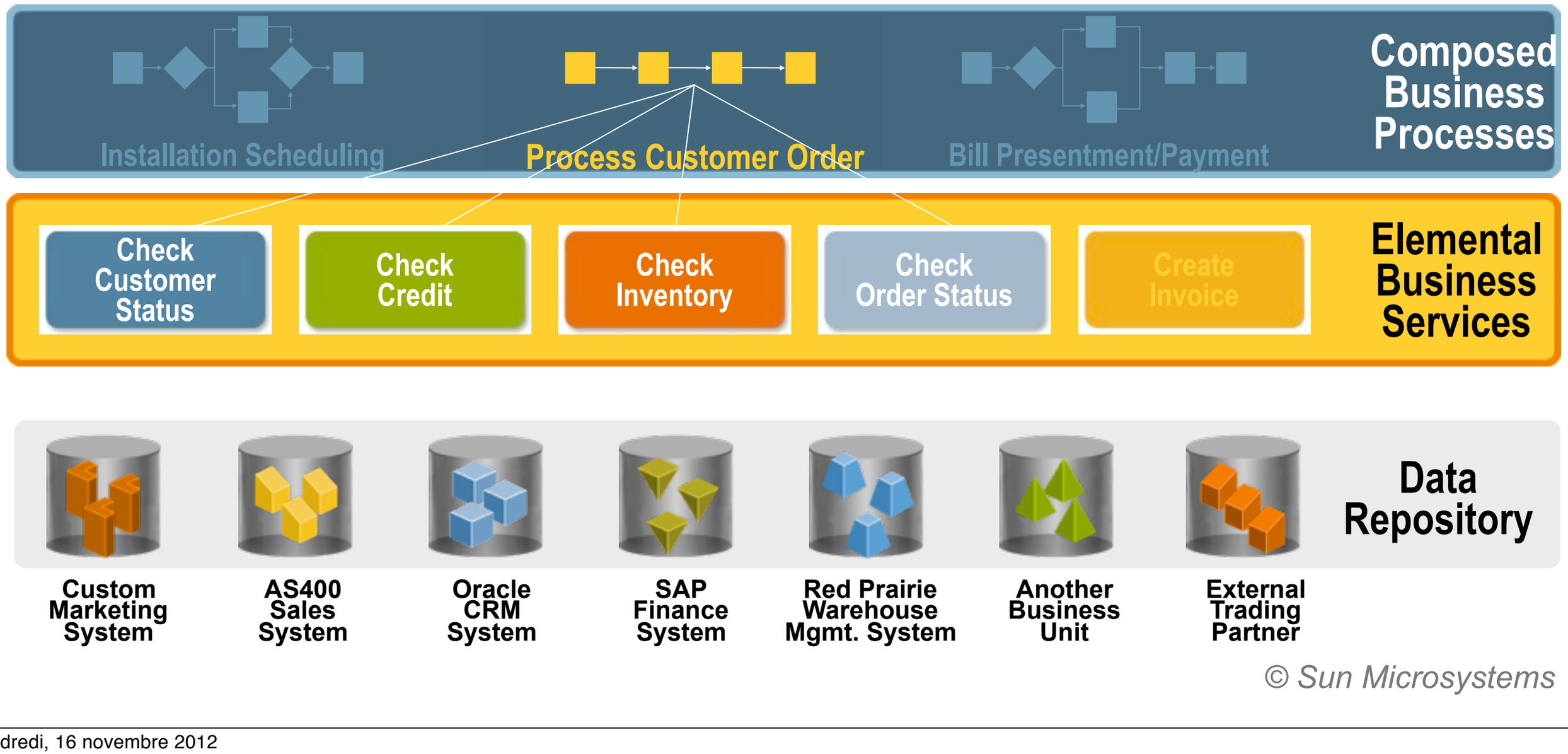
Composition de services partagés

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



Composition de services partagés

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Le cas particulier des "Web Services"

La notion de "Web Service"

- La notion de "Web Service" fait référence à un ensemble de technologies qu'on peut utiliser pour concevoir un service:
  - Des technologies qui s'appuient sur les protocoles du Web (URI, HTTP, XML).
  - Ces technologies ne sont pas liées à une plate-forme (Java, .NET, PHP, etc.).
- La notion de "Web Service" met en évidence l'évolution du Web:
  - Au début, le Web était décrit comme une grande "bibliothèque" dans laquelle des humains pouvaient naviguer.
  - De plus en plus, le Web est vu comme un ensemble de services, qui peuvent être utilisés par des machines (automatisation).

Web Services: interopérabilité ou Tour de Babel?

heig-vd
Haute École d'Ingénierie et de Gestion du Canton de Vaud

- Il existe différentes approches et différentes technologies dans le monde des "Web Services"
  - L'approche "RPC" avec par exemple SOAP
  - L'approche dite du "transfert d'état" ou REST
- Les protocoles de base (e.g. SOAP/WSDL) sont (relativement) simples, mais:
  - Ils ne couvrent pas tous les aspects nécessaires pour développer des systèmes transactionnels, fiables et sécurisés.
  - Ils ont donc dû être étendus avec de nombreux standards complémentaires.
  - Ces standards sont définis par des organisations et consortiums différents, avec des enjeux commerciaux et politiques.
  - En conséquence, il n'est pas aisé de s'y retrouver dans la liste des standards "Web Services"

Web Services Standards Overview

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Phone +49 2102 77 162-100
info@innoq.com · www.innoq.com

innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Phone +41 41 743 01 11

http://www.innoq.com/soa/ws-standards/poster/
http://www.innoq.com/soa/ws-standards/

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

SOAP

Simple Object Access Protocol

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- Description
  - SOAP est un protocole "léger" qui permet d'invoquer une méthode sur un service distant.
  - SOAP définit la structure de messages en s'appuyant sur XML. SOAP n'est pas lié à une plateforme logicielle particulière (Java, .NET, etc.).
  - Les messages SOAP peuvent être échangés avec différents protocoles de transport. HTTP est un de ces protocoles de transport.
- Historique
  - SOAP a été spécifié suite à l'explosion de XML, en 1998.
- Spécifications
  - La spécification SOAP 1.2 est une recommendation du W3C (27 avril 2007)
  - http://www.w3.org/TR/soap/

SOAP Version 1.2

Latest version of SOAP Version 1.2 specification: http://www.w3.org/TR/soap12

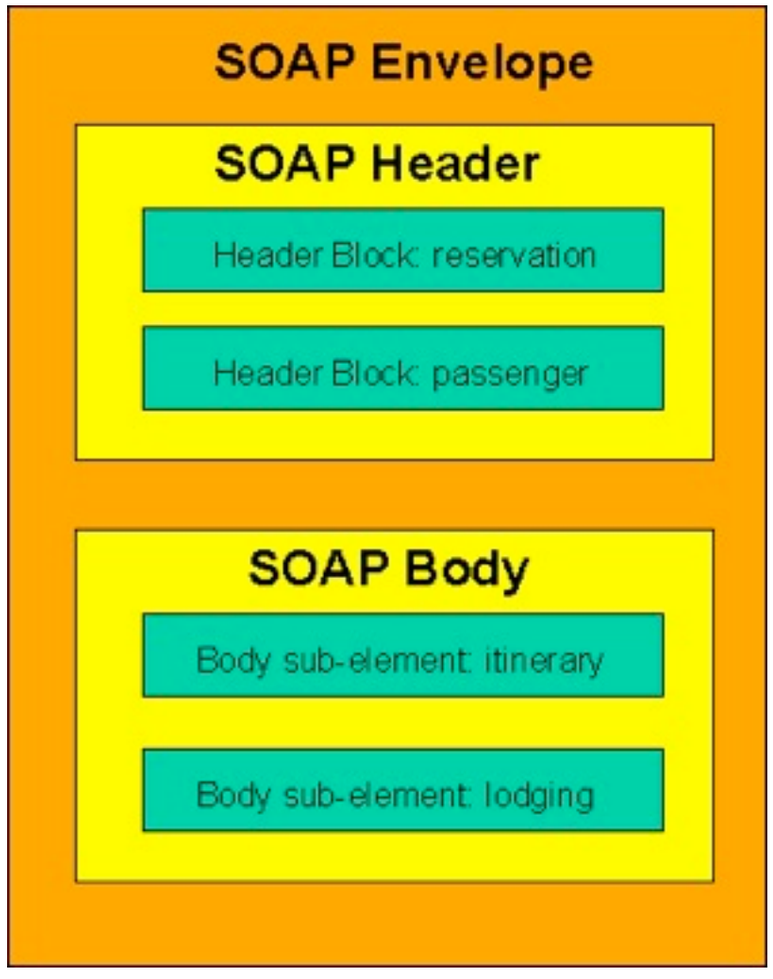
W3C Recommendation (Second Edition) 27 April 2007

SOAP Version 1.2 Part0: Primer
http://www.w3.org/TR/2007/REC-soap12-part0-20070427/ (errata)
SOAP Version 1.2 Part1: Messaging Framework
http://www.w3.org/TR/2007/REC-soap12-part1-20070427/ (errata)
SOAP Version 1.2 Part2: Adjuncts
http://www.w3.org/TR/2007/REC-soap12-part2-20070427/ (errata)
SOAP Version 1.2 Specification Assertions and Test Collection
http://www.w3.org/TR/2007/REC-soap12-testcollection-20070427/ (errata)

Structure d'un message SOAP

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- Entête
  - Est utilisée pour décrire certaines "propriétés" du message ou de l'échange.
  - Exemple: gestion de la sécurité.
  - Un des points d'extension du protocole (toutes les "propriétés" n'ont pas été définies à l'avance).
- Body
  - Contient les données applicatives à proprement parler.
  - A l'origine: une message SOAP encodait un appel de méthode (nom de méthode + paramètres).
  - De plus en plus, on utilise SOAP pour transférer des "documents" qui sont traités par le service (p.ex. une "commande client".



http://www.w3.org/TR/2007/REC-soap12-part0-20070427/

Exemple: l'API Flickr

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- **Format de requête SOAP**

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope
        xmlns:s="http://www.w3.org/2003/05/soap-envelope"
        xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema">

        <s:Body>
                <x:FlickrRequest xmlns:x="urn:flickr">
                        <method>flickr.test.echo</method>
                        <name>value</name>
                </x:FlickrRequest>
        </s:Body>

</s:Envelope>
```

Exemple: l'API Flickr

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- **Format de réponse SOAP**

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope
        xmlns:s="http://www.w3.org/2003/05/soap-envelope"
        xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema">

        <s:Body>
                <x:FlickrResponse xmlns:x="urn:flickr">
                        [escaped-xml-payload]
                </x:FlickrResponse>
        </s:Body>

</s:Envelope>
```

Exemple: l'API Flickr

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- **Format de réponse SOAP (erreur)**

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
    <s:Body>
        <s:Fault>
            <faultcode>flickr.error.[error-code]</faultcode>
            <faultstring>[error-message]</faultstring>
            <faultactor>
                http://www.flickr.com/services/soap/
            </faultactor>
            <details>
                Please see
                http://www.flickr.com/services/docs/
                for more details
            </details>
        </s:Fault>
    </s:Body>
</s:Envelope>
```

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

WSDL

WSDL: quels sont les besoins?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- SOAP est utile:
  - quand on sait où le service qu'on veut utiliser se trouve
  - quand on connaît la signature des "méthodes" qu'on veut invoquer
  - SOAP alors de faire un appel et de recevoir une réponse.
- Mais SOAP ne permet pas:
  - de rechercher un service qui soit conforme à une certaine interface
  - de générer automatiquement un "client" (on parle de stub) qui soit capable d'interagir avec le service.
  - SOAP n'est pas utilisé pour décrire l'interface des services; on a besoin d'un autre protocole pour cela!
- WSDL: Web Services Description Language répond à ce besoin:
  - en permettant la description formelle de services
  - et ainsi en permettant une automatisation de l'utilisation des services (découverte, génération de client, invocation, etc.)

As communications protocols and message formats are standardized in the web community, it becomes increasingly possible and important to be able to **describe the communications in some structured way**.

WSDL addresses this need by defining an XML grammar for **describing network services as collections of communication endpoints capable of exchanging messages**.

WSDL service definitions provide **documentation for distributed systems** and serve as a recipe for **automating** the details involved in applications communication.

http://www.w3.org/TR/wsdl

Structure WSDL

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>
    <import namespace="uri" location="uri"/>*
    <wsdl:documentation .... /> ?

    <wsdl:types> ?
        <wsdl:documentation .... />?
        <xsd:schema .... />*
        <-- extensibility element --> *
    </wsdl:types>

    <wsdl:message name="nmtoken"> *
        <wsdl:documentation .... />?
        <part name="nmtoken" element="qname"? type="qname"?/> *
    </wsdl:message>

    <wsdl:portType name="nmtoken">*
        <wsdl:documentation .... />?
        <wsdl:operation name="nmtoken">*
            <wsdl:documentation .... /> ?
            <wsdl:input name="nmtoken"? message="qname">?
                <wsdl:documentation .... /> ?
            </wsdl:input>
            <wsdl:output name="nmtoken"? message="qname">?
                <wsdl:documentation .... /> ?
            </wsdl:output>
            <wsdl:fault name="nmtoken" message="qname"> *
                <wsdl:documentation .... /> ?
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:portType>
```

Type: définition des types de données: Person, Contract, Account, etc.

Message: représentation abstraite des données échangées

Port: ensemble d'opérations. Chaque opération a des messages d'input et d'output.

Structure WSDL

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

```
<wsdl:binding name="nmtoken" type="qname">*
    <wsdl:documentation .... />?
    <-- extensibility element --> *
    <wsdl:operation name="nmtoken">*
       <wsdl:documentation .... /> ?
       <-- extensibility element --> *
       <wsdl:input> ?
           <wsdl:documentation .... /> ?
           <-- extensibility element -->
       </wsdl:input>
       <wsdl:output> ?
           <wsdl:documentation .... /> ?
           <-- extensibility element --> *
       </wsdl:output>
       <wsdl:fault name="nmtoken"> *
           <wsdl:documentation .... /> ?
           <-- extensibility element --> *
       </wsdl:fault>
    </wsdl:operation>
</wsdl:binding>

<wsdl:service name="nmtoken"> *
    <wsdl:documentation .... />?
    <wsdl:port name="nmtoken" binding="qname"> *
       <wsdl:documentation .... /> ?
       <-- extensibility element -->
    </wsdl:port>
    <-- extensibility element -->
</wsdl:service>
<-- extensibility element --> *
</wsdl:definitions>
```

Binding: spécification du protocole et du format de données concret qui sera utilisé par rapport à un port particulier.

Service: ensemble de ports.

Exemple WSDL: binding SOAP

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

    <types>
       <schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
          <element name="TradePriceRequest">
             <complexType>
                 <all>
                     <element name="tickerSymbol" type="string"/>
                 </all>
             </complexType>
          </element>
          <element name="TradePrice">
             <complexType>
                 <all>
                     <element name="price" type="float"/>
                 </all>
             </complexType>
          </element>
       </schema>
    </types>
```

Exemple WSDL: binding SOAP

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

```
<message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
       <input message="tns:GetLastTradePriceInput"/>
       <output message="tns:GetLastTradePriceOutput"/>
    </operation>
</portType>

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

Exemple WSDL: binding SOAP


```
<message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
</message>
<message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
       <input message="tns:GetLastTradePriceInput"/>
       <output message="tns:GetLastTradePriceOutput"/>
    </operation>
</portType>

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

<service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteBinding">
       <soap:address location="http://example.com/stockquote"/>
    </port>
</service>
</definitions>
```


olivier.liechti@heig-vd.ch

UDDI

olivier.liechti@heig-vd.ch

Universal Description Discovery and Integration

heig-vd
Haute École d'Ingénierie et de Gestion du Canton de Vaud

- Avec WSDL et SOAP, nous pouvons maintenant:
  - Décrire formellement des services (ce qui permet à un "programme" d'analyser l'interface du service et de l'utiliser de manière automatique).
  - Invoquer le service en passant des paramètres, afin de recevoir une réponse (qui pourra également être traitée par un "programme").
- Pour compléter notre "architecture" orientée services, nous avons encore besoin:
  - D'un mécanisme pour répertorier les services (annuaire de services).
  - Ce mécanisme doit être utilisable par des "programmes"
  - Un service doit pouvoir s'enregistrer dans l'annuaire
  - Un client doit pouvoir interroger l'annuaire pour obtenir la référence d'un service qui répond à ses besoins.
- UDDI est une spécification qui répond à ce besoin
  - ebXML est une autre approche qui répond au même besoin
  - JAXR est une interface java standardisée pour interagir avec ces registres.

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Développement de Web Services en Java (SOAP)

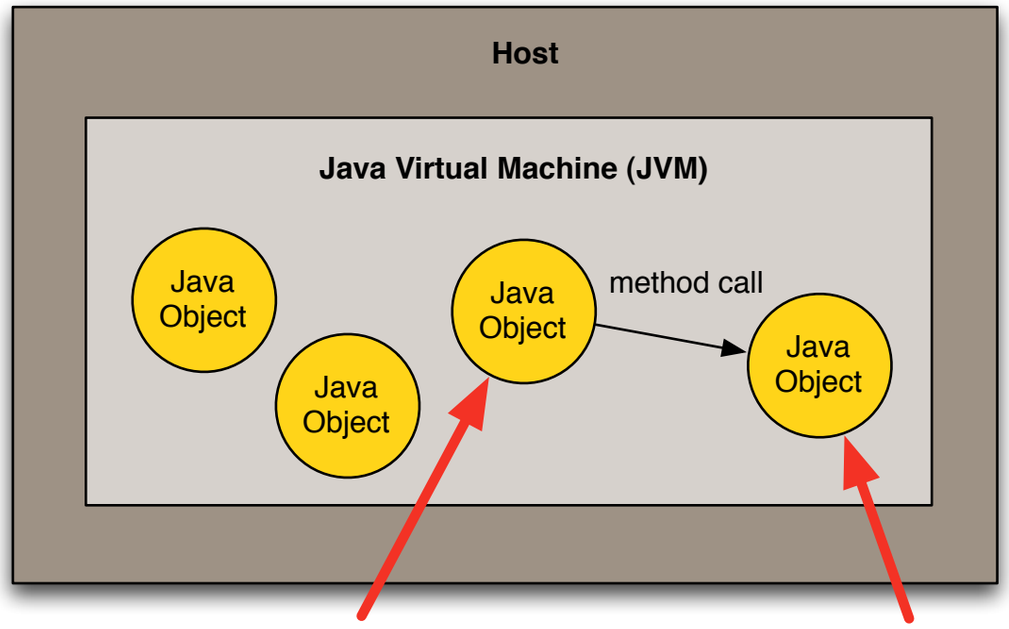
Vous aimez XML? Vous aimez les parsers?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- On pourrait très bien écrire du code pour:
  - générer des appels SOAP dans un client
  - traiter des appels SOAP dans un service
- Mais en choisissant ses outils, on peut éviter du travail inutile et sujet à erreurs:
  - Les IDEs (NetBeans, eclipse) fournissent des outils qui facilitent la création et l'utilisation de Web Services.
  - Les plate-formes modernes (Java EE, .NET) intègrent la notion de Web Service et prennent en charge une grande partie de la "plomberie".
  - De nombreux outils et framework Open Source complètent cette offre.
- En conséquence:
  - Il est très facile d'implémenter et de déployer un web service simple.
  - Il est très facile d'implémenter un client pour ce web service.
  - Les choses se compliquent quand même un peu quand on considère des "vrais" services.

Service à l'intérieur d'une JVM

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud




```
public class ClockClient

  private ClockService service;

...

  public String showTime() {
    System.out.println("It is " +
      service.getTime());
  }
}
```

ClockClient.java

```
public interface ClockService {

  public String getTime();
}
```

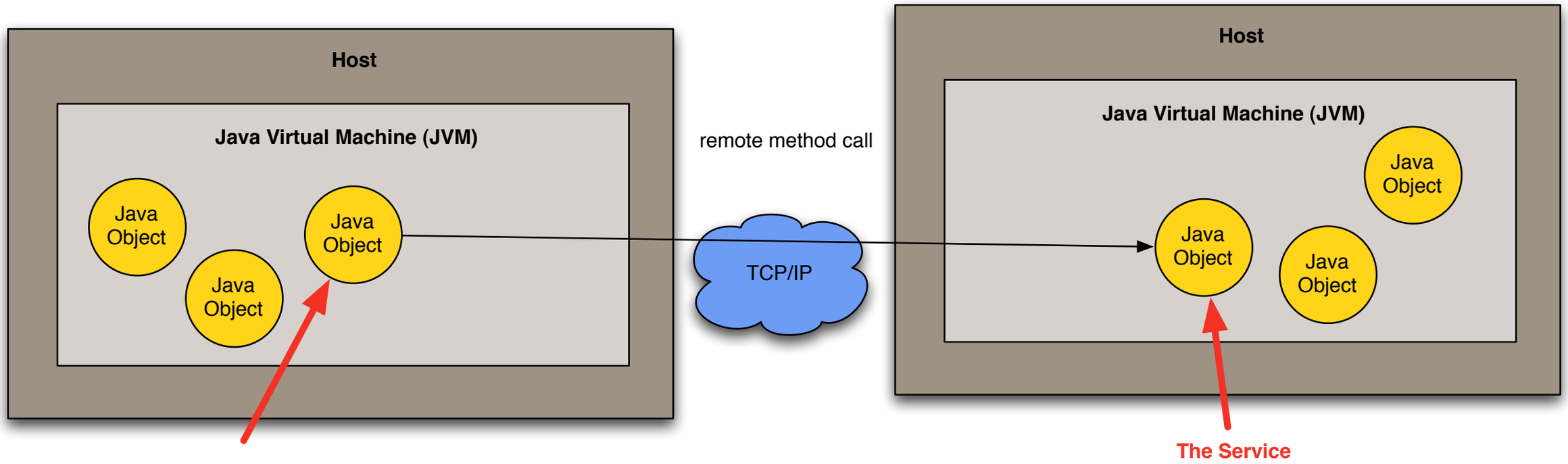
ClockService.java

```
public class ClockServiceImpl
  implements ClockService {

  public String getTime() {
    return new java.util.Date();
  }
}
```

ClockServiceImpl.java

Service dans une JVM distante

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud




```
public class ClockClient

}
```

générer du XML (SOAP) ?

ouvrir un socket?

parser du XML (réponse SOAP)

?

ClockClient.java

```
public interface ClockService {

  public String getTime();
}
```

ClockService.java

```
public class ClockServiceImp
  implements ClockService {

}
```

générer du XML (SOAP) ?

écouter sur un socket?

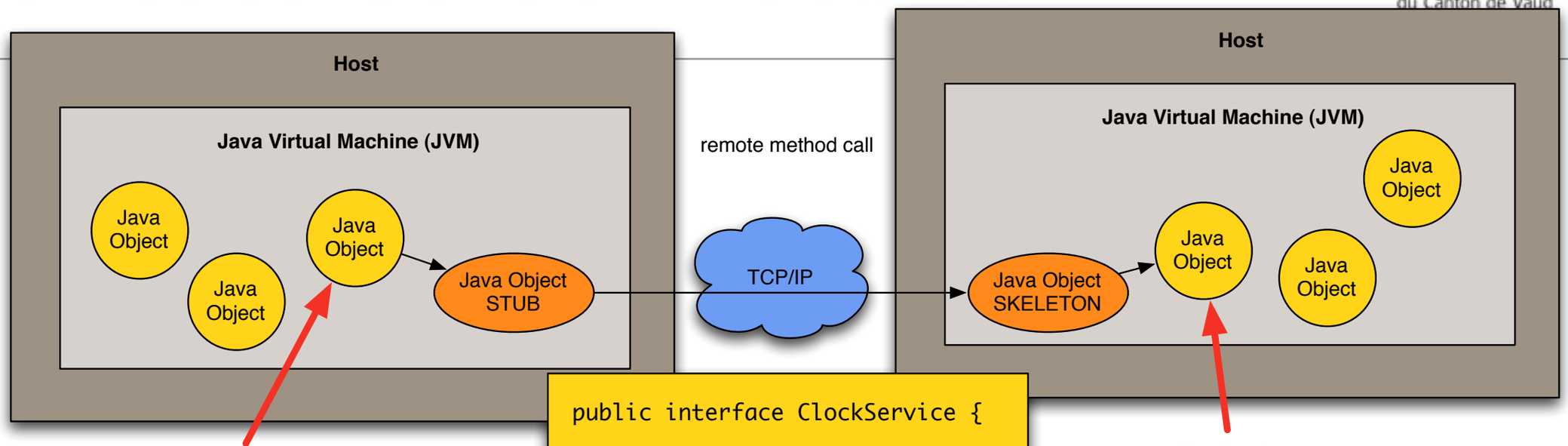
parser du XML (SOAP)

?

ClockServiceImpl.java

Service dans une JVM distante

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud




```
public interface ClockService {

  public String getTime();
}
```

ClockService.java

```
public class ClockClient

  private ClockService service;

...

  public String showTime() {
    System.out.println("It is " +
      service.getTime());
  }
}
```

ClockClient.java

```
public class ClockServiceStub
  implements ClockService {

  public String getTime() {
    ... bla bla... SOCKET ...
    XML... bla bla bla... SOAP ...
    bla bla OUTPUTSTREAM... bla bla
    INPUTSTREAM....

    return time;
}
```

ClockServiceStub.java

```
public class ClockServiceSkeleton {

  private ClockService realService;

  private void setup() { ... }
  private void listen() { ... }
  private void processCall {
    realService.getTime();
  }
}
```

ClockServiceSkeleton.java

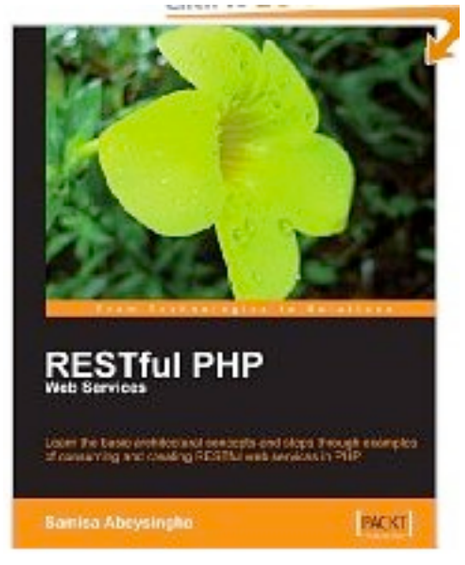
```
public class ClockServiceImpl
  implements ClockService {

  public String getTime() {
    return new java.util.Date();
  }
}
```

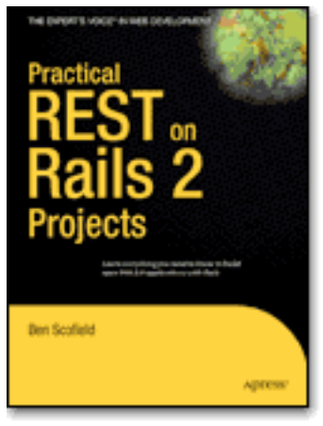
ClockServiceImpl.java

Generated automatically by special tools!

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Les Web Services “REST”





Le style d’architecture REST

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- REST: REpresentational State Transfer
- REST est un style d'architecture que l'on peut utiliser pour construire des systèmes distribués.
- REST a été décrit dans la thèse de doctorat de Roy Fielding (qui a contribué à la spécification HTTP, au serveur apache, à la communauté apache).
- Le WWW est un exemple de système distribué qui est construit selon les principes d'une architecture REST.

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

HTTP est utilisé pour interagir avec des "ressources"

Qu'est-ce qu'une "ressource"?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- A première vue, on pourrait penser qu'une "ressource" est un "fichier" stocké sur un serveur web:
  - un document HTML, un document XML, une image PNG, etc.
- Cette définition est appropriée pour le web "statique":
  - le client envoie une requête HTTP au serveur;
  - la requête spécifie une URL qui identifie un document (e.g. /dir1/menu.html)
  - le serveur renvoie le contenu du document au client.
- Mais le web est bien plus qu'une simple "bibliothèque" de documents:
  - au travers du web, nous interagissons avec des services; le contenu des réponses envoyées au client est alors généré dynamiquement;
  - au travers du web, nous interagissons avec le monde "physique".
  - Nous avons donc besoin d'une définition plus large pour le terme de "ressource"...

Rappel: qu'est-ce qu'une "ressource"?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- Une ressource est "quelque chose" que l'on peut nommer et identifier de manière unique:
  - Exemple 1: un article particulier publié dans le journal "24 heures"
  - Exemple 2: la collection des articles publiés dans une rubrique du journal
  - Exemple 3: le CV d'une personne
  - Exemple 4: le cours actuel de l'action Nestlé
  - Exemple 5: le distributeur de boissons dans le hall de l'école
  - Exemple 6: la liste des notes obtenues par l'étudiant Jean Dupont
- Une URL (Uniform Resource Locator) est une chaîne de caractères qui permet d'identifier et de localiser une ressource:
  - Exemple 1: http://www.24heures.ch/vaud/vaud/2008/08/04/trente-etudiants-partent-rencontre-patrons
  - Exemple 2: http://www.24heures.ch/articles/vaud/actu
  - Exemple 5: http://www.smart-machines.ch/customers/heig/machines/8272

Ressource vs. représentation

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- Une "ressource" peut donc être quelque chose d'abstrait (de l'information, un concept, un état) ou quelque chose qui a une existence tangible dans le monde physique (un objet, une personne).
- Le protocole HTTP permet le transfert d'information entre un client et un serveur.
- L'information qui est transférée lors d'un échange HTTP n'est pas une ressource. C'est la représentation d'une ressource!
- La représentation d'une ressource est un ensemble de données (bytes) accompagné d'un ensemble de méta-données décrivant les données:
  - Un document HTML, une image, un son sont des exemples de représentations de ressources.
  - Exemples de méta-données: définition du type, longueur, encodage, etc.
  - Une ressource peut avoir plusieurs représentation:
    - Exemple 1: la ressource identifiée par l'URL "http://www.abc.com/monCV" peut avoir une représentation HTML, une représentation PDF, etc.

Interagir avec une ressource?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- HTTP définit un ensemble de "méthodes"
- Ces méthodes sont des "verbes" qu'on peut appliquer à une resource:
  - GET: retrieve whatever information is identified by the Request-URI
  - POST: used to request that the origin server accept the entity enclosed in the request as a new subordinate of the ressource identified by the Request-URI in the Request-Line
  - PUT: requests that the enclosed entity be stored under the supplied Request-URI.
  - DELETE: requests that the origin server delete the ressource identified by the Request-URI.
  - HEAD: identical to GET except that the server MUST NOT return a message-body in the response
  - TRACE: used for debugging (echo)
  - CONNECT: reserved for tunneling purposes

Principes d'une architecture REST

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- L'état de l'application est défini dans un ensemble de ressources:
  - Utilisateurs, photos, commentaires, tags, albums, etc.
- Toute ressource peut être référencée selon une syntaxe standard (URL)
- Une ressource peut avoir plusieurs représentations.
- La même interface (les mêmes verbes) est utilisée pour interagir avec toutes les ressources (GET, PUT, POST, DELETE)
- Le protocole de communication est:
  - client-serveur
  - sans état (stateless)
  - cacheable
- Ces propriétés ont un effet positif sur les qualités de service (scalabilité, performance, disponibilité, etc.).
  - Référence: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Modéliser des Web Services avec REST

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

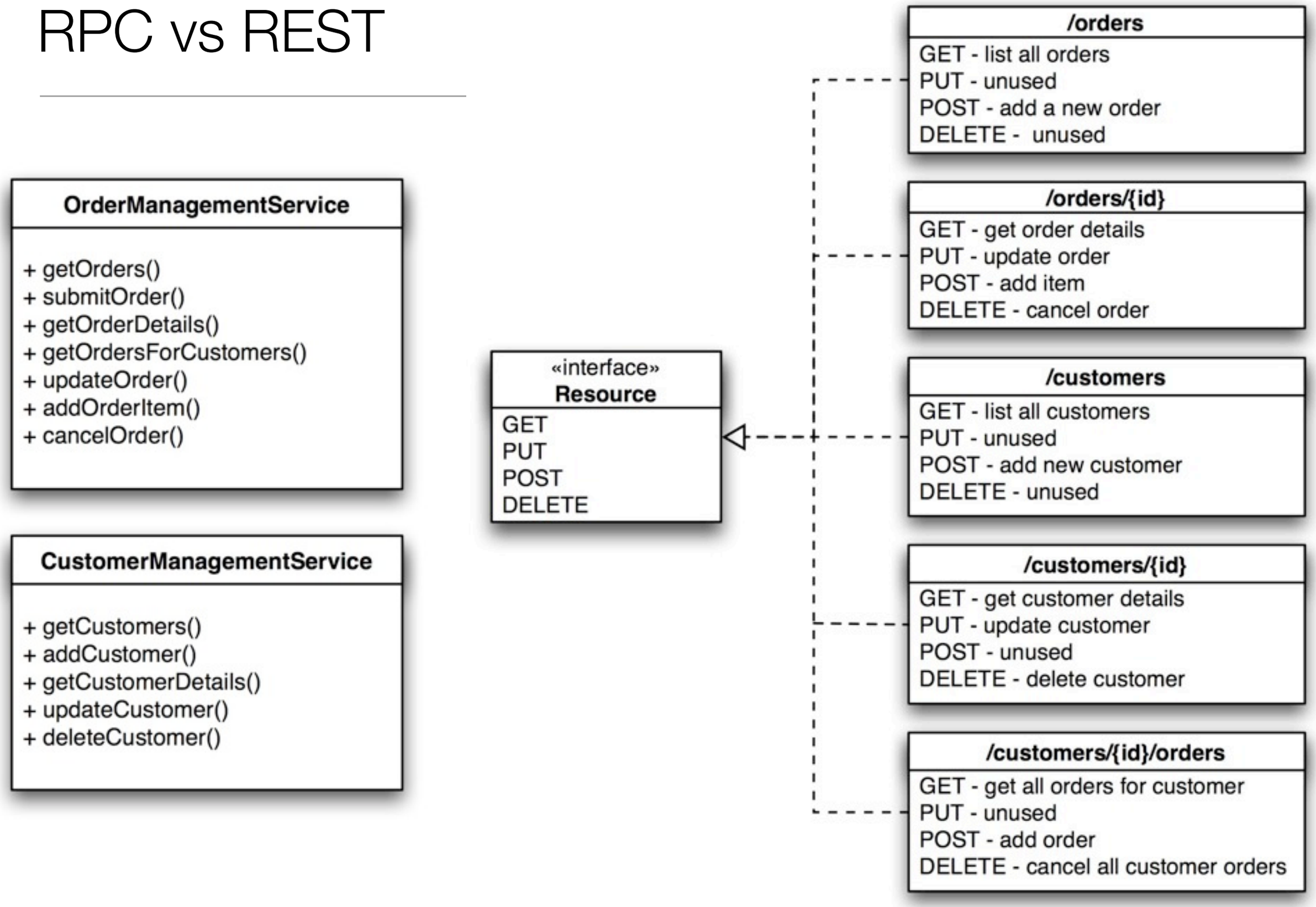
- L'approche est différente de celle utilisée dans une optique RPC.
- Les VERBES sont définis par le protocole (GET, PUT, POST, DELETE)
- On doit identifier les ressources (les NOMS).
- On doit ensuite définir la structure des URLs qui permettent d'interagir avec les NOMS.
- Quelques exemples:
  - http://www.photos.com/users/oliechti identifie une ressource de type "utilisateur". Un client peut faire un "HTTP GET" pour en obtenir une représentation ou un "HTTP PUT" pour mettre à jour l'état de l'utilisateur.
  - http://www.photos.com/users identifie une ressource de type "collection d'utilisateurs". Un client peut faire un "HTTP POST" pour ajouter un utilisateurs, ou un "HTTP GET" pour obtenir la liste des utilisateurs.

Plus de détails...

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- L'article "A Brief Introduction to REST", par Stefan Tilkov, présente les concepts principaux de manière très claire.
  - http://www.infoq.com/articles/rest-introduction
  - Cet article fait partie de la matière à étudier pour le cours et pour l'examen!

RPC vs REST


OrderManagementService
+ getOrders()
+ submitOrder()
+ getOrderDetails()
+ getOrdersForCustomers()
+ updateOrder()
+ addOrderItem()
+ cancelOrder()
CustomerManagementService
+ getCustomers()
+ addCustomer()
+ getCustomerDetails()
+ updateCustomer()
+ deleteCustomer()
«interface»
Resource
GET
PUT
POST
DELETE
/orders
GET - list all orders
PUT - unused
POST - add a new order
DELETE - unused
/orders/{id}
GET - get order details
PUT - update order
POST - add item
DELETE - cancel order
/customers
GET - list all customers
PUT - unused
POST - add new customer
DELETE - unused
/customers/{id}
GET - get customer details
PUT - update customer
POST - unused
DELETE - delete customer
/customers/{id}/orders
GET - get all orders for customer
PUT - unused
POST - add order
DELETE - cancel all customer orders

Comment écrire un "RESTful" Web Service?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- Du côté serveur, on pourrait tout faire avec des servlets, mais cela demanderait beaucoup de travail:
  - Parser les URLs
  - Faire le mapping entre les URLs et des classes d'objets représentant les ressources
  - Gérer les différentes représentations de ressources
  - etc.
- D'autre part, cela ne nous permettrait pas de nous concentrer sur l'essentiel: les ressources définissant l'état de notre application!
- Heureusement, il existe des frameworks et outils pour nous faciliter la vie.
- C'est vrai pour différents langages de programmation, et notamment pour Java.
- Il existe même un standard Java défini dans ce but: JAX-RS, défini dans le JSR 311.
  - Sun propose une implémentation de référence, appelée Jersey.

Support des IDEs

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- NetBeans offre également des fonctions pour faciliter le développement de Web Services "RESTful".
- Nous allons voir ensemble comment utiliser ces fonctions.
- Références:
  - JAX-RS: https://jsr311.dev.java.net/nonav/releases/1.0/spec/index.html
  - Jersey: https://jersey.dev.java.net/
  - NetBeans & REST: http://www.netbeans.org/kb/60/websvc/rest.html