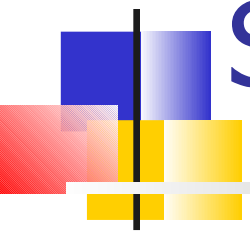
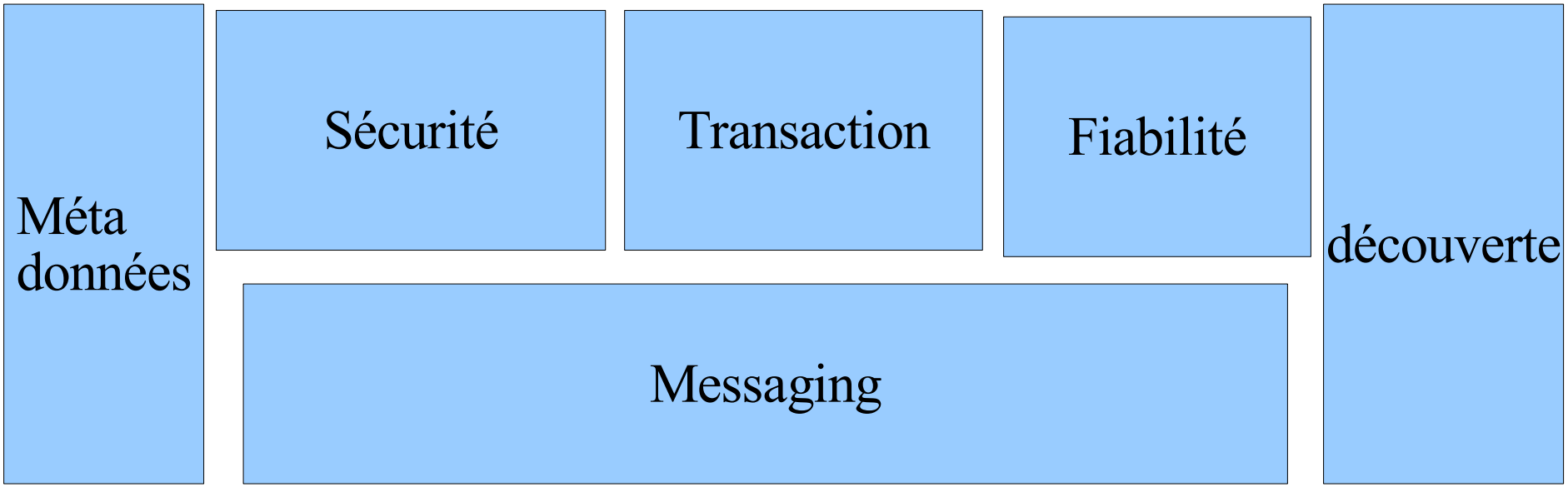


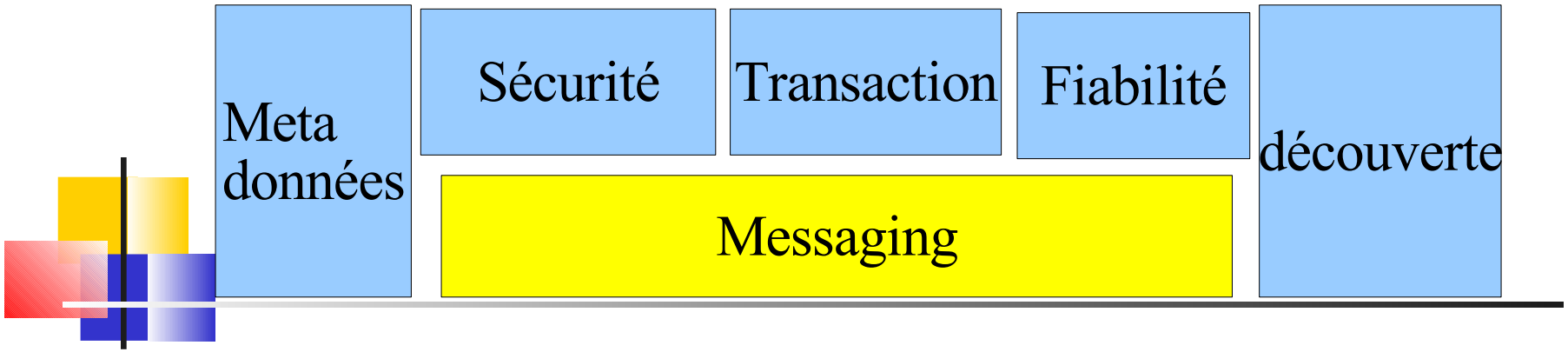
Services Web Part 2

Plan : spécifications liées aux services Web, OSGI, uGaps

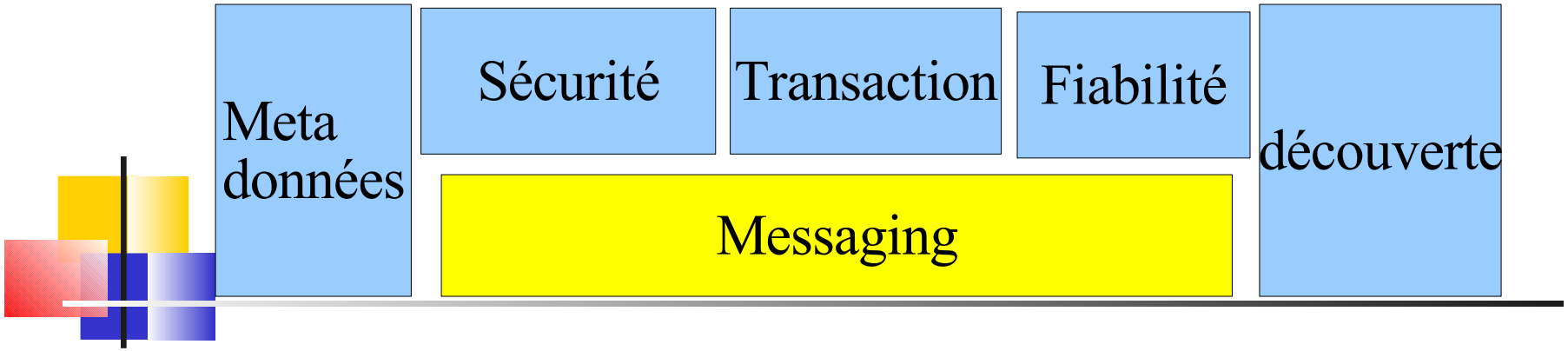
Spécifications des Services Web : WS-*

- Les spécifications fournissent les abstractions & informations pour
  - une communication fiable et sécurisée,
  - Un support transactionnel aux processus métiers



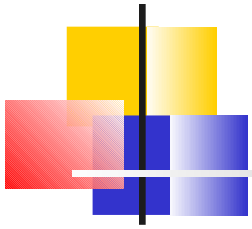


- **SOAP** permet l'échange **de messages XML** dans un environnement distribué
  - Définit les règles d'encodage et des conventions pour définir les RPC
  - S'adapte à plusieurs protocoles de transport
    - Toutefois, **HTTP** est largement sur-représenté
- SOAP laisse de nombreuses questions ouvertes ...



- Comment acheminer des données
  - ne correspondant pas à du XML?
  - Via des protocoles autres que HTTP?
  - De façon « fortement » asynchrone?
- Spécifications complémentaires
  - MTOM (*Message Transmission Optimisation Mechanism*)
    - Permet d'envoyer des données binaires (attachements)
    - fait suite à 2 spécifications DIME et WS-Attachment
  - SOAP-over-UDP, WS-Addressing /2003,
  - WS-Eventing, WS-Notification /2003

WS-Addressing (2004)



- Identifier les points d'accès d'un service Web
- Nécessaire pour sécuriser l'identification des points d'accès
- **Découpler l'identification d'un point d'accès du protocole utilisé pour transporter les messages SOAP**

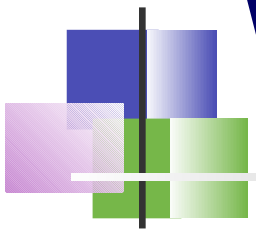
Requête SOAP

- Le point d'accès du service est identifié dans l'entête HTTP => SOAP est dépendant du protocole de transport

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml
Content-Length: nnnn
SOAPMethodName: Some-Namespace-URI#GetLastTradePrice

<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1”>
    <SOAP:Body>
        <m:GetLastTradePrice xmlns:m="Some-Namespace-URI”>
            <symbol>DIS</symbol>
        </m:GetLastTradePrice>   </SOAP:Body></SOAP:Envelope>
```

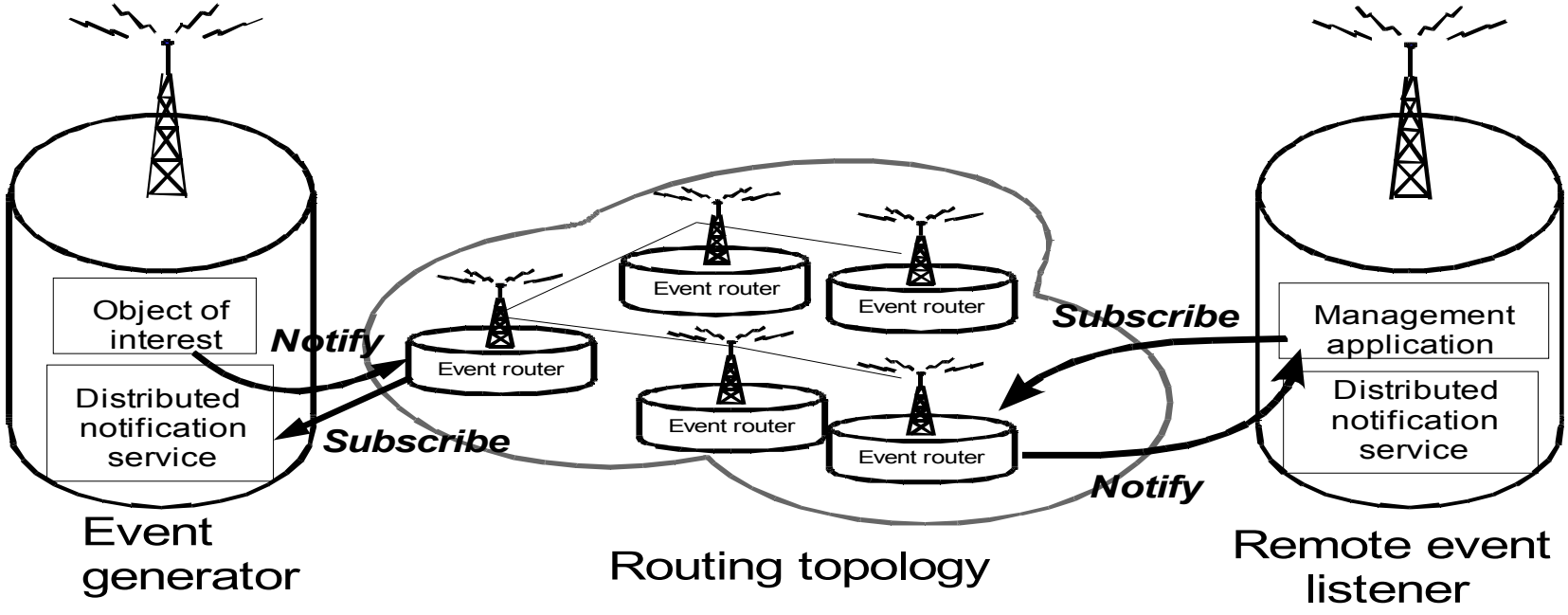
WS-Adressing Bilan



- SOAP spécifie la structure des messages pour invoquer des services
- WS-Adressing fournit un format d'adressage au niveau des messages SOAP
- WS-Adressing permet de décorréler SOAP du protocole de transport
- Comment définir une communication inter service Web plus flexibles, asynchrone ?

WS-Notification, WS-Eventing

- Notification d'événements
  - Se connecter à un service Web
  - Publier des événements
  - Souscrire aux événements



WS-Notification, WS-Eventing

- Notification et souscription sont deux messages envoyés dans un seul sens
- Deux approches pour gérer des notifications
  - Approche centralisée (broker d'événements possiblement fédérés)
  - Approche décentralisée
- Filtrage des événements, agrégation sont aussi gérés au niveau des spécifications

Exemple de souscription

```
<s12:Envelope  xmlns:s12="http://.../soap-envelope"
    xmlns:wsa="http://.../addressing" xmlns:wse="http://.../eventing"
     xmlns:ew="http://.../warnings" >
     <s12:Header>
     <wsa:Action>       http://.../eventing/Subscribe </wsa:Action>
     <wsa:MessageID> uuid:d7c574259  </wsa:MessageID>
     <wsa:ReplyTo>
    <wsa:Address>http://..../MyEventSink</wsa:Address>
     </wsa:ReplyTo>
   <wsa:To>http://www..../oceanwatch/EventSource</wsa:To>
  </s12:Header>
```

@ Exemple donné dans la spécification W3C

Suite souscription (où notifier?, identification de la souscription)

```
<s12:Body>
   <wse:Subscribe>
     <wse:Delivery>
      <wse:NotifyTo>
         <wsa:Address> http://.../MyEventSink/OnStormWarning
    </wsa:Address>
      <wsa:ReferenceProperties>
         <ew:MySubscription>2597</ew:MySubscription>
       </wsa:ReferenceProperties>
       </wse:NotifyTo>
    </wse:Delivery>
   </wse:Subscribe>
 </s12:Body> </s12:Envelope>
```

@ Exemple donné dans la spécification W3C

Notification (exemple du w3C portant sur la météo : le vent)

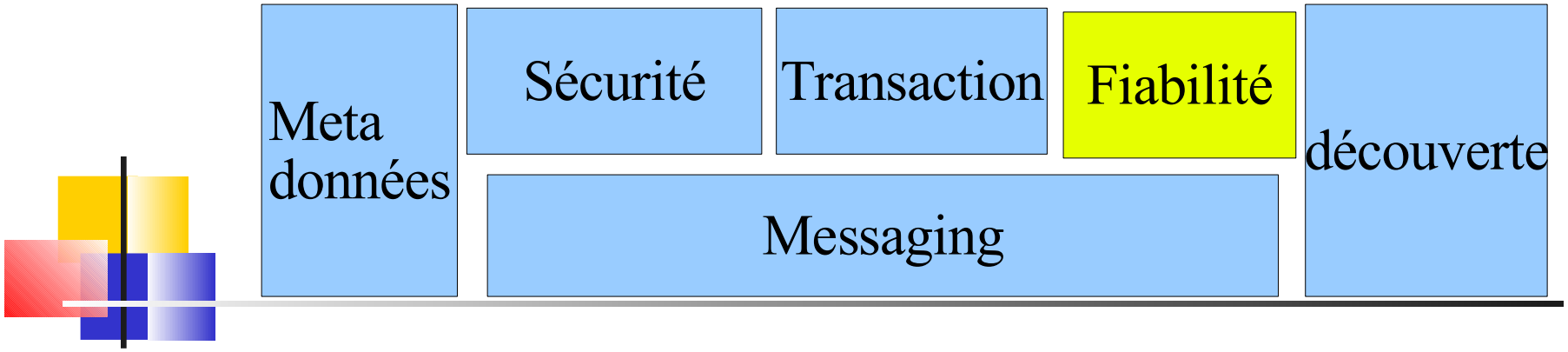
```
<s12:Envelope      xmlns:s12="http://.../soap-envelope"
   xmlns:wsa="http://.../addressing" xmlns:ew="http://.../warnings"
   xmlns:ow="http:/.../oceanwatch" >
  <s12:Header>
  <wsa:Action> http://../oceanwatch/WindReport</wsa:Action>
     <wsa:MessageID> uuid:568b4ff2-5fd8d7f </wsa:MessageID>
      <wsa:To>http://www..../OnStormWarning</wsa:To>
     <ew:MySubscription>2597</ew:MySubscription>
     <ow:EventTopics>weather.report </ow:EventTopics>
   </s12:Header>
```

@ Exemple donné dans la spécification W3C

Suite notification

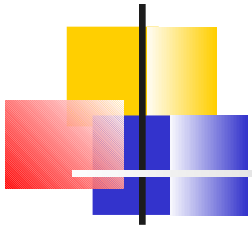
```
<s12:Body>
   <ow:WindReport>
     <ow:Date>030701</ow:Date>
   <ow:Speed>65</ow:Speed>
    <ow:Location>BRADENTON BEACH</ow:Location>
     <ow:County>MANATEE</ow:County>
     <ow:State>FL</ow:State>
     <ow:Lat>2746</ow:Lat>
     <ow:Long>8270</ow:Long>
           ...
   </ow:WindReport>
 </s12:Body>
</s12:Envelope>
```

@ Exemple donné dans la spécification W3C



- WS-ReliableMessaging : délivrer des messages de façon fiable malgré des erreurs (réseau, logiciel)
  - Ordonnancement de messages (SOAP) : chaque séquence de messages identifiée, un numéro pour chaque message
  - Perte des messages (SOAP) : s'assurer qu'un message envoyé est effectivement reçu via des accusés de réception positifs et négatifs

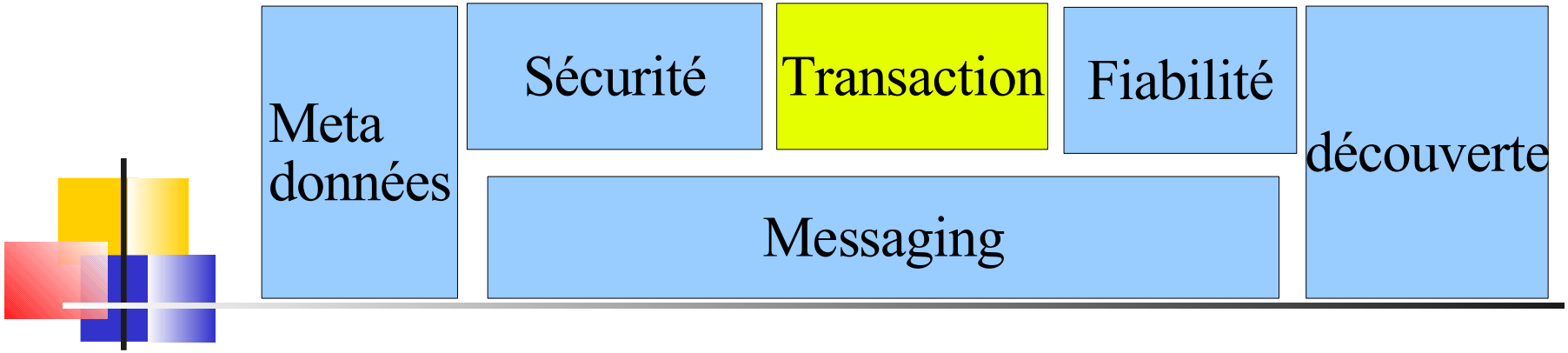
WS-ReliableMessaging



- Oui, mais pourquoi de la fiabilité?
  - SOAP peut utiliser UDP comme transport
  - TCP fournie des garantie pour 1 saut (et donc pas pour du multi-sauts)
  - Les applications (notamment les transactions) ont des exigences très contraignantes

WS-Reliable Messaging – Guaranties

- WS-Reliable Messaging supporte plusieurs formes de garanties
  - At-Least-Once: garantie le délivrement
  - At-Most-Once: Garantie sur l'élimination des messages dupliqués
  - Exactly-Once: Garantie à la fois le délivrement et l'élimination des duplicats
- Bilan : ces garantis sont des pré-requis pour un support des transactions et des processus métiers

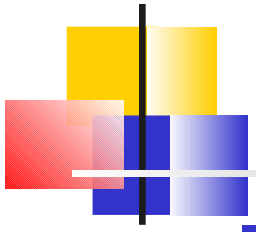


- Besoin : support des transactions et des processus métiers
  - Transactions
    - WS-AtomicTransaction
    - WS-BusinessActivity
  - Les processus métiers (avec composition et l'intégration des services Web)
    - WSCL, WS-choreography, WS-orchestration, BPEL4WS (Business Execution Language for Web Services)

Transaction traditionnelles

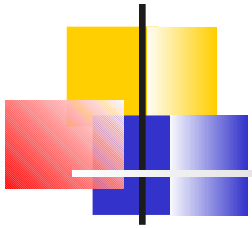
- Propriétés traditionnelles des transactions
  - Temps de vie limitée, couplage fort
  - ACID
    - Atomicity : le tout ou rien, si une partie de la transaction est un échec, alors toute la transaction est un échec
    - Consistency : seules des données valides sont sauvegardées. En cas d'invalidité, la transaction sera retrogradées de façon à maintenir un état cohérent
    - Isolation : les transactions opèrent de façon isolée
    - Durability

Transactions sur le Web

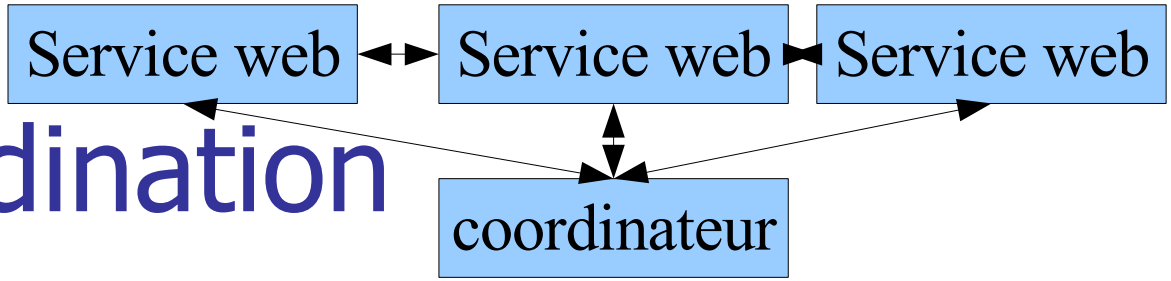


- Propriétés des transactions pour les Web services
  - Transactions distribuées
  - Long terme
  - Les transactions ACID sont trop contraignantes : de longues transactions peuvent réduire des accès concurrents
  - Il faut supporter des compensations (scénario de l'agence de voyage) et non pas verrouiller les ressources

Transaction (2/2)



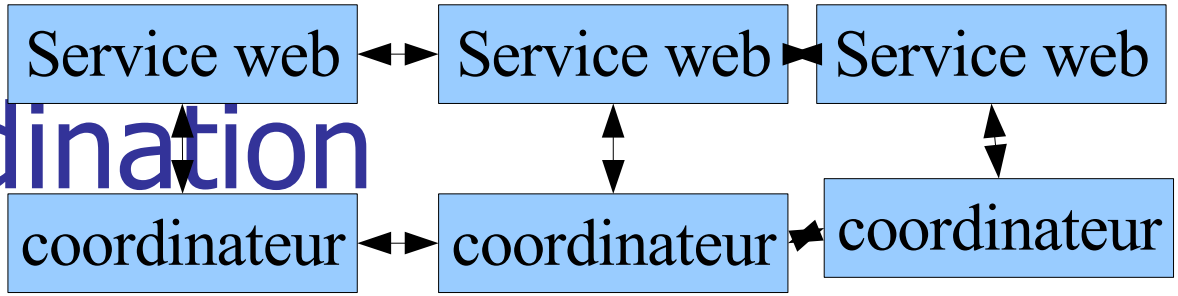
- Solution : 2 spécifications distinctes
  - WS-Atomic transaction (court terme)
  - WS-Business Activity (long terme, tolére la défaillance des transactions, compensables)
- Une spécification supplémentaire, WS-Coordination pour coordonner les transactions distribuées



WS-coordination

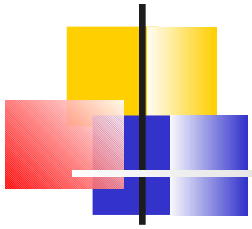
- Contexte :
  - De nombreux services Web distribués
  - Des échanges complexes entre les participants
- Besoin :
  - coordonner les activités (**transactions**) & les participants
- Solution : définition d'un coordinateur + des protocoles de coordination

WS-coordination



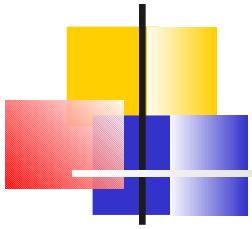
- Est une spécification d'une infrastructure de coordination
- Une application souhaite créer un service transactionnel ou non
- Un service d'activation permet à l'application de créer
  - Un **coordinateur** (instance)
  - Un **contexte de coordination** pour une activité; ce contexte est placé dans les messages

WS-coordination : Model



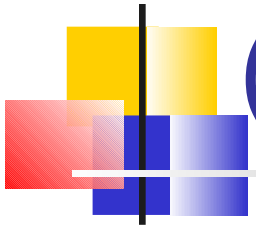
- **Un service d'enregistrement** permet d'enregistrer les protocoles auprès de l'infrastructure de coordination
  - Quels sont les protocoles de coordinations?
  - A qui envoyer des notifications lorsque le protocole passe à une autre étape?
- WS-coordination ne vise pas à définir un nouveau protocole de coordination

Transactions distribuées (1/2)



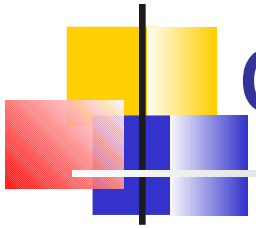
- Un protocole de verrouillage à deux phases assure une terminaison atomique de la transaction
- Un protocole basé sur le consensus pour prendre la décision assez longue
- Etape 1 : préparation
  - Le Coordinateur s'assure que les participants sont prêts (message à chaque participant)

Transactions ACID distribuées (2/2)



- Étape 2 : vote
  - Participants: votent pour abandonner (abort) ou terminer (commit)
- Etape 3 : prise de décision du coordinateur
  - Le coordinateur notifie les participants de la décision (terminaison, abandon)
- Etape 4 : acquittement des participants

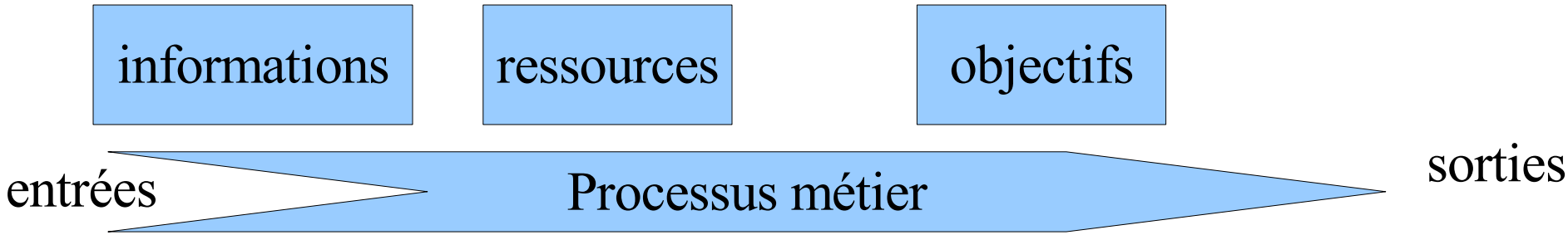
WS-Transaction et WS-coordination



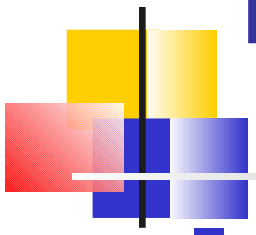
- Fournit un support
  - pour gérer
    - des transactions long-terme/short-terme
    - Des transactions Web distribuées compensables
  - Coordonner des activités transactionnelles
    - Une infrastructure de coordination
    - Des abstractions nécessaires (protocoles de coordination, type de coordination, contexte de coordination)
- Comment gérer des processus complexes?

Processus métiers

- Un processus métier :
  - une collection d'activités conçues pour un métier (utilisateur et marché) donné
  - Implique une emphase sur le métier effectué dans l'organisation
- Processus : collection d'activités ordonnées (notion de workflow)

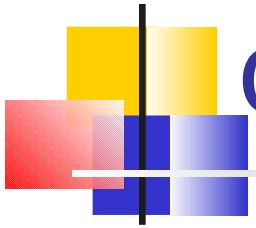


Processus Web



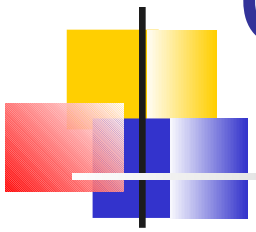
- Un processus métier peut être vu comme un ensemble d'interactions entre des services Web (workflow)
- Besoin : définir de façon abstraite et concrète un processus
  - Décrire comment des services Web peuvent s'inter-connecter, s'intégrer, se composer
  - ➔ Langage de chorégraphie (conversation), d'orchestration

Chorégraphie versus orchestration



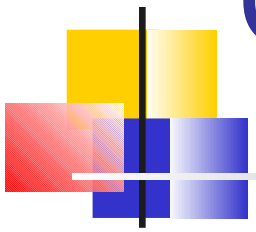
- Chorégraphie :
  - Un processus est un échange de messages ordonnés comme c'est le cas lors d'une **conversation**
  - 2 langages : WSCL, WS-choreography
- Orchestration : Un processus est
  - une suite d'opérations ordonnées (**workflow**)
  - sous le contrôle d'un coordinateur central (chef d'orchestre correspondant à un partenaire)

Conversation (1/2)



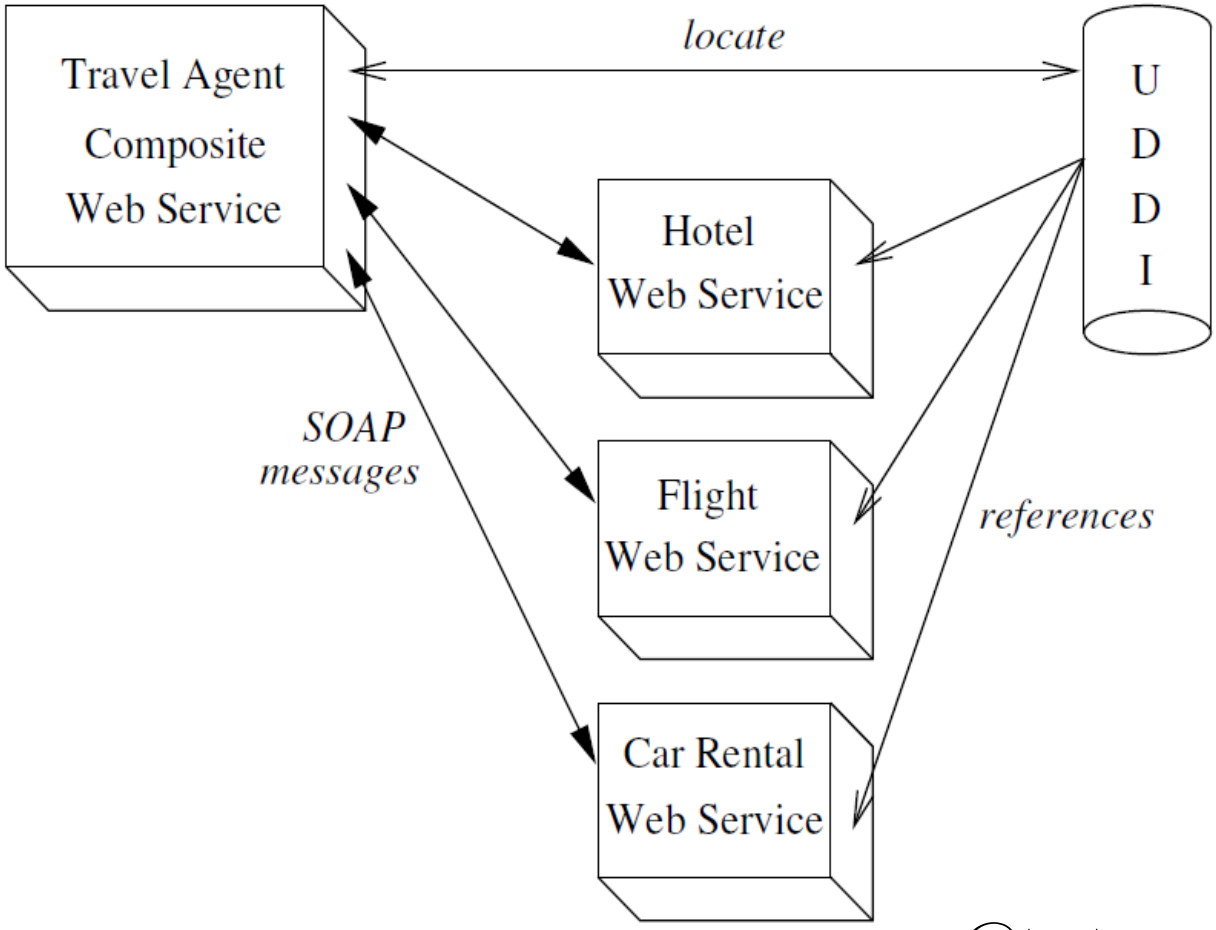
- Limitations de WSDL :
  - WSDL ne définie qu'une liste d'opérations
  - WSDL ne définit pas dans quels ordres les opérations doivent être invoquées
- Par exemple, il faut déjà se logger à un ordinateur avant d'accéder à un fichier, et non le contraire

Conversation (2/2)



- Fournit le vocabulaire nécessaire pour définir l'ensemble de tous les ordres correctes, les contraintes d'ordre, sous quelles conditions la conversation prend fin
  - Logique temporelle, chorégraphie, échange de messages, point d'accès
  - Les processus privés (que se passe t-il à l'intérieur du service) et publiques (que voit-on de l'extérieur). Remarque : un processus publique doit être en accord avec les processus privé

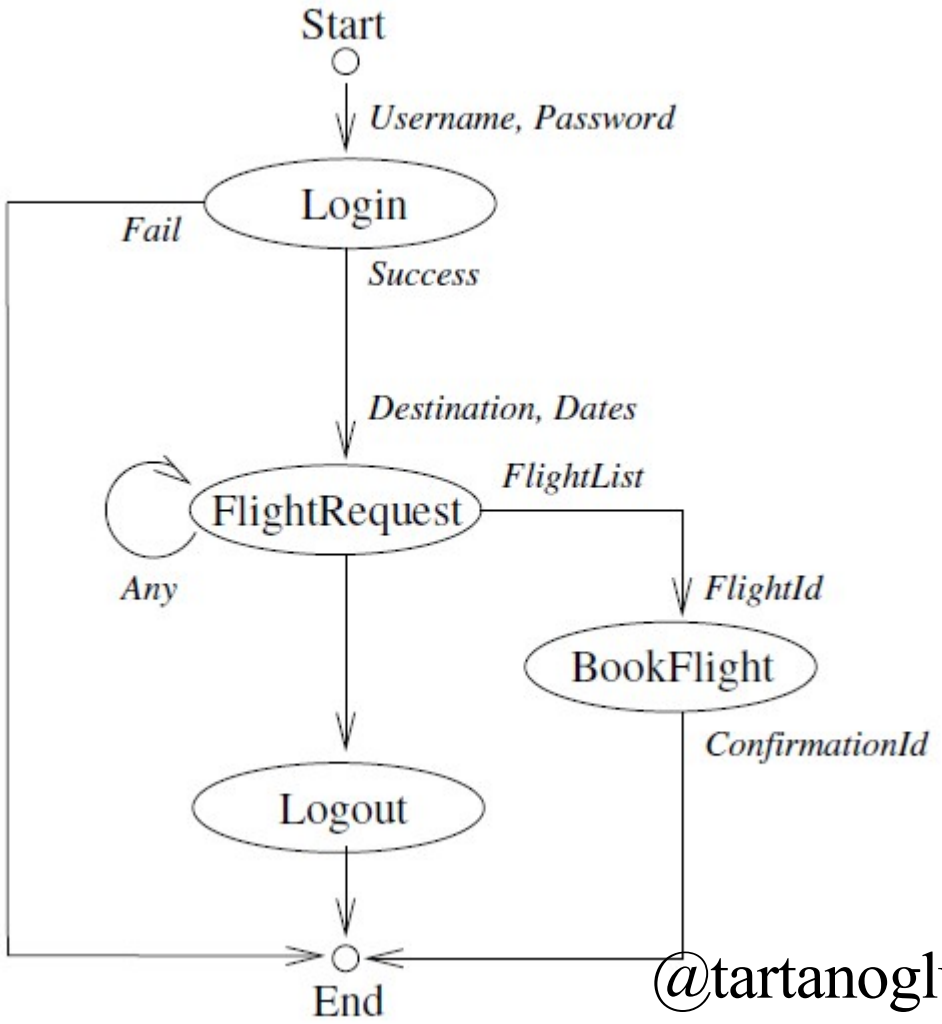
Exemple : l'agence de voyage



@tartanoglu

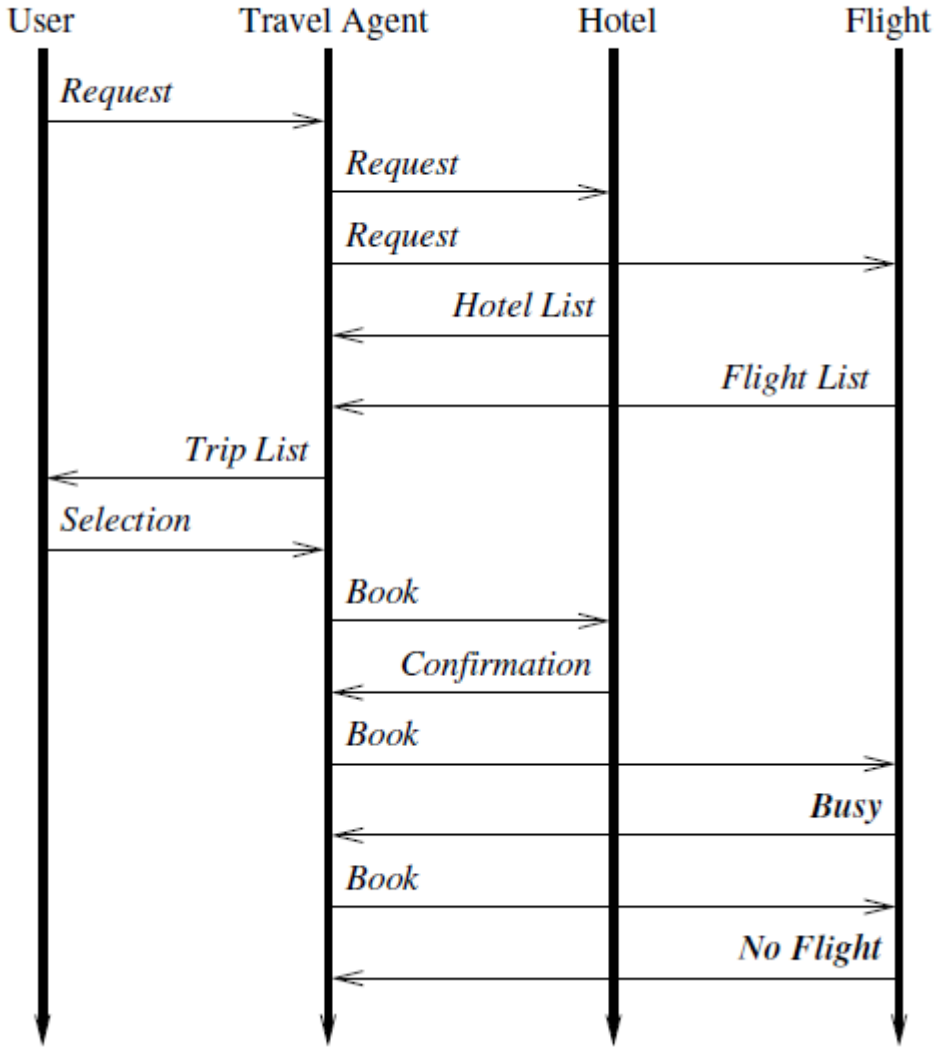
Représentation graphique d'une conversation

- Finite state machine :
  - Modéliser les états suivis
  - Transition : invocation d'une opération



Représentation graphique d'une conversation

Diagramme de séquence

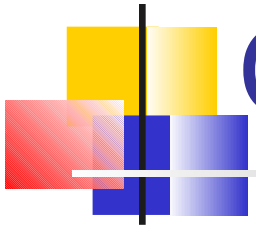


WSCL : Web Service Conversation Language

```
<Conversation name="TravelAgency"
        initialInteraction="Start" finalInteraction="End" >
 <ConversationInteractions>
  <Interaction interactionType="SendReceive" id="Payment">
  <OutboundXMLDocument id="Invoice" hrefSchema="http://.../invoice.xsd"/>
  <InboundXMLDocument id="Payment" hrefSchema="http://.../Payment.xsd"/>
  </Interaction>
<ConversationTransitions>
      <Transition>
        <SourceInteraction href="Quote"/>
        <DestinationInteraction href="Purchase"/>
      </Transition>
      ...
```

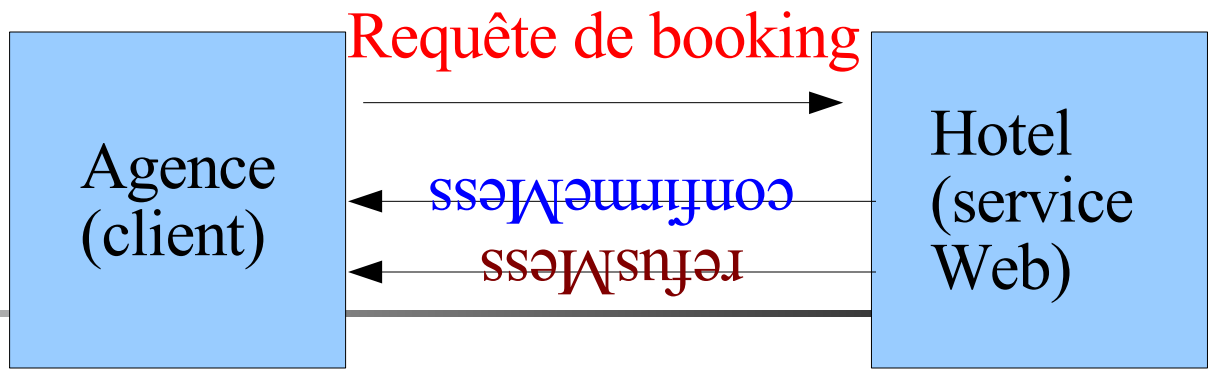
- Conversation = liste d'interactions et de transitions
- Interactions : Send, Receive, SendReceive, Empty
- Transition : ordre des interactions

WSCI : Web service Choreography Interface



- Une chorégraphie
  - Décrit le flot du **point de vue de chaque service**
  - **est insérée dans le document WSDL de chaque service**
- L'élément `<interface>`définie l'interface publique (visible depuis l'extérieur)
- L'élément `<model>` pour combiner plusieurs interfaces et de les lier entre elles

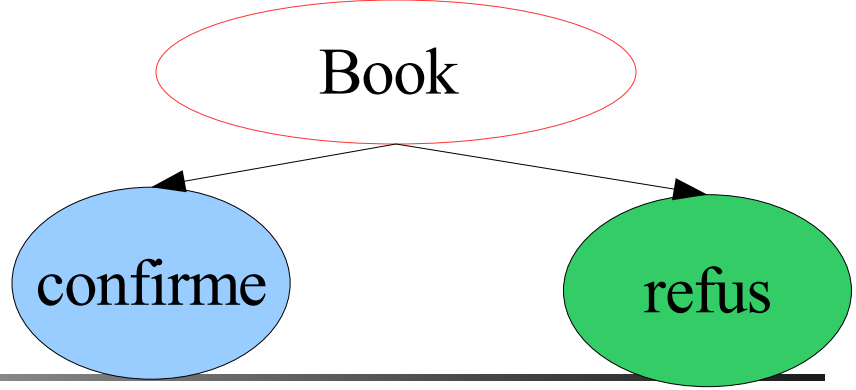
Exemple



- WSDL du service de booking de l'hotel

```
<wsdl:portType name="TAtoHotel">
  <wsdl:operation name="Book">
    <wsdl:input  message="bookingReq" />
  </wsdl:operation>
  <wsdl:operation name="Confirme">
    <wsdl:output message="ConfirmeMess" />
  </wsdl:operation>
 <wsdl:operation name="Refus">
    <wsdl:output message="refusMess" />
  </wsdl:operation>
</wsdl:portType>
```

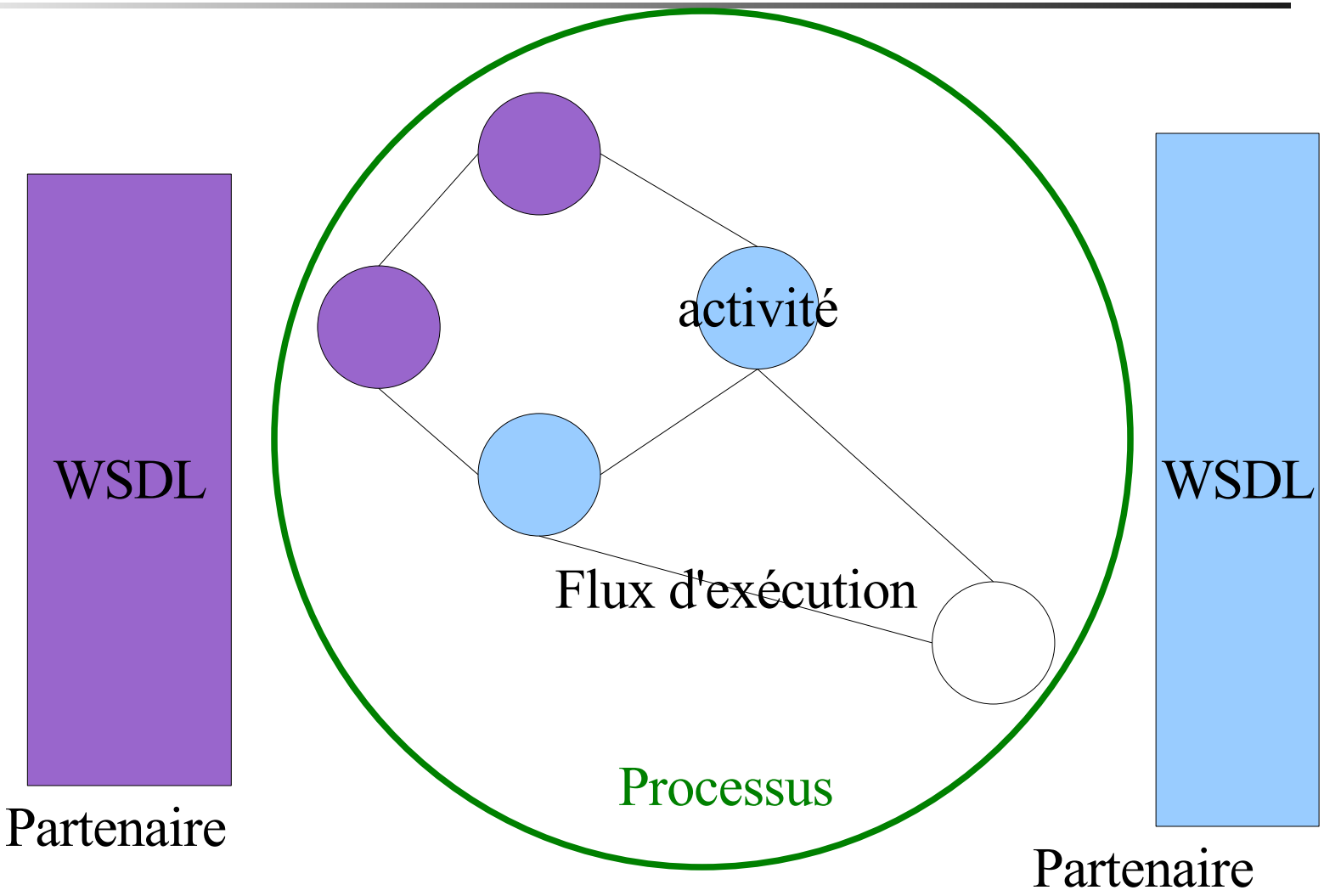
Suite…



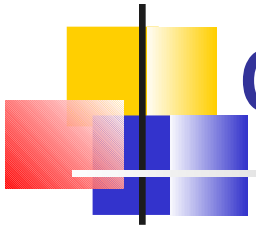
- Interface du service de booking définissant la choréographie

```
<interface name="BookingService">
  <process name="BookTrip" instantiation="message">
  <sequence>
    <action name="ReceiveBooking" operation= "TAtoHotel/Book"/>
    <switch>
    <case>
    <condition>placesAvailable</condition>
   <action name="SendConfirme"operation="TAtoHotel/Confirme"/>
    </case>
    <default><action name="SendRefus" operation =
"TAtoHotel/Refus"/> </default>  </switch></sequence>
  </process></interface>
```

Orchestration et choréographie (BPEL4WS)

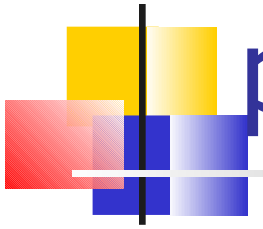


Orchestration et choreographie (BPEL4WS)

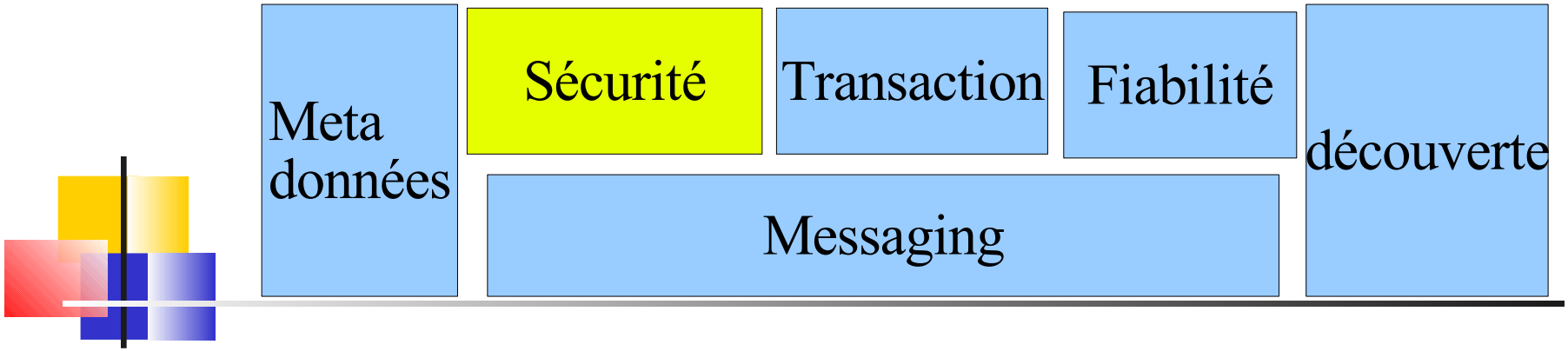


- Les messages échangés entre les activités sont des messages SOAP
- WSDL décrit un ensemble d'activités
- BPEL4WS décrit
  - un partenaire (document XML)
  - Une activité
  - Flux (séquences, switch, ...)
  - Synchronisation, concurrence

Bilan – Transaction et processus

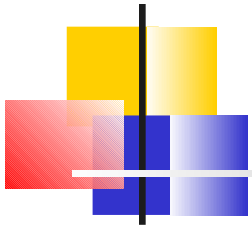


- Ensemble de spécification pour la gestion automatique des transactions et des processus métiers dans les services Web
  - Transactions atomiques et Web
  - Coordination des transaction mais aussi des activités
  - Framework : Microsoft, IBM, BEA



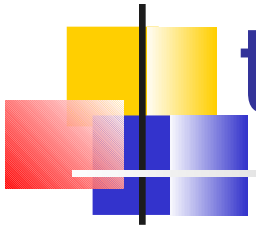
- Spécifications liées à la sécurisation
  - Sécurité des échanges : WS-security
  - Confiance : SAML, WS-Trust, WS-SecureConversation, WSFederation
  - Infrastructures de clés publiques XML : XKMS
- Autres spécifications liées (méta données avec la famille WS-Policy-*) , et la fiabilité

WS et sécurité



- Deux besoins identifiés :
  - Sécuriser les interactions (messages)
    - Transport (HTTP notamment)
    - Message (SOAP)
  - ➔ WS-Security (2004) correspond à la brique de base pour la sécurisation des échanges
  - Sécuriser les services Web (les processus) -> politiques de sécurité, risque, confiance/trust

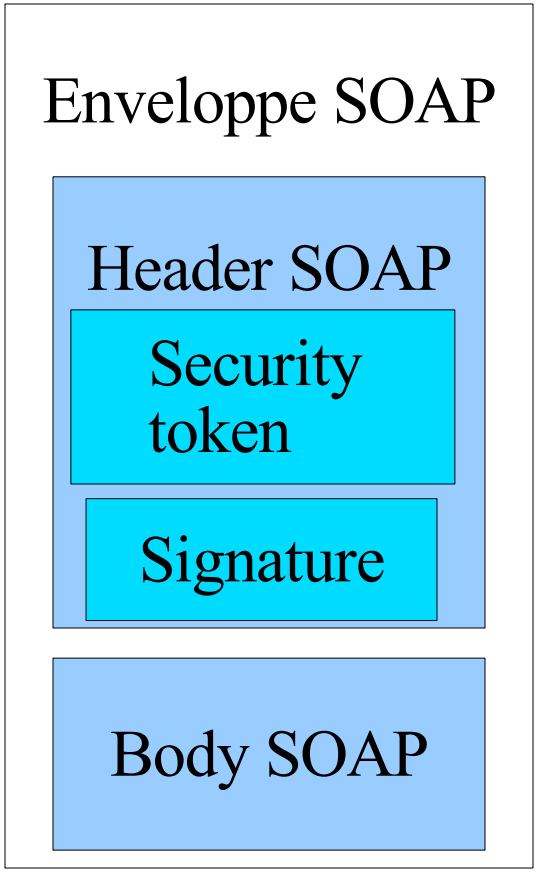
WS-Security - Sécurité et transport HTTP



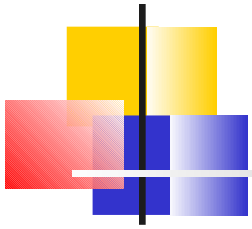
- Utilisation de HTTP et SSL
- Confidentialité
  - Connections chiffrées SSL point à point
- Authentification
  - Schémas d'authentification de HTTP
  - Certificats SSL
- Autorisation : implémentation de politiques au niveau des infrastructures

WS-Security - Interactions sécurisées avec SOAP

- Intégrité : adjonction d'une ou plusieurs signatures XML
- Confidentialité : chiffrement(**s**) XML, **s** si plusieurs acteur**s**,
- Authentification
  - Informations stockées dans un « securitytoken »
    - Par ex : nom, password

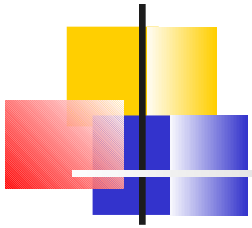


WS-security



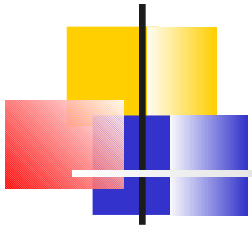
- Chiffrement est effectué de bout en bout (client-service Web)
- Intégrité est fournie dans un environnement multi-sauts
- Si les contraintes de sécurité ne sont pas satisfaites, une faute (SOAP) sera envoyée
  - Exemple de faute : Security Tocken Invalid, Decryption failure

WS-Policy



- Fournir les conditions suivant lesquelles un service doit être fourni
- Une politique = une ou plusieurs assertions représentant
  - Les capacités du service Web
  - Les exigences du Service Web
    - Ex : demander à ce qu'une requête soit chiffrée
- Transmettre au service ce qu'il attend

WS-SecurityPolicy

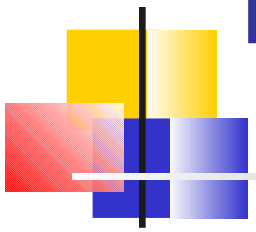


- Définit comment les messages sont sécurisés, fiabilisés sur un chemin de communication
  - Comment dois -je sécuriser l'acheminement d'un message (quel chiffrement, quel protocol, ...)
- Est attaché à un document WSDL

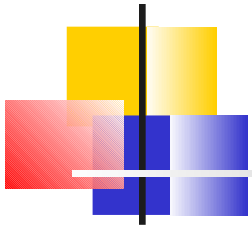
WS-SecurityPolicy- Exemple

```
<sp:TransportBinding>
  <Policy>
    <sp:TransportToken>
      <Policy> <sp:HttpsToken RequireClientCertificate="false" />
      </Policy>
    </sp:TransportToken>
    <sp:AlgorithmSuite>
      <Policy>  <sp:Basic256Rsa15 /></Policy>
    </sp:AlgorithmSuite>
    ...
  </Policy>
</sp:TransportBinding>
```

Bilan - WS-security



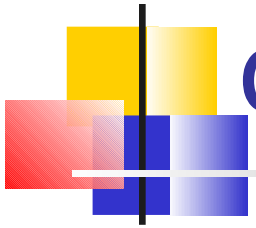
- En se basant sur des politiques de sécurité (WS-SecurityPolicy), WS-security fournit une sécurisation des échanges au niveau
  - Du transport
  - Des messages SOAP
- Au niveau applicatif : le service Web gère lui-même son schémas de sécurité



WS-Trust

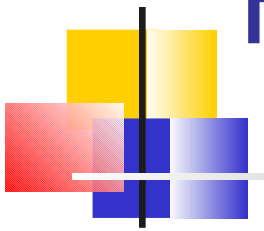
- Objectifs :
  - Établir la confiance avant l'échange de messages SOAP-WS-Security
  - Permettre à un service (Security Tocken Service) de demander des infos à un tier de confiance
- Nécessités :
  - Assurer la gestion (création, update, destruction) et l'échange de security-tockens
  - Assurer découverte & fédération de cercles de confiances

Fédération de cercles de confiance



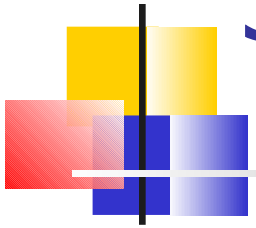
- Une fédération correspond à un ensemble de domaines ayant établis une relation de confiance
- Une fédération opère indépendamment des frontières administrative d'un réseau
- Exemple de fédération : un utilisateur de carte bleu retirant de l'argent depuis une banque amie (confiance entre les banques)

Fédération - Approche



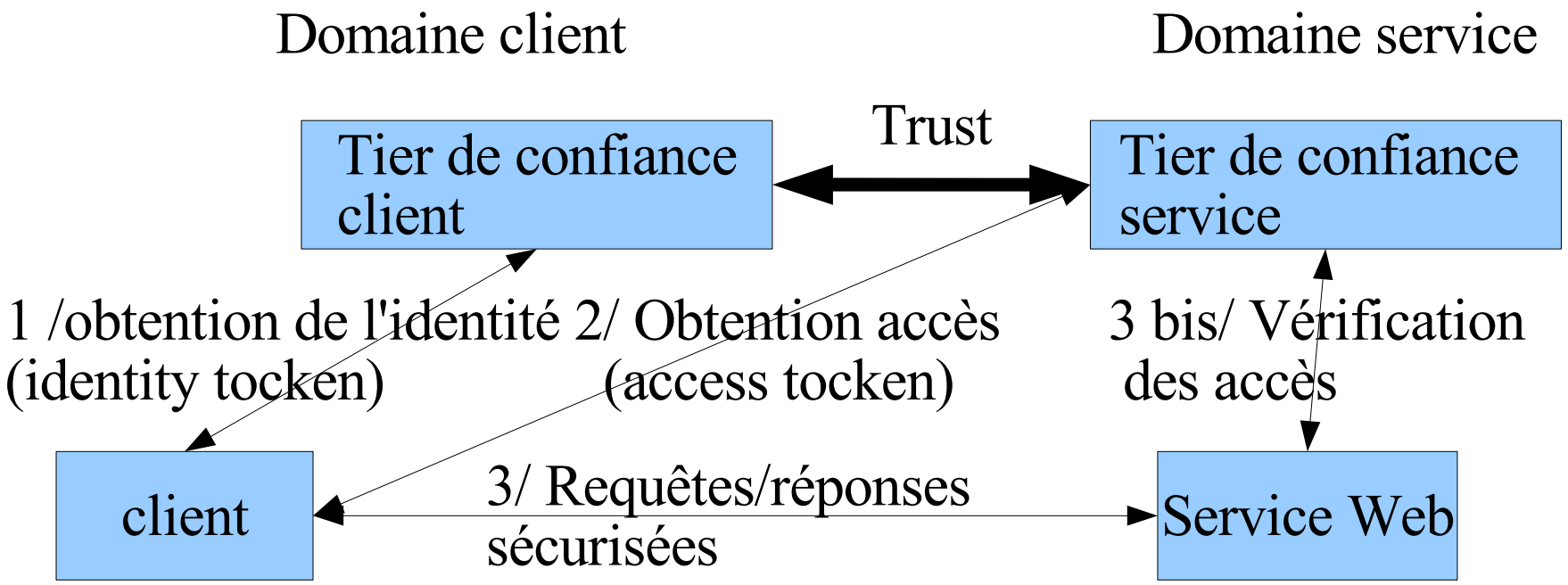
- Fournit un accès à un domaine de confiance en utilisant les identités des domaines
- Nécessite : établir la relation de confiance en
  - Partageant les informations relatives à l'identité, l'authentification et l'autorisation
  - Attribuant des identités : fédérations **anonymes** (pseudonymes, privacy) mais authentifiées
  - Se basant sur un brocker de confiance

3 Topologies de confiances

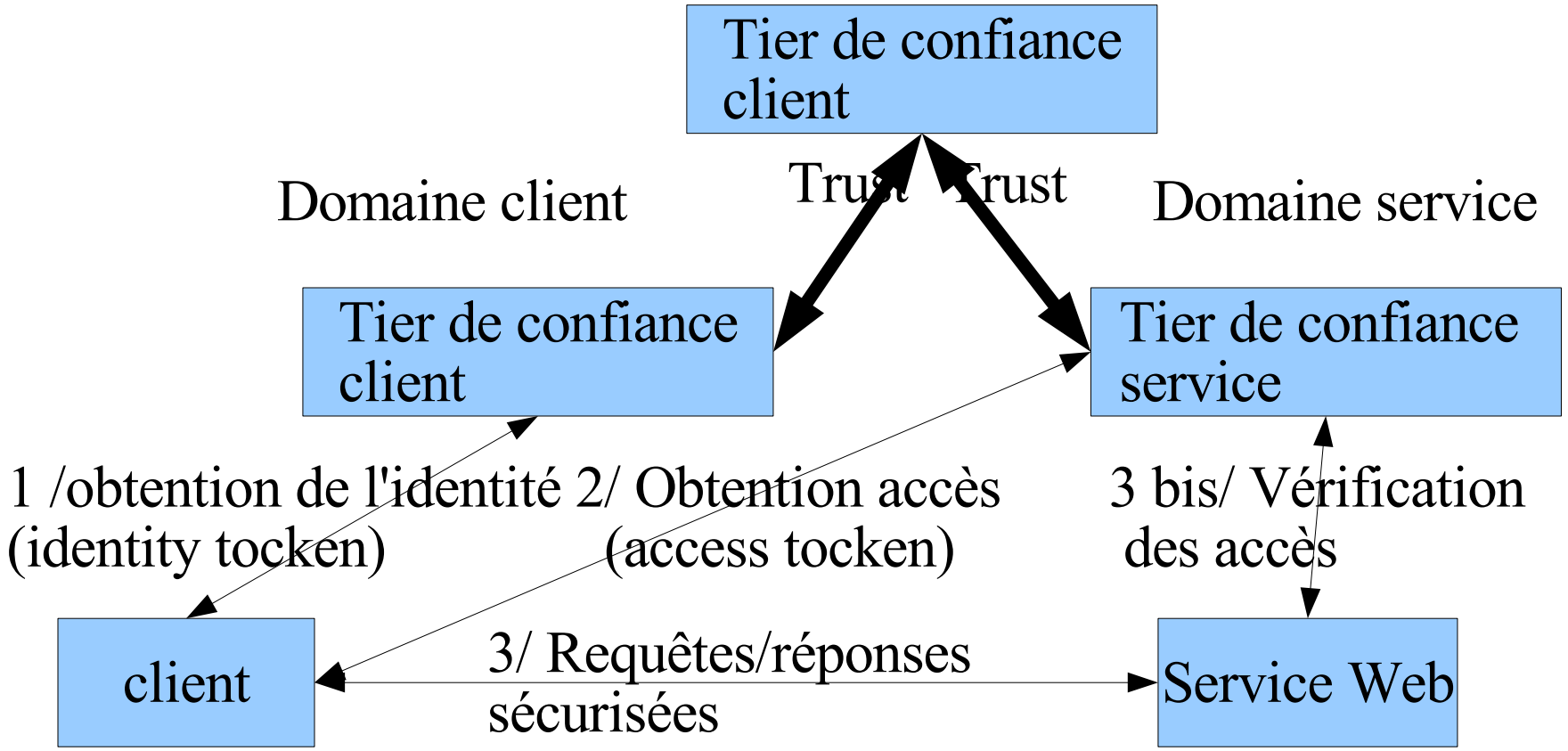


- Confiance directe : le tiers de confiance du client connaît le tiers de confiance du service Web
- Confiance indirecte : le tiers de confiance du client ne connaît pas directement le tiers de confiance du service Web
- Délégation de la confiance

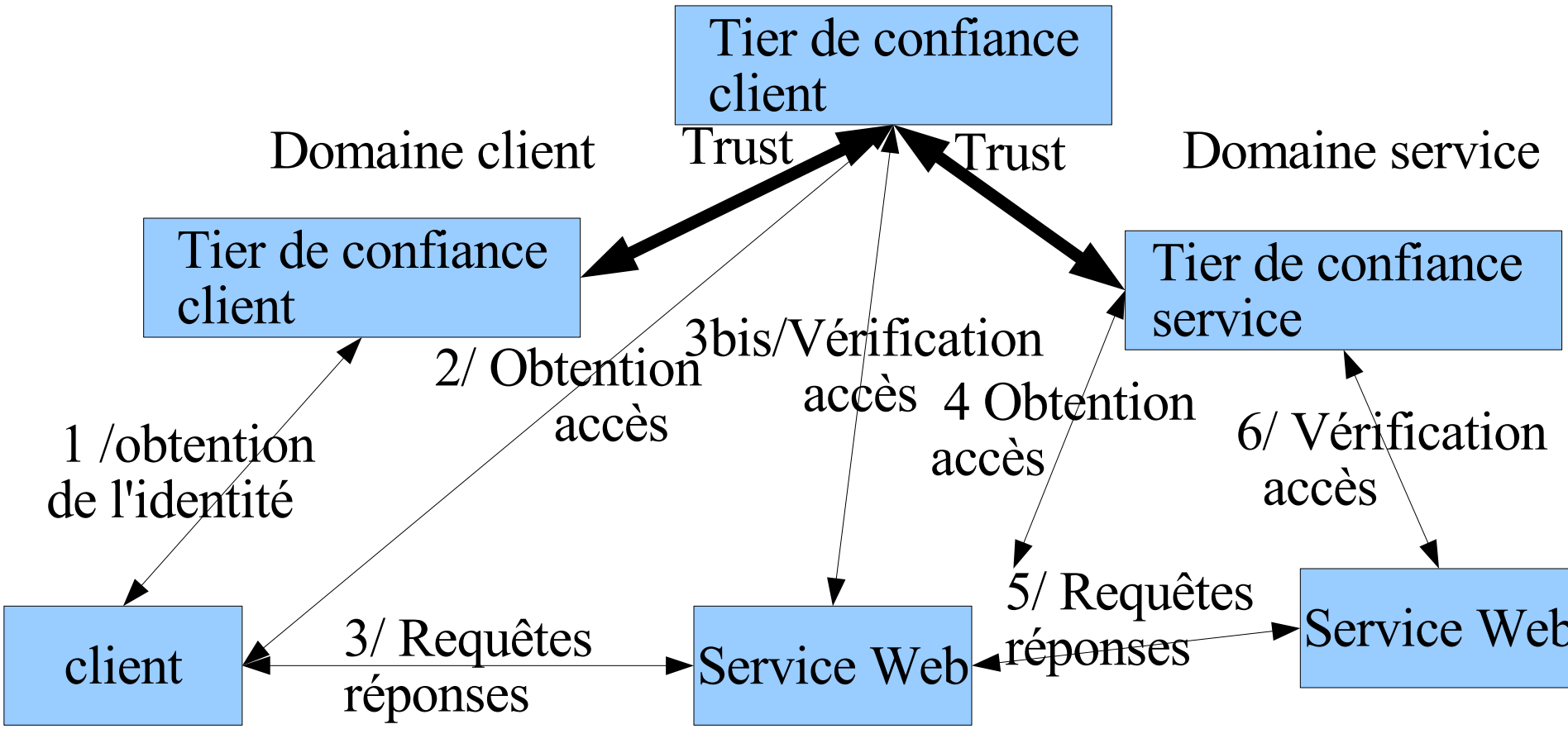
Topologie directe



Topologie indirecte (hiérarchie de confiance)

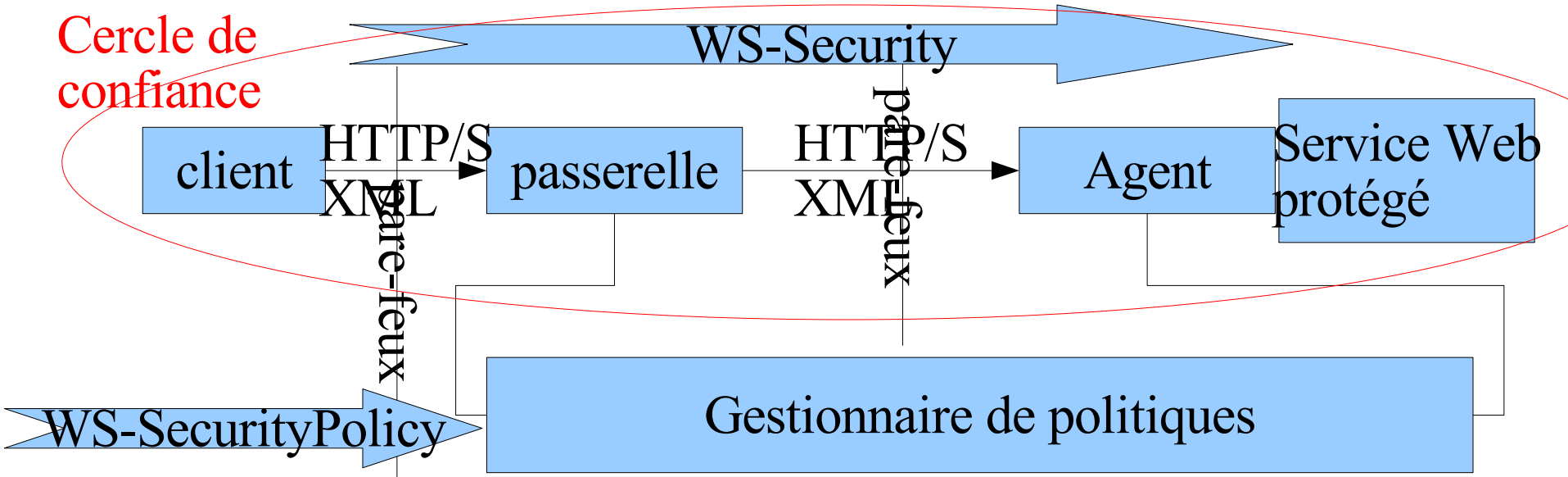


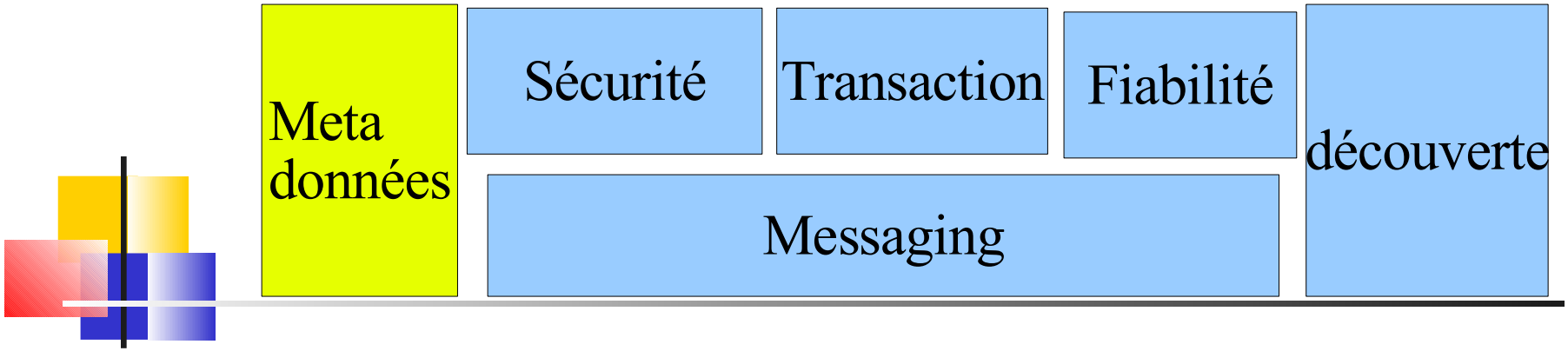
Topologie – délégation



Bilan - sécurité

- Deux paradigmes :
  - WS-Trust, FS-federation : gestion de la confiance
  - WS-security, WS-Policy : chiffrement, authentification, intégrité ect... suivant des politique





- Description des services
  - WSDL
- Politiques
  - WS-Policy, WS-PolicyAssertions, WS-PolicyAttachment, WS-MTOMPolicy
- WS-MetadataExchange, les ontologies
- Profile des terminaux : WS-I Basic Profile

Bilan – WS-*

- Les services Web n'ont pas pour objet de réinventer
  - de nouvelles plateformes de développement,
  - de nouveaux protocoles d'échanges,
  - de nouveau mécanismes (transaction, sécurisation, composition, chorégraphie, fiabilité..)
- Les technologies des services Web ont pour objectif que
  - de décrire des mécanismes existants (RPC ect...)
  - de faire interagir des services distribués sur l'Internet/Web

Bilan – Ouverture (2/2)

- Les technologies des services Web ne cessent de se développer
  - Ensemble croissant de spécifications WS-*
  - Support du W3C, consortium OASIS
- Toutefois, elles ne suffisent pas pour gérer les fonctionnalités suivantes :
  - Cycle de vie du service (déploiement, configuration, mise à jour automatique, gestion des versions ect...)
- D'ou « l'apparition d'initiatives complémentaires », comme ...

OSGI Open Service Gateway Initiative

- 1999 : début de l'histoire,
  - OSGI alliance , http://www.osgi.org
  - focus sur java et l'embarqué
- 2003 : support pour des dispositifs mobiles
- 2004 : adoption

Développement d'une application

- Question : comment construire des applications
  - modulaires
  - dynamiques
- En suivant une approche client/serveur
  - fournisseur de service
    - implémente un interface (implémente le service)
  - Utilisateur du service
    - recherche le service à partir de son interface
    - Comment lier le client au service ?
  - → Découplage entre client et service
  - → Exemple : les services Web

Comment construire une application client serveur

- En suivant une approche objet
  - Objet client et service
  - Classes, interfaces
- En suivant une approche modulaire
  - On améliore le concept d'objet
  - Modularité : garder le cœur de l'application petit
  - Fournir la faculté d'ajouter des fonctionnalités autours de ce cœur
- Quelle plateforme standard propose une telle construction ?

OSGI (1/2)

- Fournit un environnement pour développer
  - des services
  - des composants
- fournit les élément nécessaires pour
  - décomposer/packager une application en services
  - pour s'abstraire en fournissant des composants
- Se base sur une plateforme Java
  - OSGI = système de module dynamique pour java

OSGI (2/2)

- Fournit tout le support nécessaire au cycle de vie d'une application
  - Fournit les éléments nécessaires à la maintenance
- Est un intergiciel (en anglais middleware)
  - Peut être utilisé sur différentes plate formes
    - Limiter le problème de la portabilité
    - Gérer l'application sur le dispositif
  - Pour une distribution large échelle
    - Supporter une myriad de versions différents, d'adaptations nécessaires

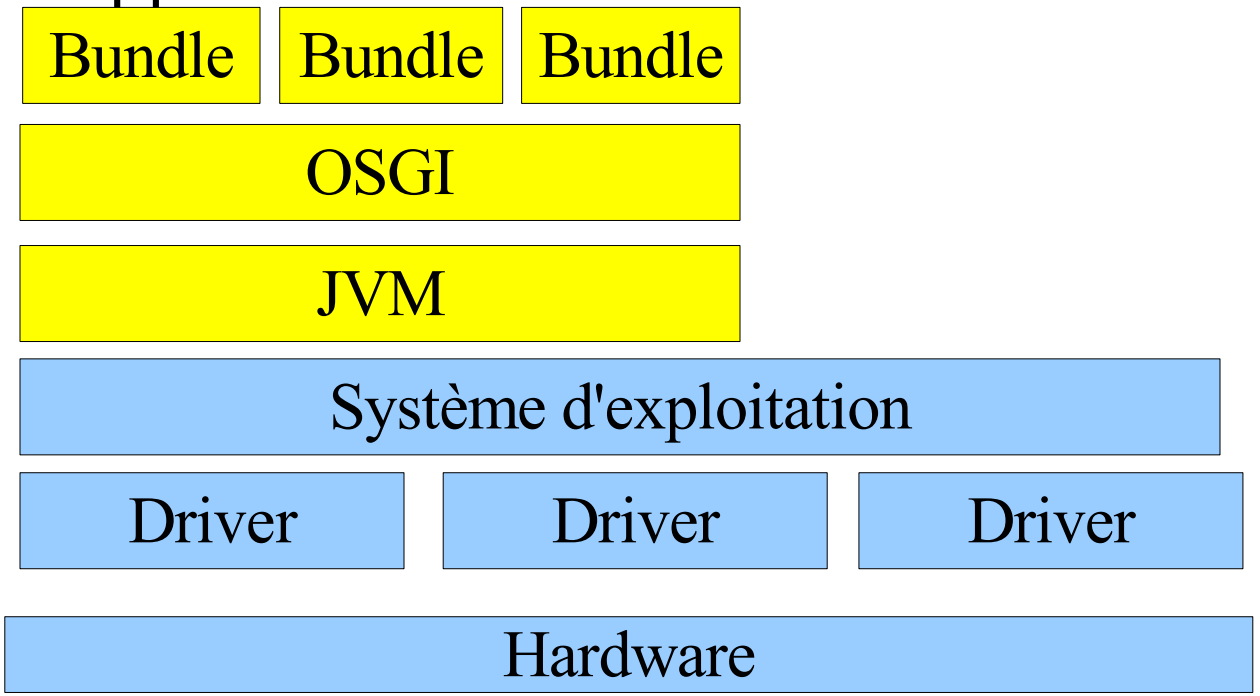
Intergiciel OSGI

JVM : Java Virtual Machine : permet à une application java de s'exécuter

Deux couches au dessus de java :

OSGI : permet à un bundle de s'exécuter (se charger)

bundle : application



Bénéfices résultants du recours à un intergiciel

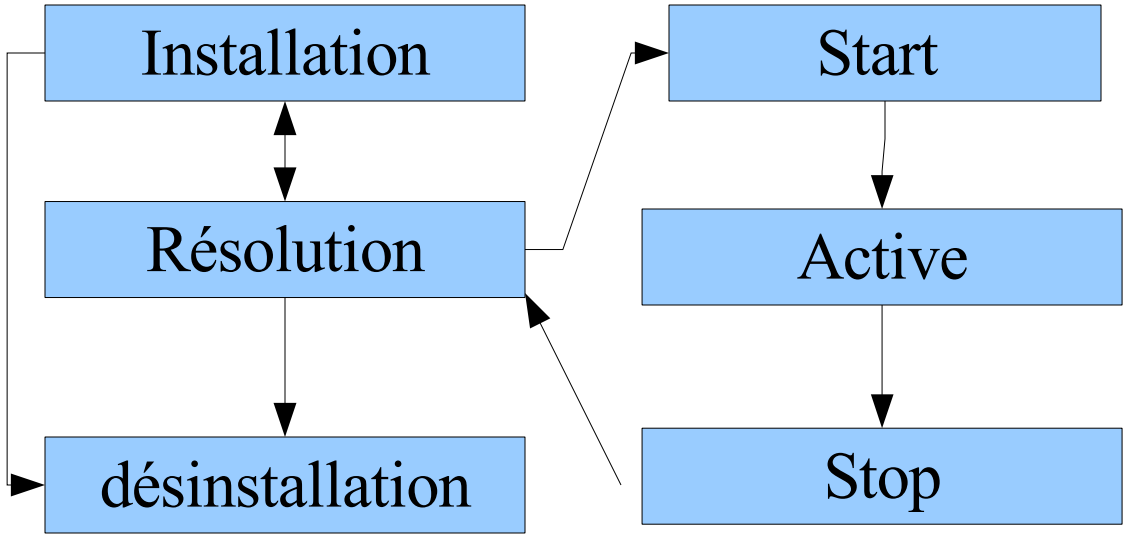
- OSGI est une plate forme
  - pour **construire** des applications java à base de composants : bundles
  - Pour **maintenir ces applications durant tout leur cycle de vie**
- Une application peut s'exécuter sur différents types de
  - matériels (hardwares)
  - logiciels

Bundle

- **Un bundle est**
  - **un module, c'est-à-dire l'unité de base pour déployer et modulariser**
- **En pratique : un bundle contient**
  - Du code (ensemble de classes dans un package)
  - Des méta données (contexte, description du bundle)
  - Des ressources (d'autre fichier jar)

Cycle de vie d'un bundle

- Cycle de vie :
  - Installation
  - Mise à jour
  - Exécution
  - Arrêt
  - Désinstallation



- OSGI fournit un support au cycle de vie d'une application
- Comment de pas stopper l'exécution de l'ensemble de l'application durant le cycle de vie service/de l'application ?

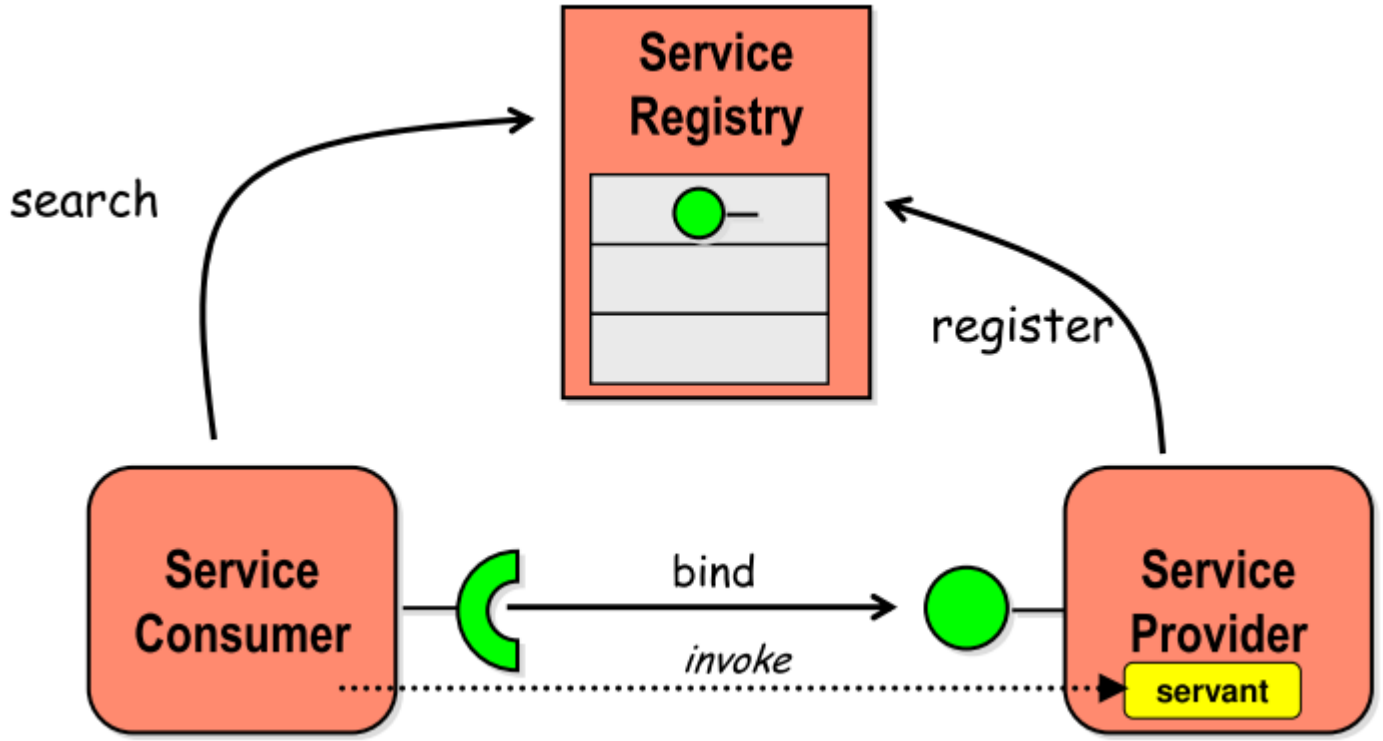
Le cycle de vie et la dynamicité mis en pratique

- Framework OSGI offre le support pour
  - Installer le code et les ressources
    - Pour cela, il lie le manifeste
  - Résoudre les dépendances (lier)
  - Contrôler le cycle de vie, les versions
- En pratique :
  - Appeler l'activateur du bundle pour démarrer le bundle
  - Gérer les class paths
  - Appeler de désactivateur de bundle
  - Nettoyer après la désactivation du bundle

Bundle et SOA

Un bundle peut aussi publier des services ...

Annuaire de services (publication & langage de demande)

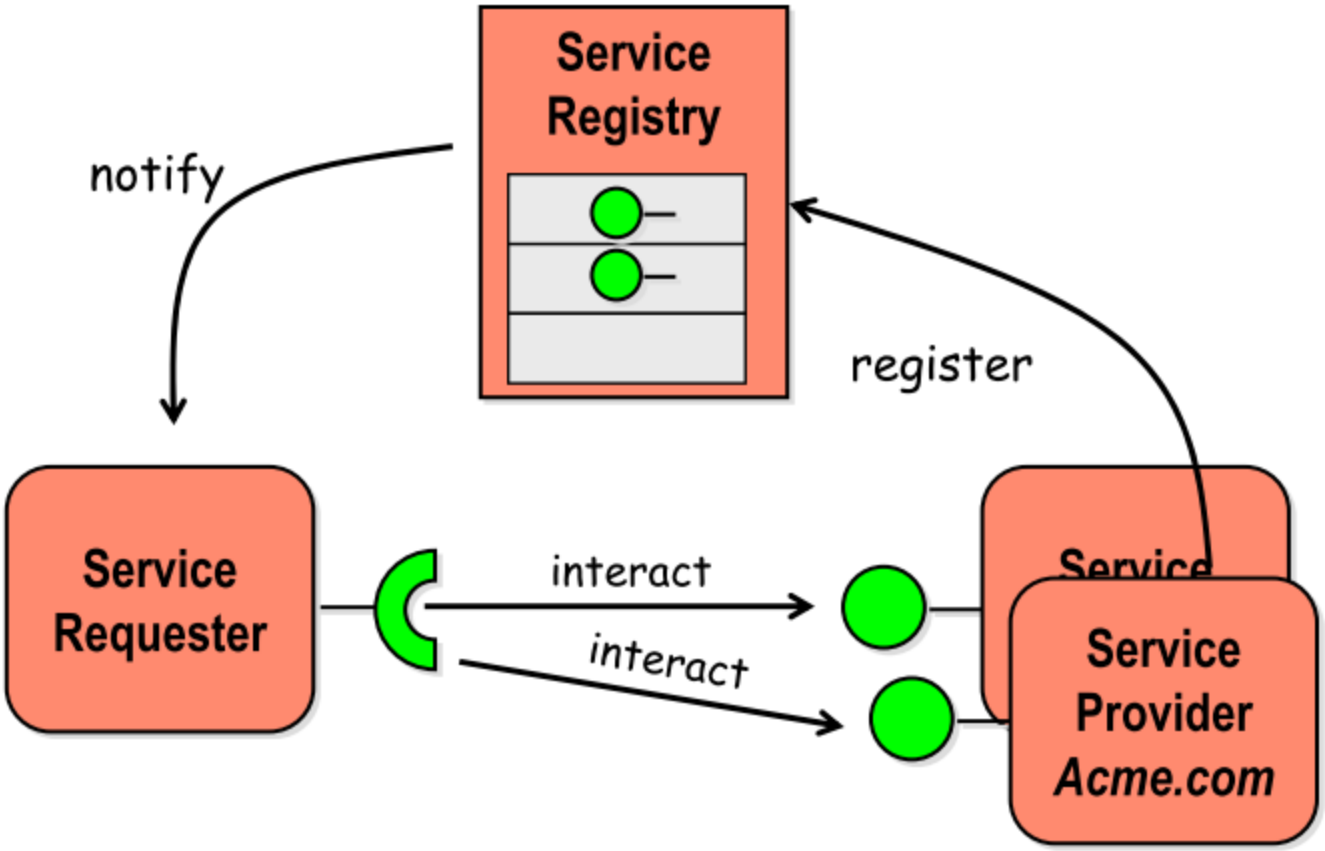


@Eric-Gressier-Soudan

SOA et approche à base de composants

- Faciliter l'assemblage de composants
  - Composant : unité de base défini à des fins de compositions et de réutilisation
  - Connecteur : composant de communication regroupe l'ensemble des mécanismes de communication distantes permettant d'interagir entre les composants
- Tout comme un service, un composant est une boite noire
- Avec une phase d'assemblage

Avec édition de liens dynamiques



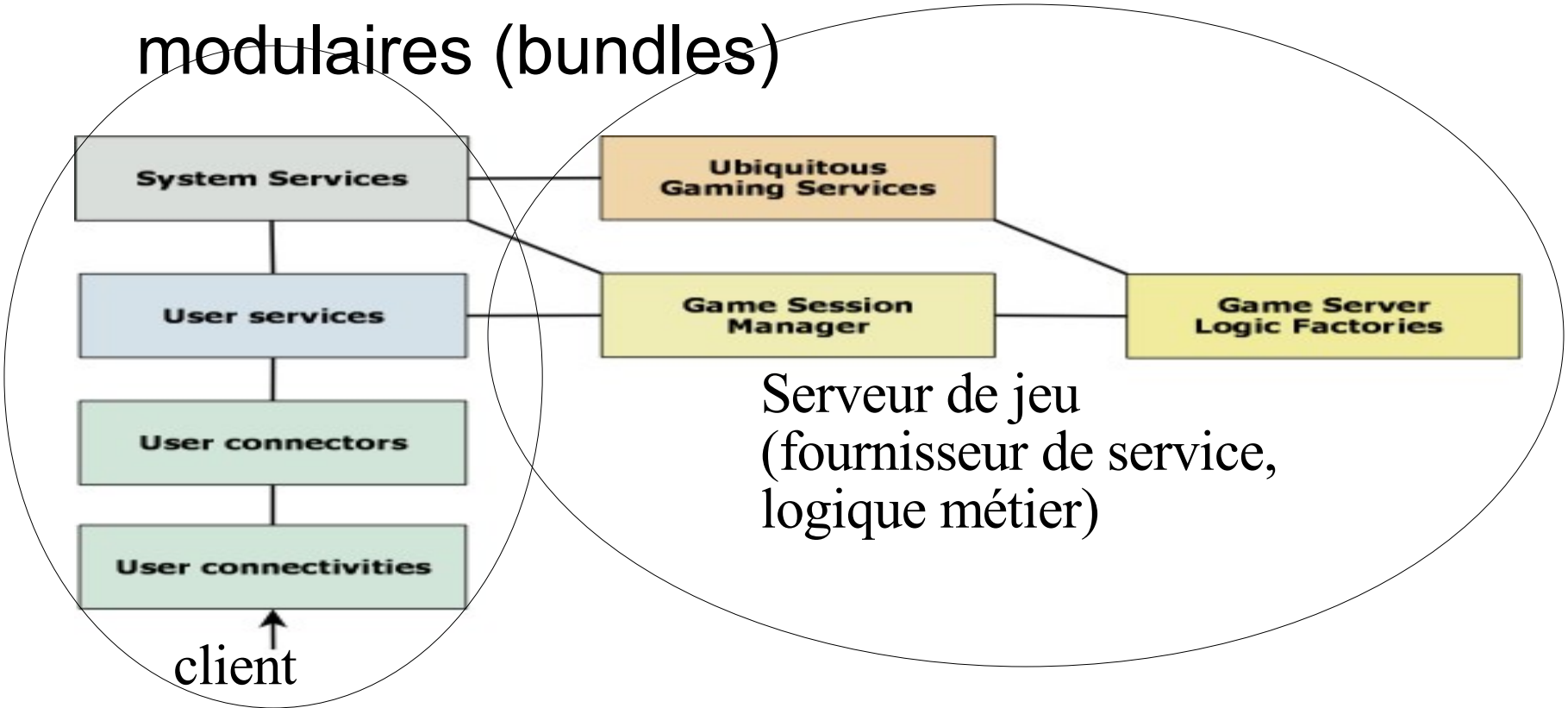
@Eric-Gressier-Soudan

Bilan - OSGI

- OSGI fait partie du paysage des architectures orientées service
- Assure la gestion du cycle de vie d'un service /bundle
  - Déploiement dynamique (sans reboot!)
  - Gestion de versions, configuration, gestion des liens, collaborations
- OSGI vise les systèmes contraints java avec de nouveaux marchés (M2M)

UGASP – OSGI et les jeux ubiquitaires...

- Intergiciel dédié pour supporter les jeux ubiquitaires multi-joueurs
- Basé sur OSGI -> fournit des services modulaires (bundles)



UGASP – OSGI et les jeux ubiquitaires...

- Est reconfigurable à la volée
  - Peut être déployé sur des systèmes embarqués
- Assure la liaison entre le client (le joueur) et le serveur de jeu
- Intergiciel développé avec Eclipse
- Plus d'informations : http://gasp.ow2.org/ubiquitous-osgi-middleware.html
- Une démo...