LSR-Adèle

SEP · CONACYT
Consejo Nacional de Ciencia y Tecnología
MÉXICO

Vers un modèle à composants orienté services pour supporter la disponibilité dynamique

Humberto Cervantes

29 Mars 2004

Plan de la thèse

Problématique

Fondations

Modèle à composants orienté services

Réalisation

Évaluations

Conclusions et perspectives

Plan de la thèse

Problématique

- Disponibilité dynamique

Fondations

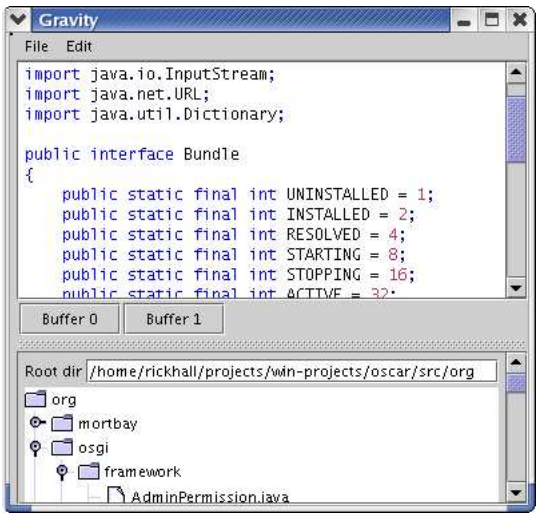
Modèle à composants orienté services

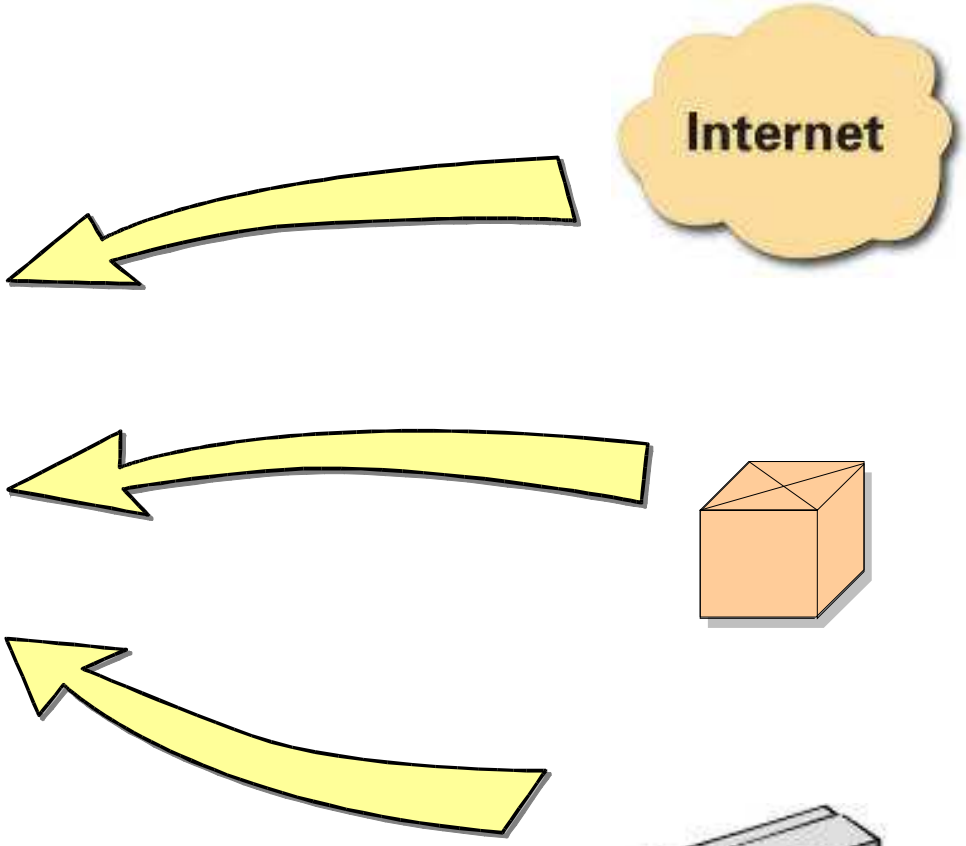
Réalisation

Évaluations

Conclusions et perspectives

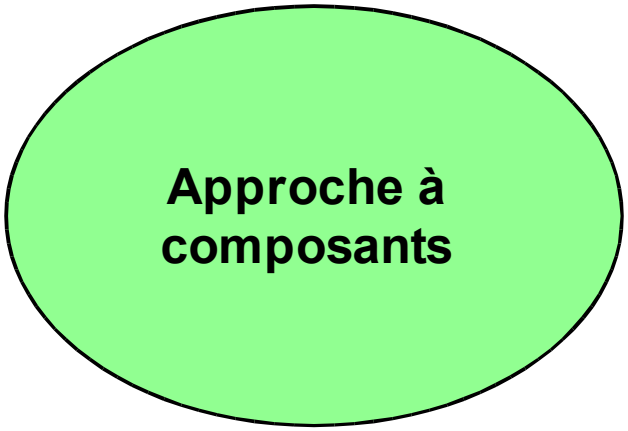
Disponibilité dynamique





Support *explicite* de la disponibilité dynamique?

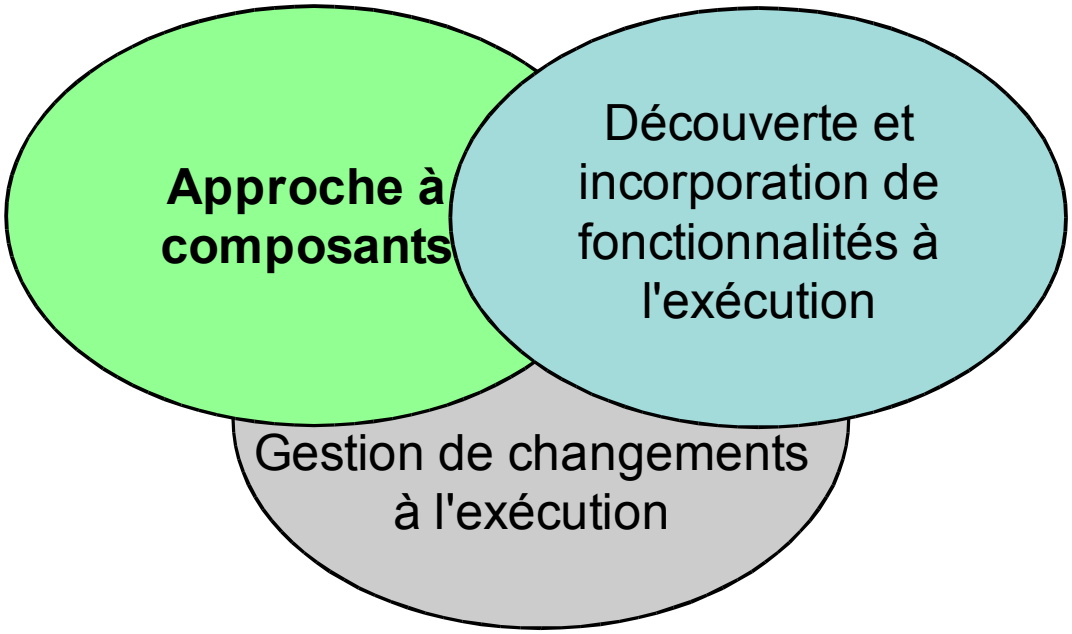
Approche



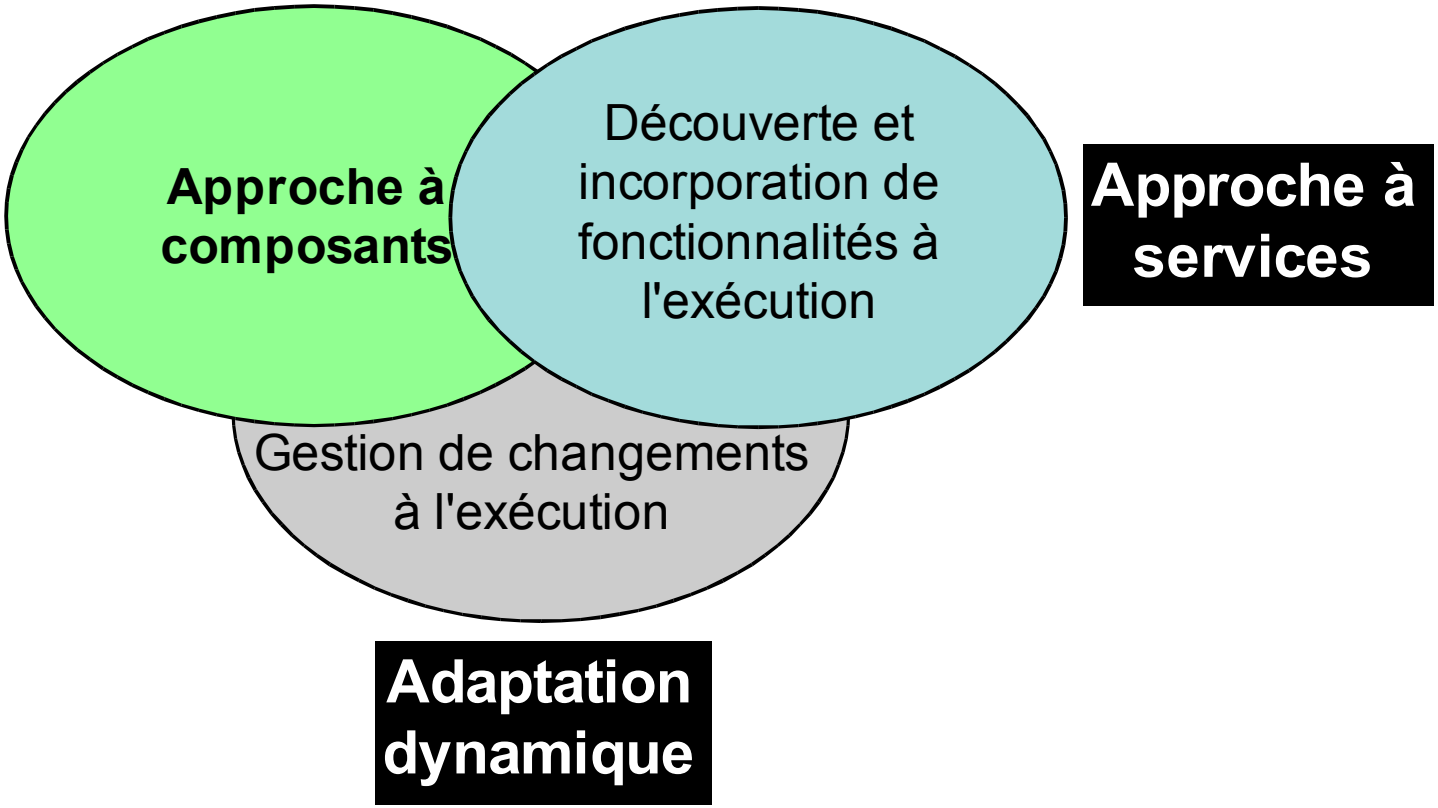
Composants: briques logicielles réutilisables

- Applications construites à partir d'assemblage
- Disponibilité dynamique n'est pas un concept de base

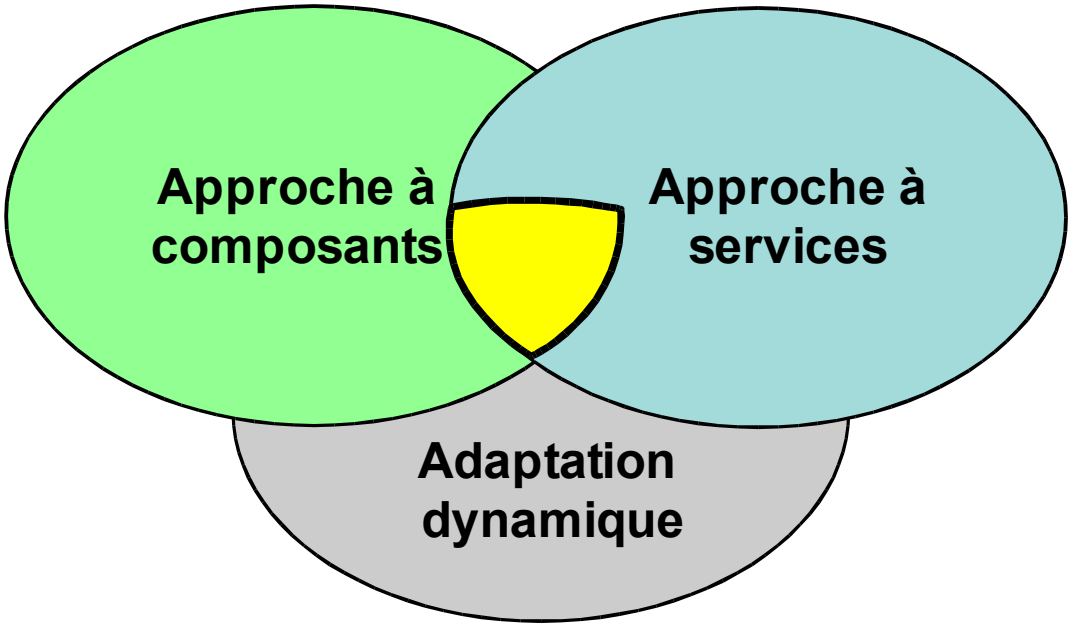
Approche


Approche à composants
Découverte et incorporation de fonctionnalités à l'exécution
Gestion de changements à l'exécution

Approche


Approche à composants
Découverte et incorporation de fonctionnalités à l'exécution
Approche à services
Gestion de changements à l'exécution
Adaptation dynamique

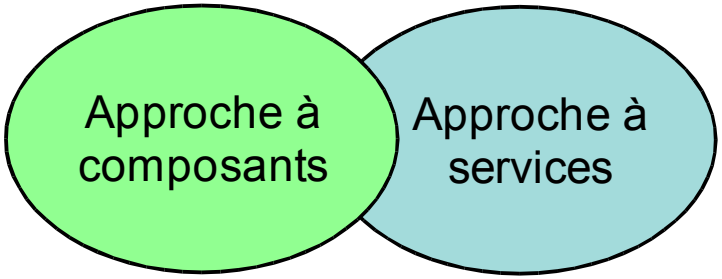
Approche



Modèle à composants orienté services

- Support *explicite* de la disponibilité dynamique dans un modèle à composants

Approches existantes



Combinaison concepts services et composants

- Avalon
- Web service components
- CCM et Traders

Se focalisent sur découverte en temps d'exécution

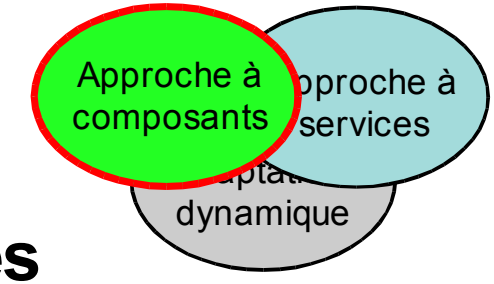
- Pas de gestion de la disponibilité dynamique

Plan de la thèse

Problématique

Fondations

- Approche à composants
- Approche à services
- Adaptation dynamique



Modèle à composants orienté services

Réalisation

Évaluations

Conclusions et perspectives

Approche à composants

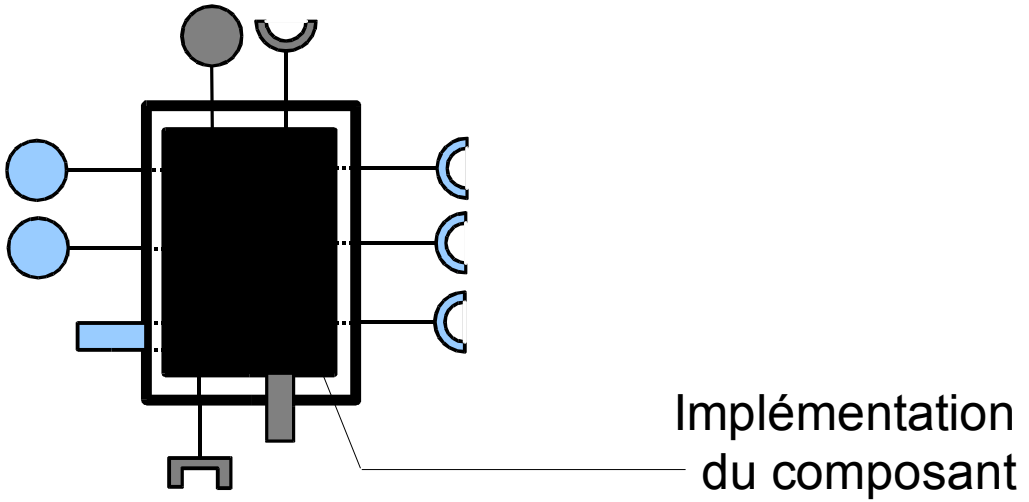
Définition de Szyperski

- unité binaire de composition
- interfaces spécifiées de façon contractuelle
- dépendances explicites
- déployé de façon indépendante
- sujet à composition par des tierces

Mais aussi...

- Paquetage
- Classe (composant)
- Instance

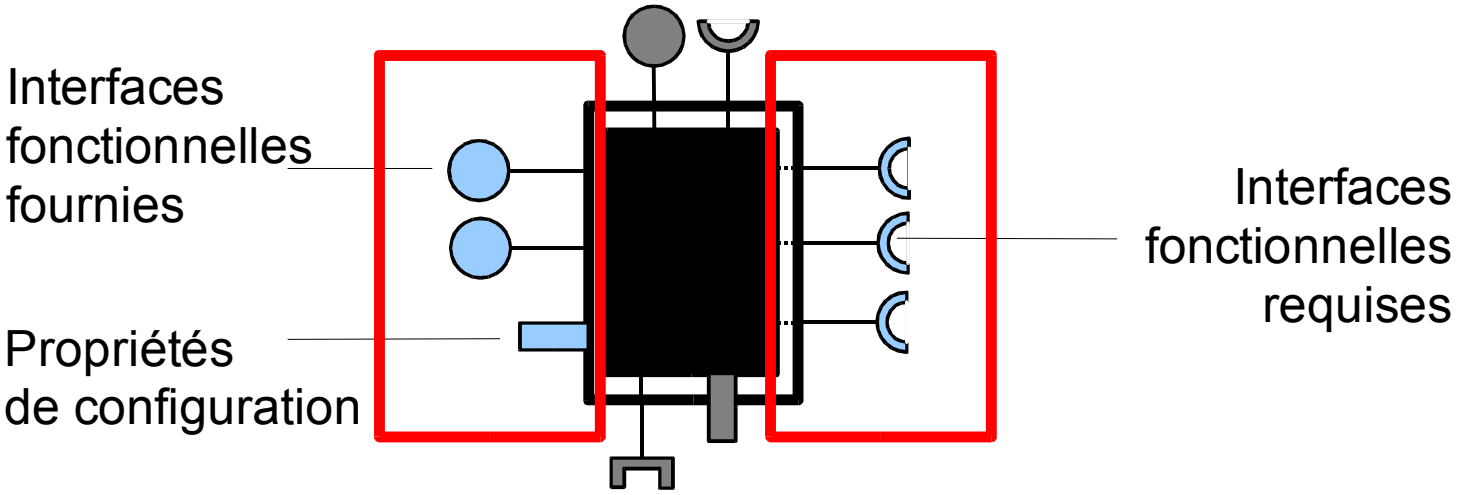
Éléments fournis et requis



Composant définit explicitement ce qu'il fournit et ce qu'il requiert

- Assemblage
- Déploiement
- Gestion des instances à l'exécution

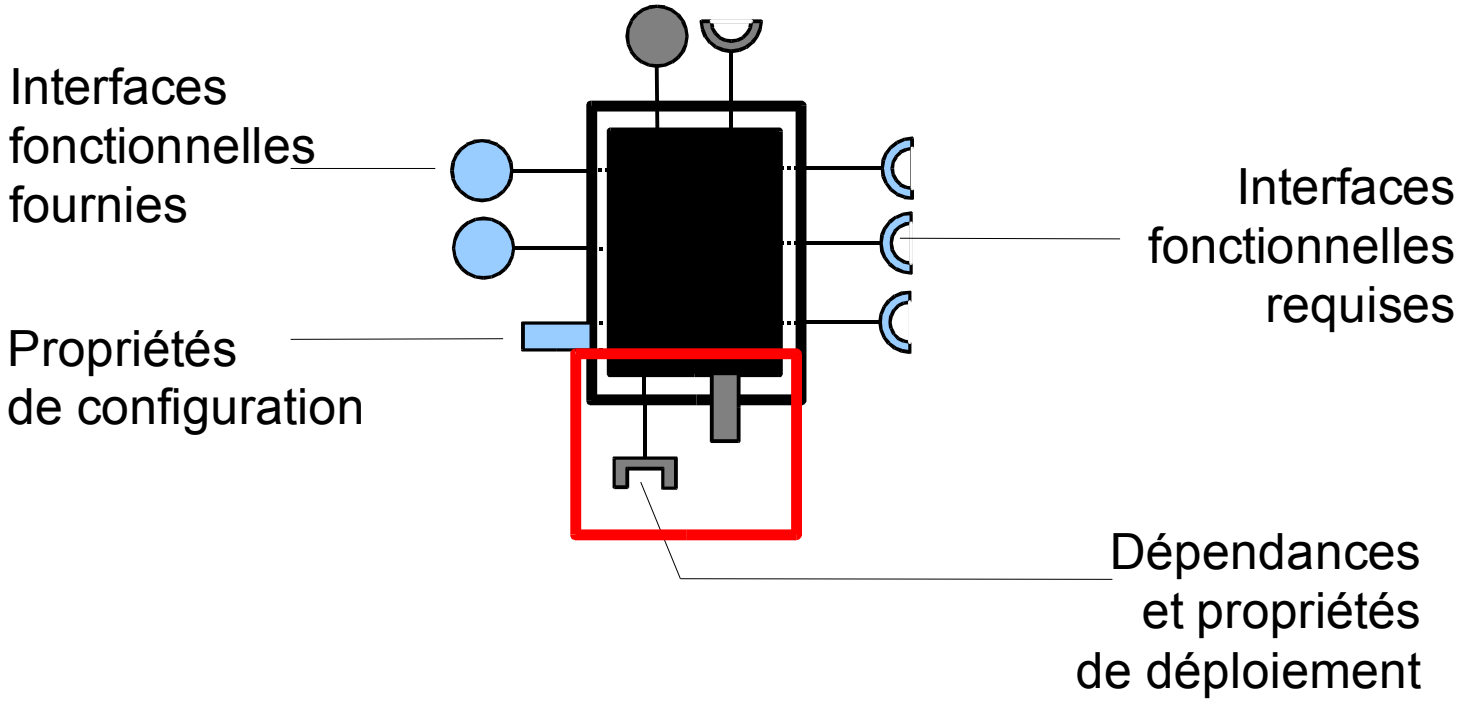
Éléments liés à l'assemblage



Représentent la *vue externe* du composant

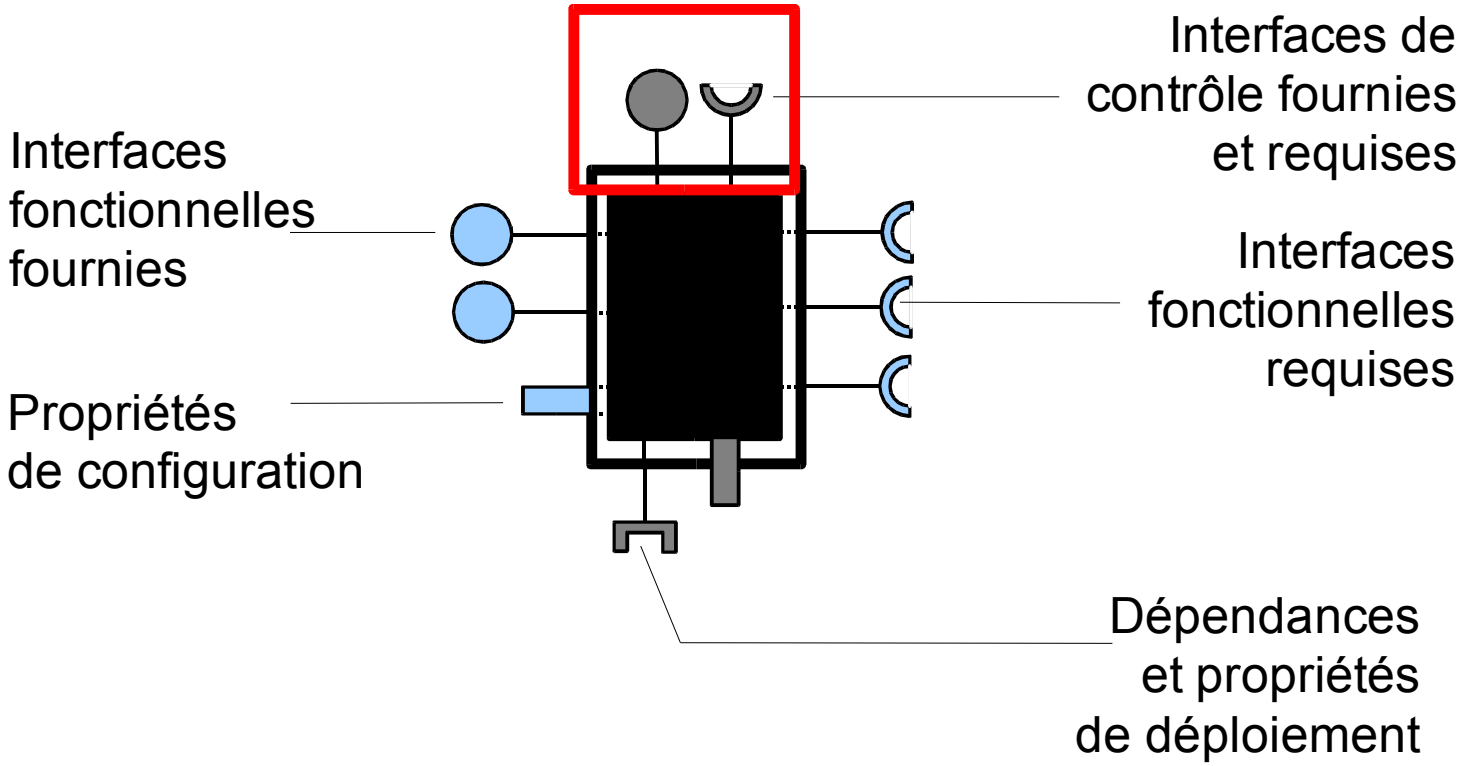
- Ce que voient les clients
- Aspects fonctionnels
- Type de composant

Éléments liés au déploiement



Remplies au moment du déploiement

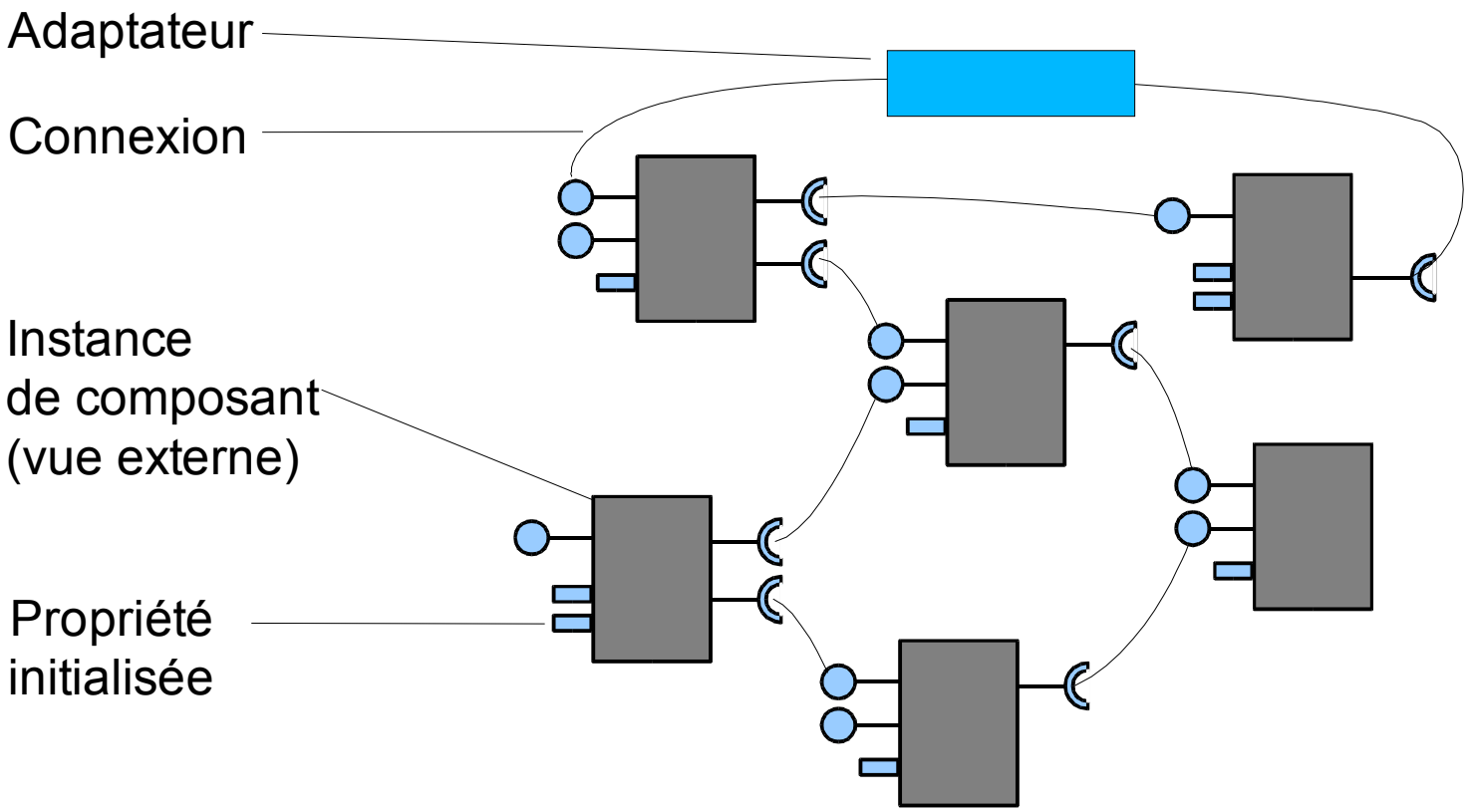
Éléments liés à l'exécution



Gestion du cycle de vie des instances

Assemblage et composition

Assemblage réalisé à travers *la composition*.



Assemblage et composition

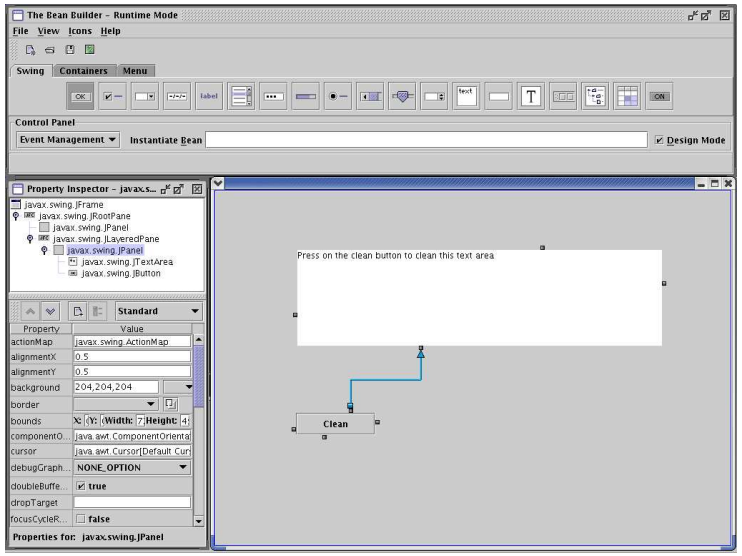
Types de *composition*.

```
System simple_cs = {
  Component client = {Port sendRequest }
  Component server = {Port receiveRequest}
  Connector rpc = { Roles {caller, callee} }
  Attachments : {
    client.sendRequest to rpc.caller;
    server.receiveRequest to rpc.callee }
}
```

Déclaratif

```
Server server = (Server) Beans.instantiate
(this.getClassLoader(),
"org.examples.Server");
Client client = (Client) Beans.instantiate
(this.getClassLoader(),
"org.examples.Client");
server.add(client);
```

Impératif



Visuel

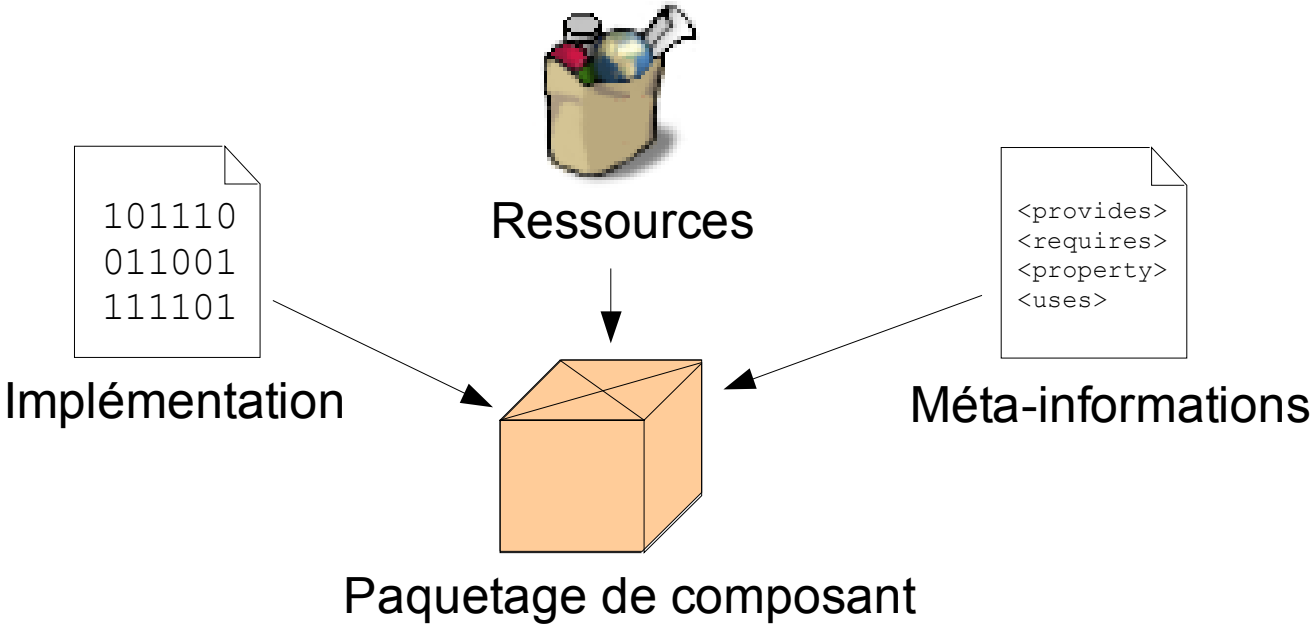
Assemblage et composition

Composition *hiérarchique*

Paquetage de composant

Livraison et déploiement indépendants

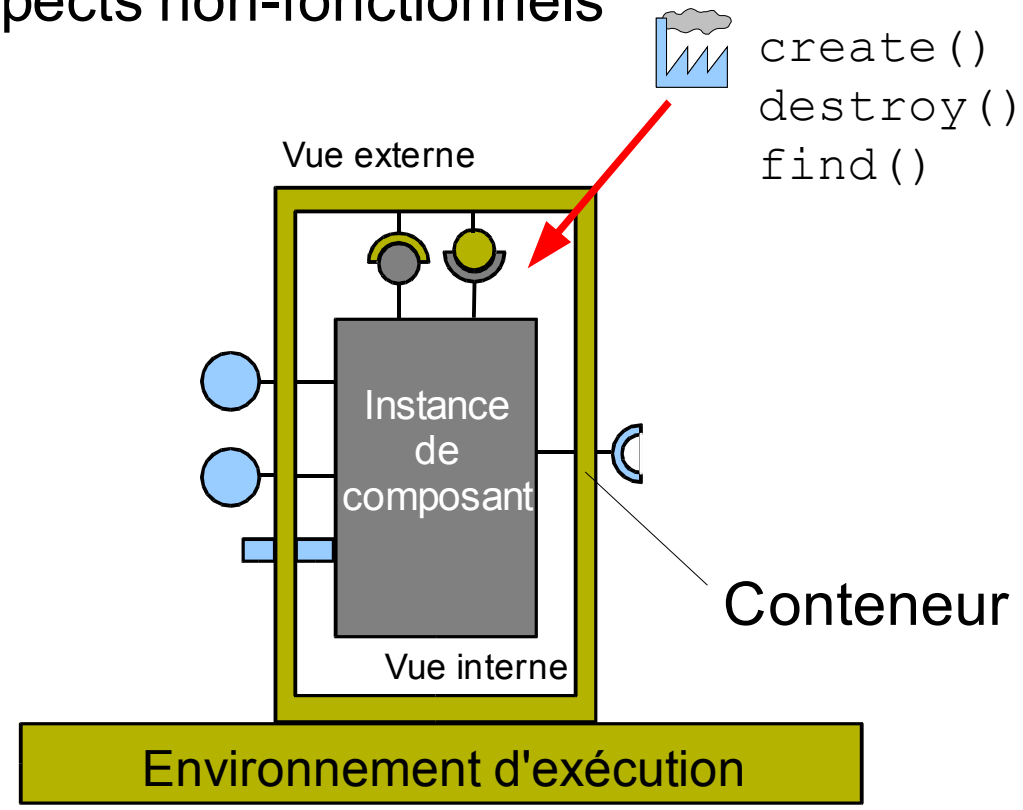
- Paquetage contient tout ce qui est nécessaire au composant, sauf les dépendances explicites
- Binaire



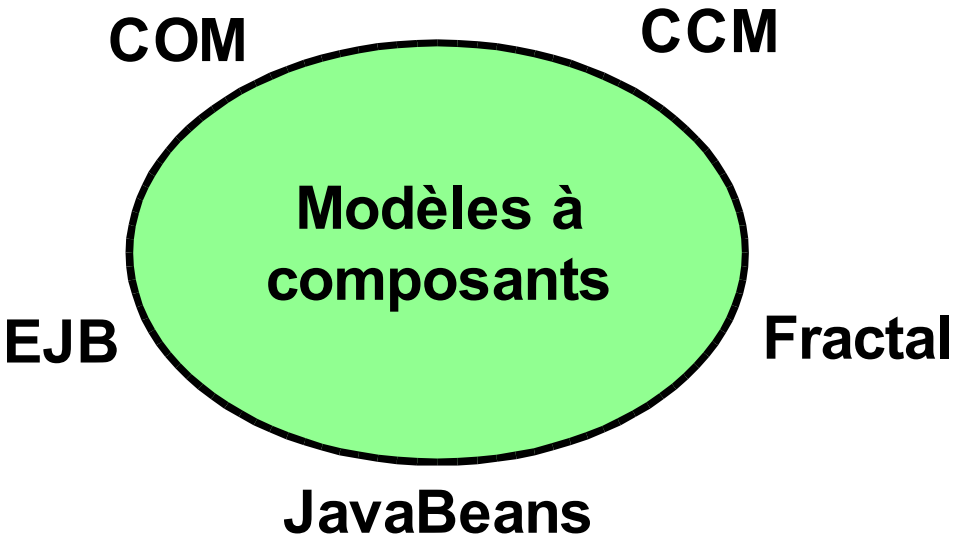
Fabriques et conteneurs

Instances créées à partir de fabriques et placées dans un conteneur

- Gestion du cycle de vie
- Gestion d'aspects non-fonctionnels



Modèles à composants

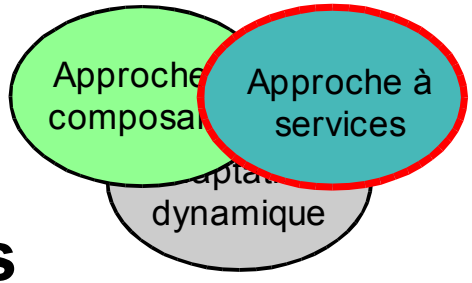


Plan de la thèse

Problématique

Fondations

- Approche à composants
- Approche à services
- Adaptation dynamique



Modèle à composants orienté services

Réalisation

Évaluations

Conclusions et perspectives

Approche à services

Services : fonctionnalités réutilisables

- Décrits de façon contractuelle
- Fournisseurs de service substituables

Assemblage réalisé à partir de descripteurs

- Informations syntaxiques + sémantiques

Intégration *tardive* des fournisseurs

- Mécanismes de découverte dynamique
- Patron d'interaction de l'approche à services

Disponibilité dynamique est une hypothèse

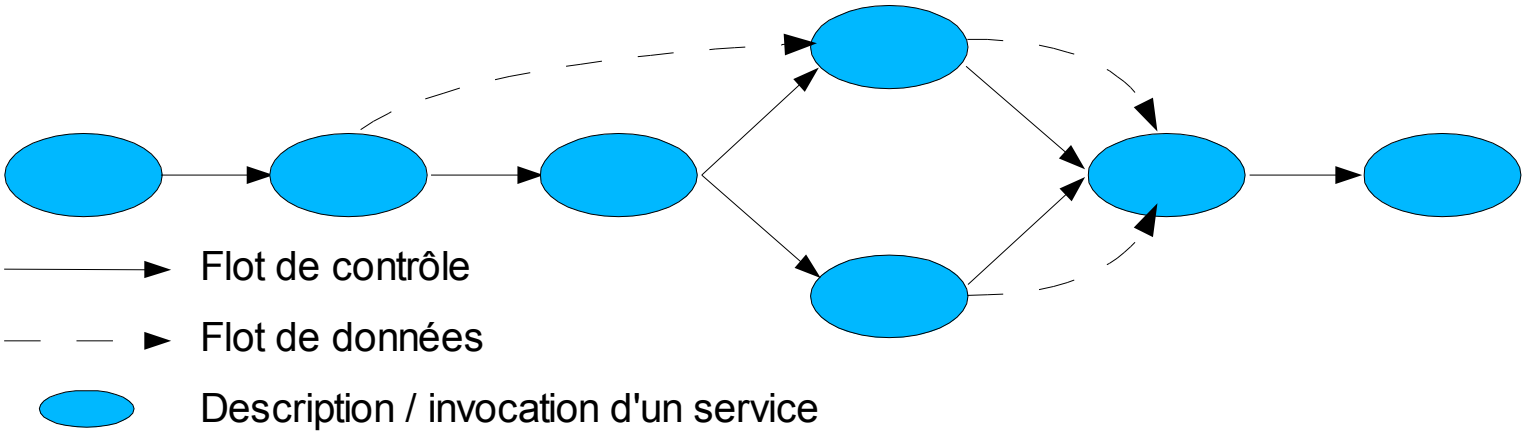
Composition de services

Composition basée sur descripteurs de service

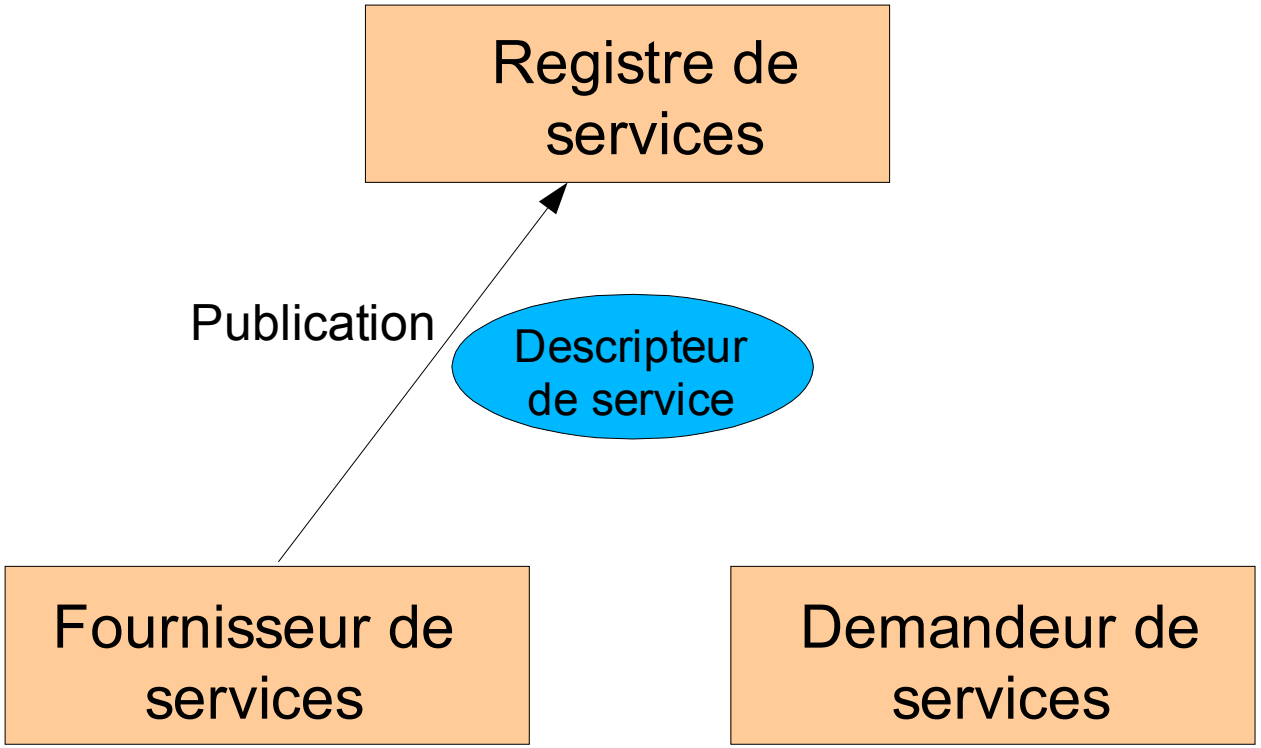
- Fournisseurs intégrés avant ou durant exécution

Approche à flot favorisée

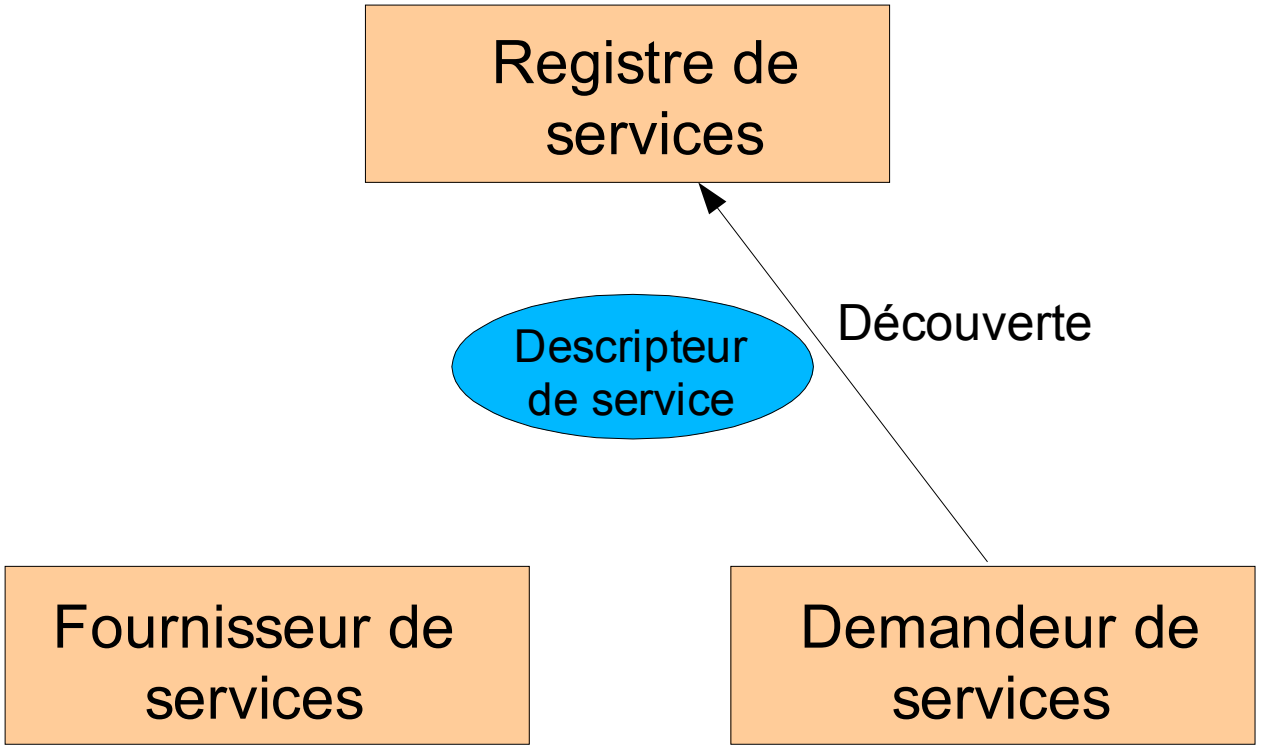
- Invocation séquentielle
- Transfert de données



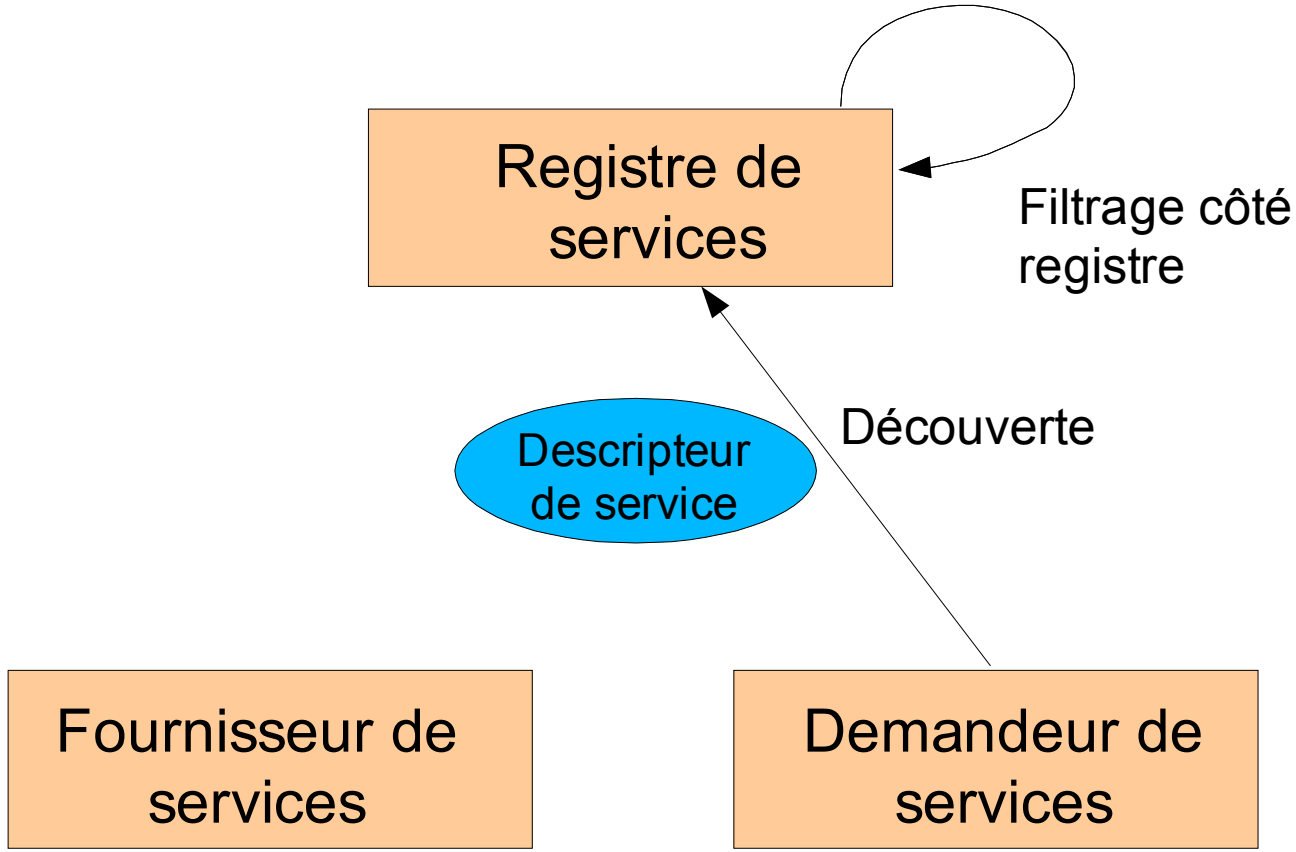
Patron d'interaction



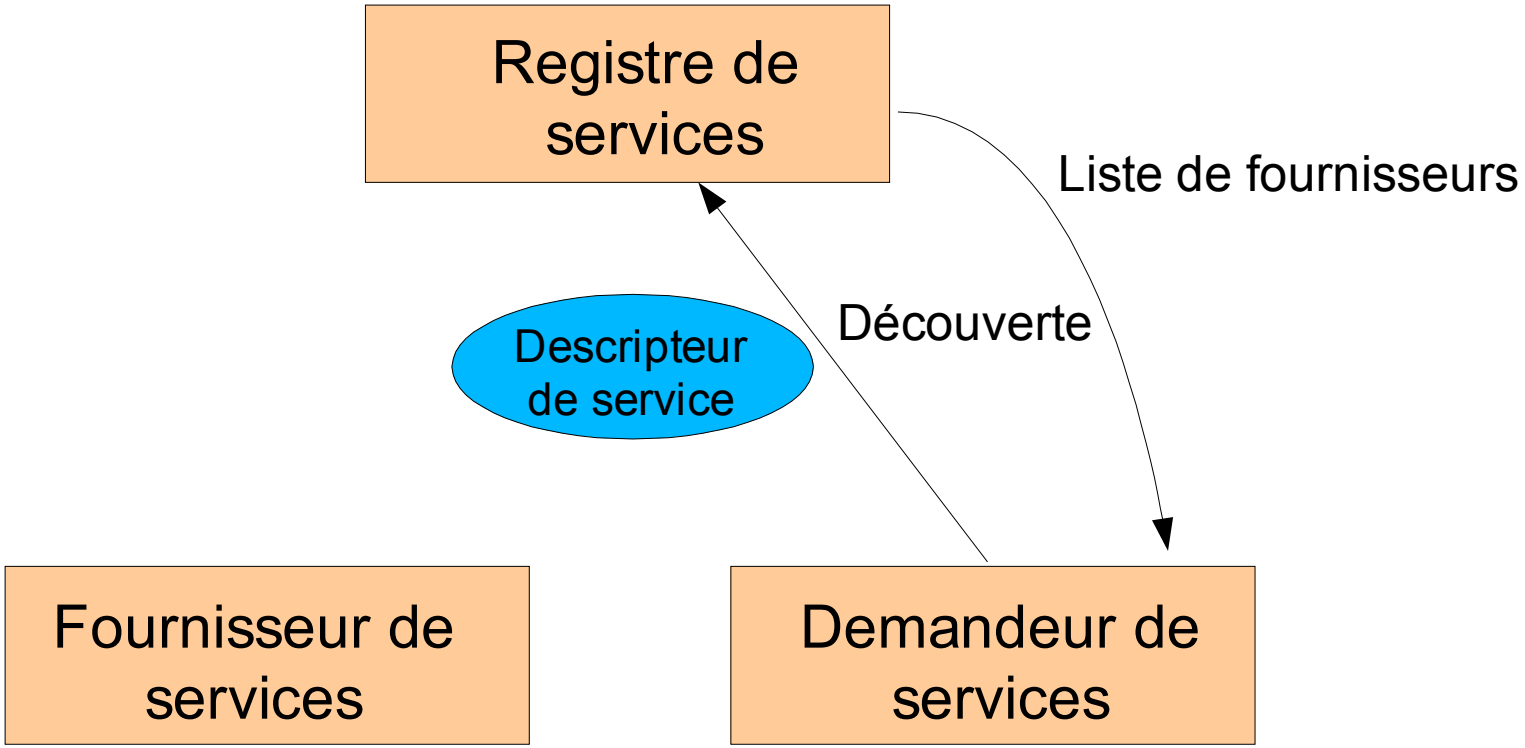
Patron d'interaction



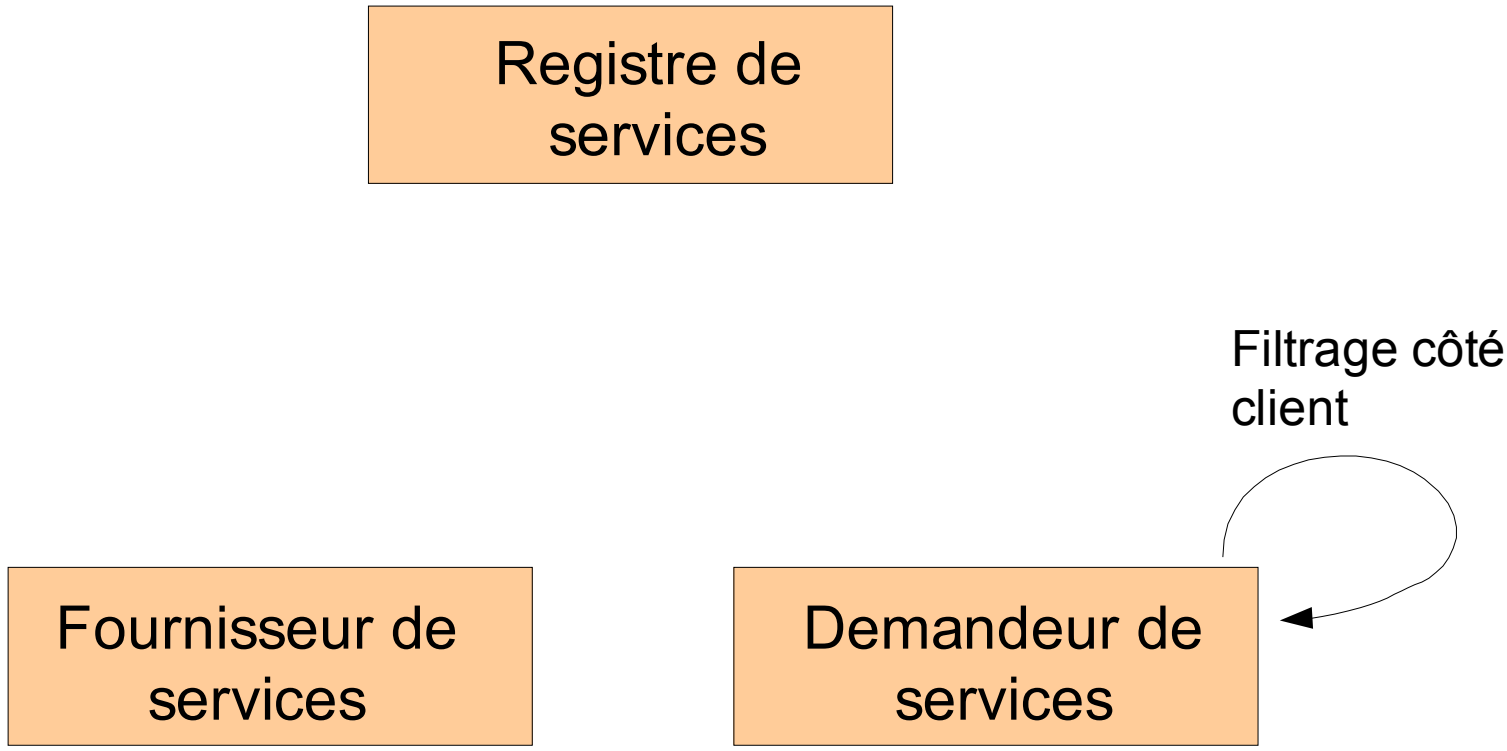
Patron d'interaction



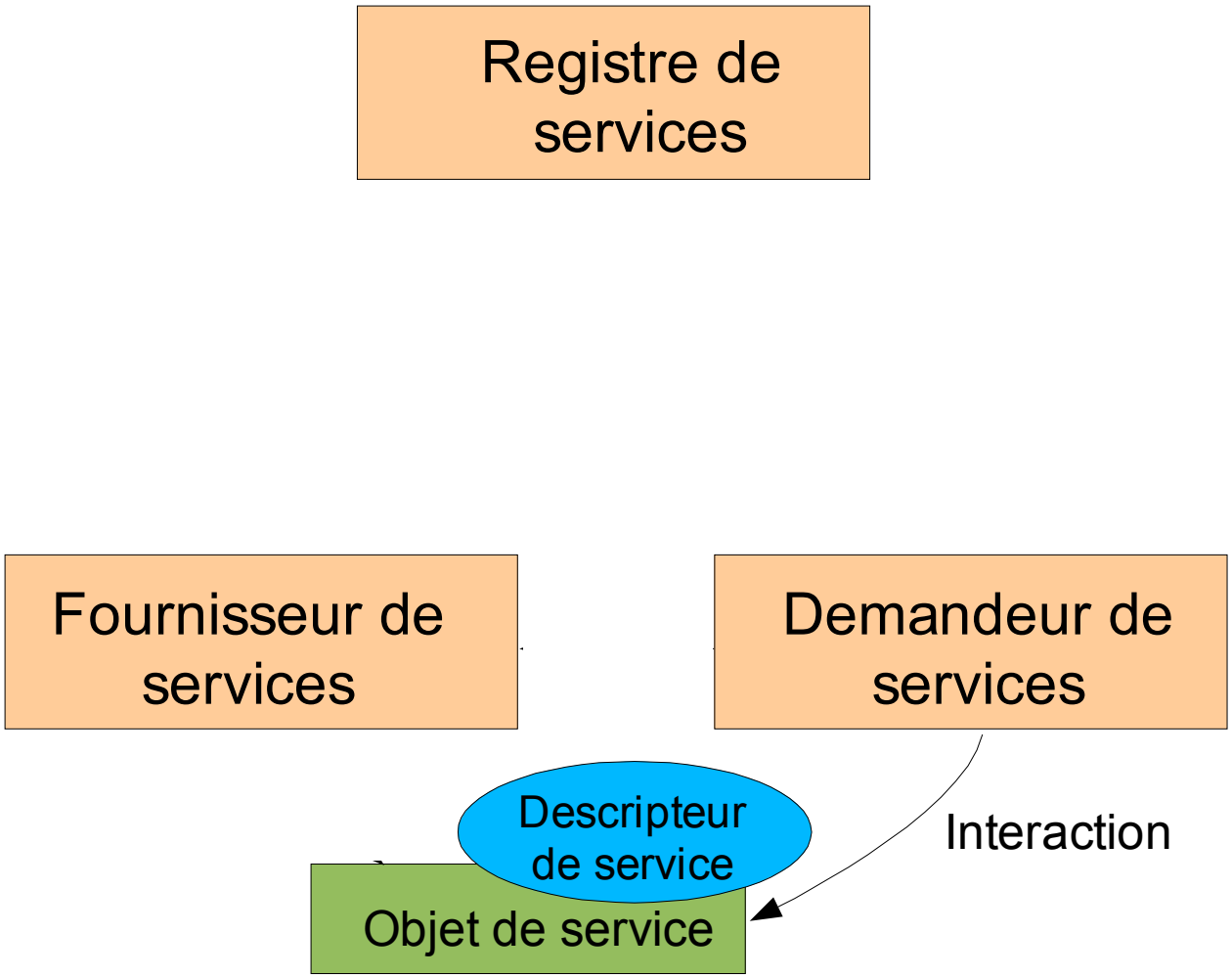
Patron d'interaction



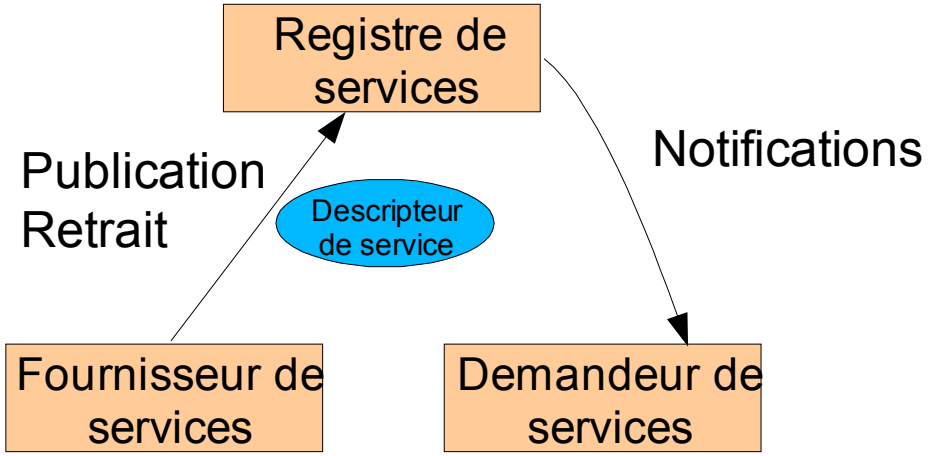
Patron d'interaction



Patron d'interaction



Disponibilité dynamique



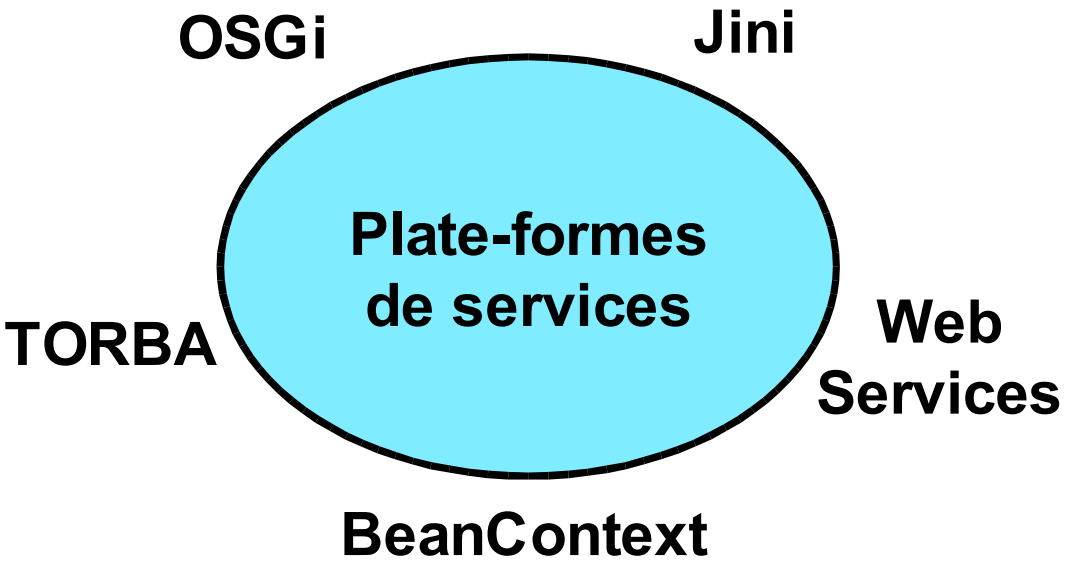
Mécanismes de notification

- Arrivée, départ, changement des services

Comment gérer les changements?

- A la charge des demandeurs
- Mécanismes de transactions dans composition à flots

Plate-formes de services

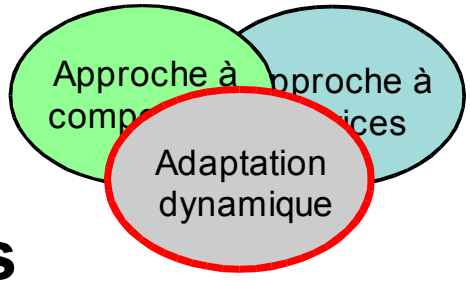


Plan de la thèse

Problématique

Fondations

- Approche à composants
- Approche à services
- Adaptation dynamique



Modèle à composants orienté services

Réalisation

Évaluations

Conclusions et perspectives

Adaptation dynamique

Adaptation d'une application pendant l'exécution

- Application adaptative
- Auto-adaptation

Logique d'adaptation

- Supervision
- Adaptation

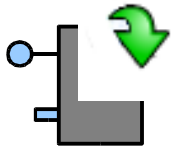
Adaptation à travers *reconfiguration dynamique*

- Modification de l'architecture d'une application à l'exécution

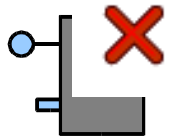
Reconfiguration dynamique

Opérations primitives

- create
- destroy
- bind
- unbind
- set
- move



`create()`



`destroy()`

`bind()`

`unbind()`

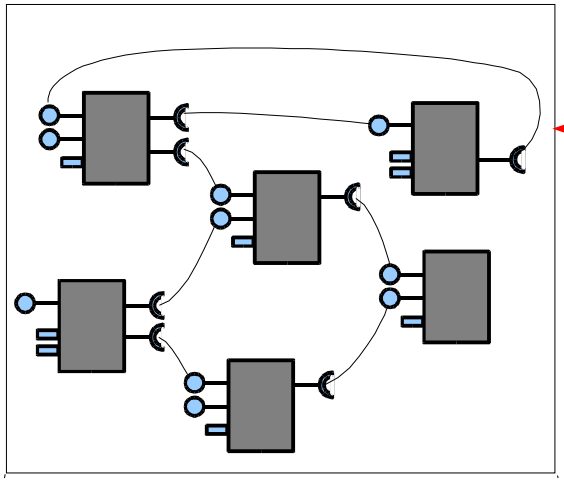
A travers une représentation architecturale

- Participation des composants

Préservation de la consistance

- État correct après changements

Adaptation dynamique



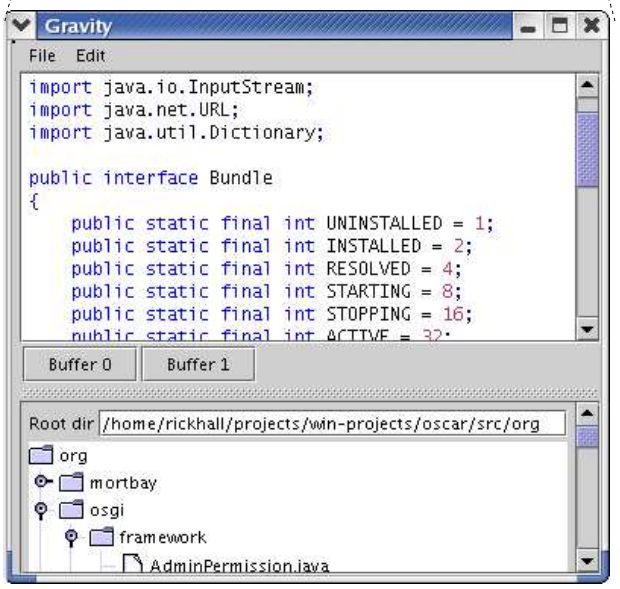
Opérations primitives

Logique d'adaptation

Supervision de conditions externes

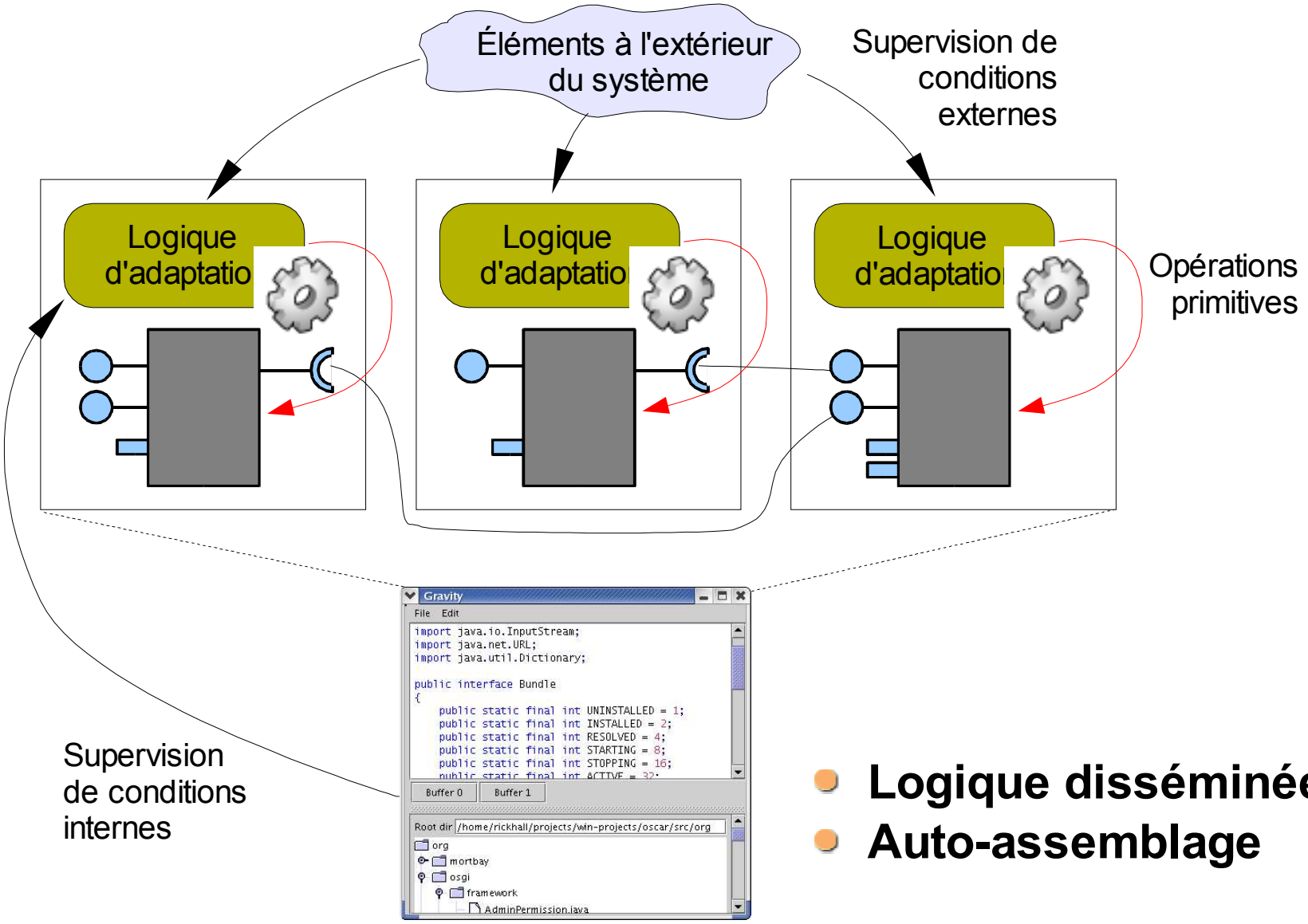
Supervision de conditions internes

Éléments à l'extérieur du système



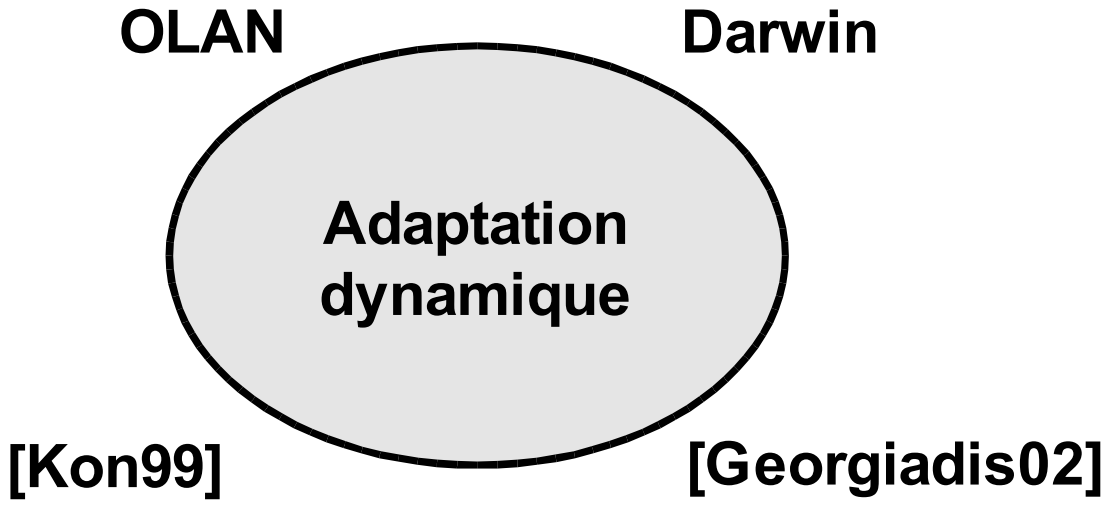
- **Logique centralisée**

Adaptation dynamique



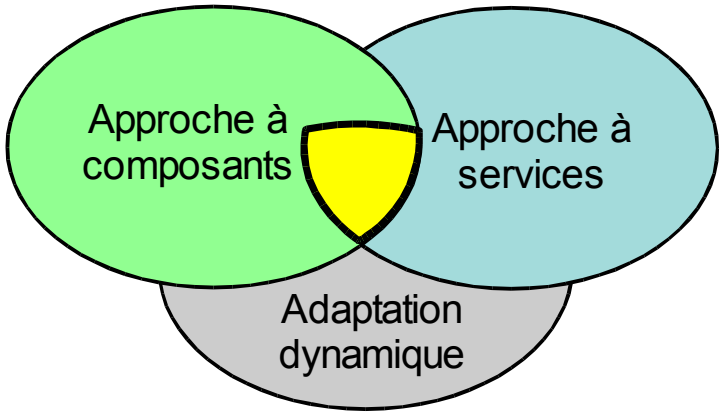
- **Logique disséminée**
- **Auto-assemblage**

Adaptation dynamique



Modèle à composants orienté services

- Fondation



- Découverte dynamique
- Intégration tardive
- Disponibilité dynamique

- Auto-adaptation
- Logique disséminée
- Adaptation gérée par environnement d'exécution

Plan de la thèse

Problématique

Fondations

Modèle à composants orienté services

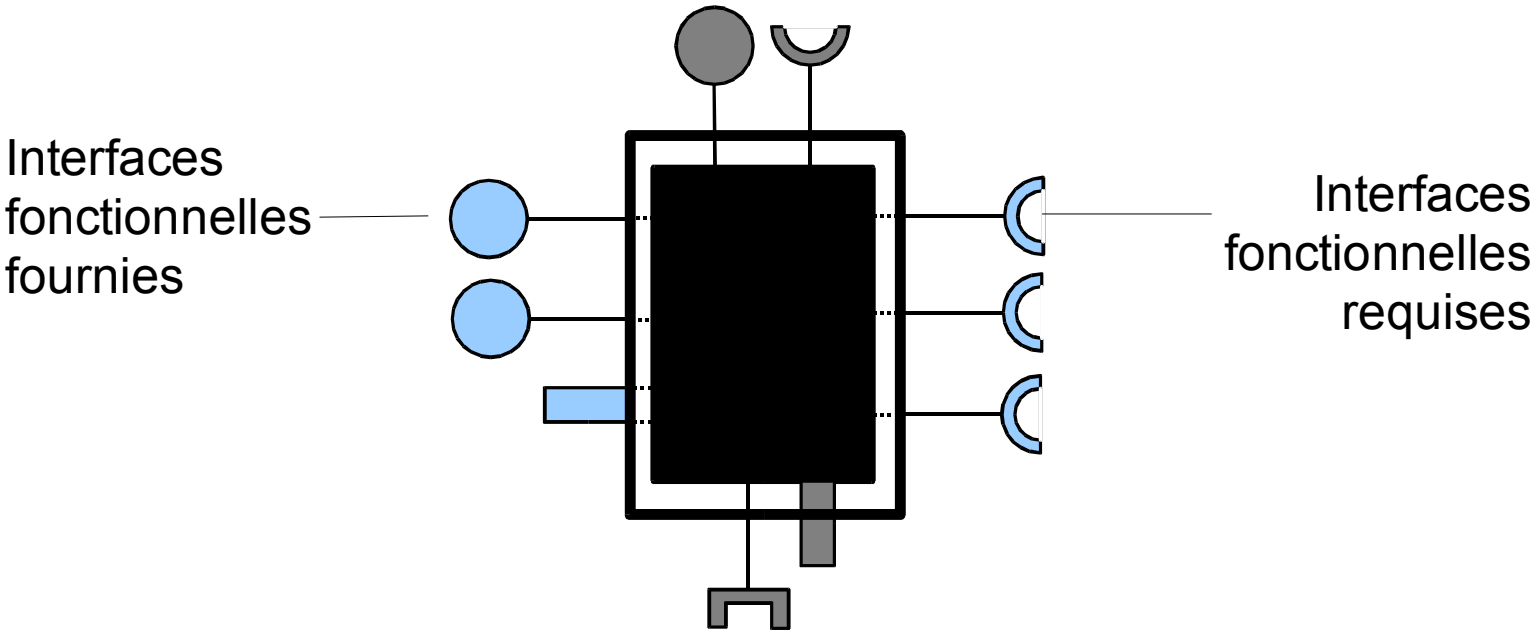
- Composants à services
- Gestion des instances
- Assemblage et évolution d'une application
- Types d'instances
- Imprévisibilité et espaces de résolution
- Gestion de composition
- Descripteur de composition

Réalisation

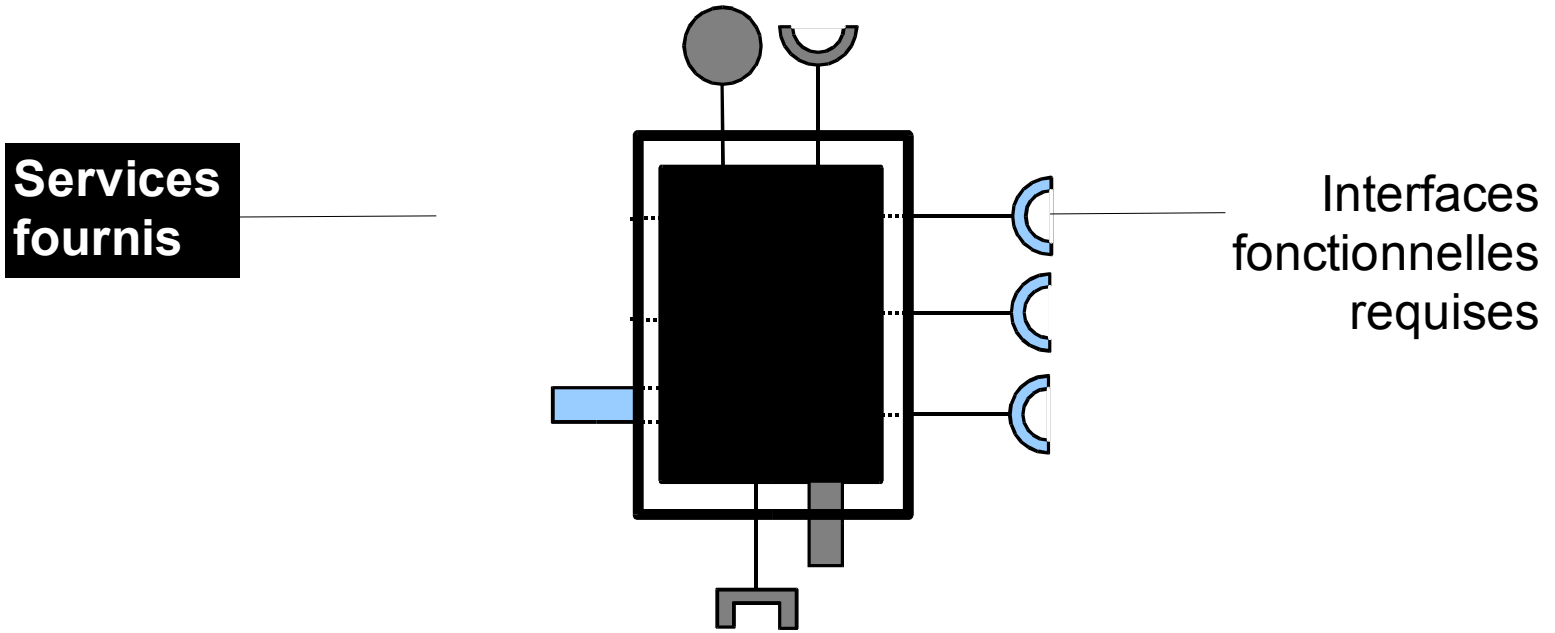
Évaluations

Conclusions et perspectives

Composant à services



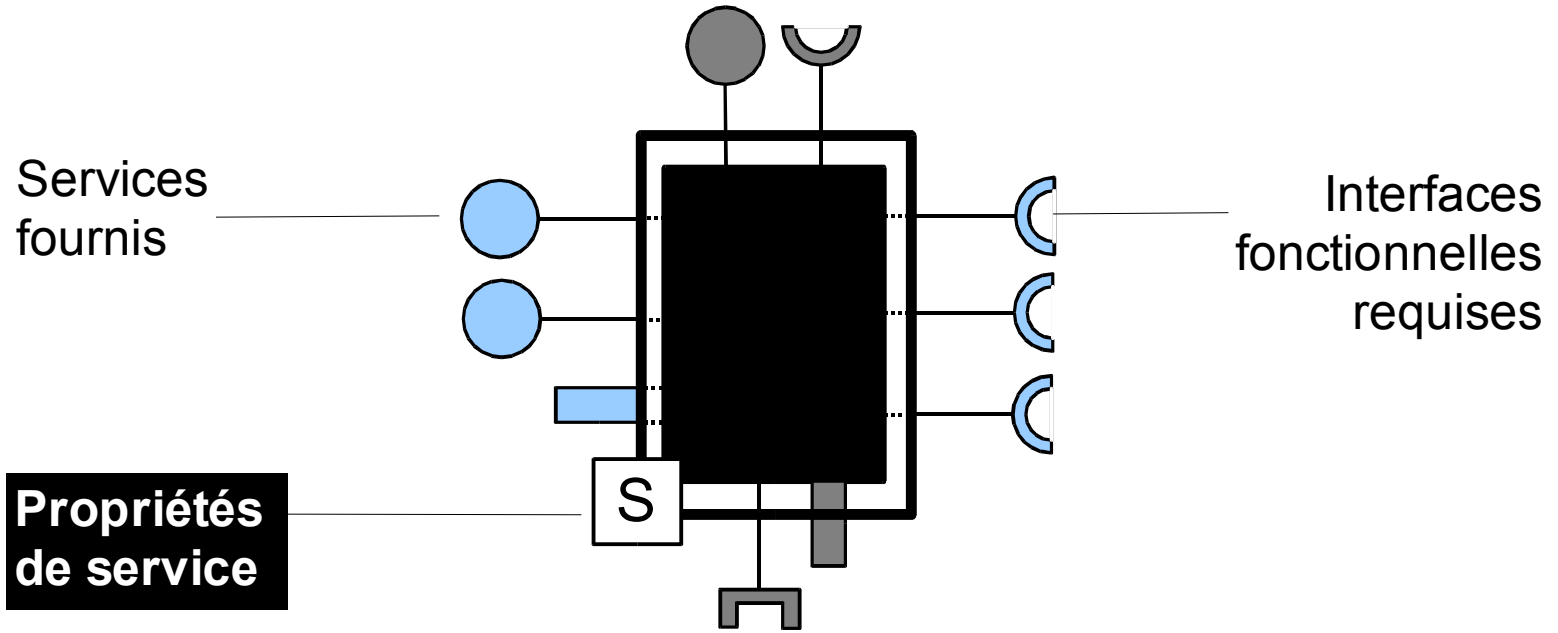
Composant à services



Publication dans un registre de services

- Services fournis par les instances du composant
- Disponibles dynamiquement par rapport à *validité*

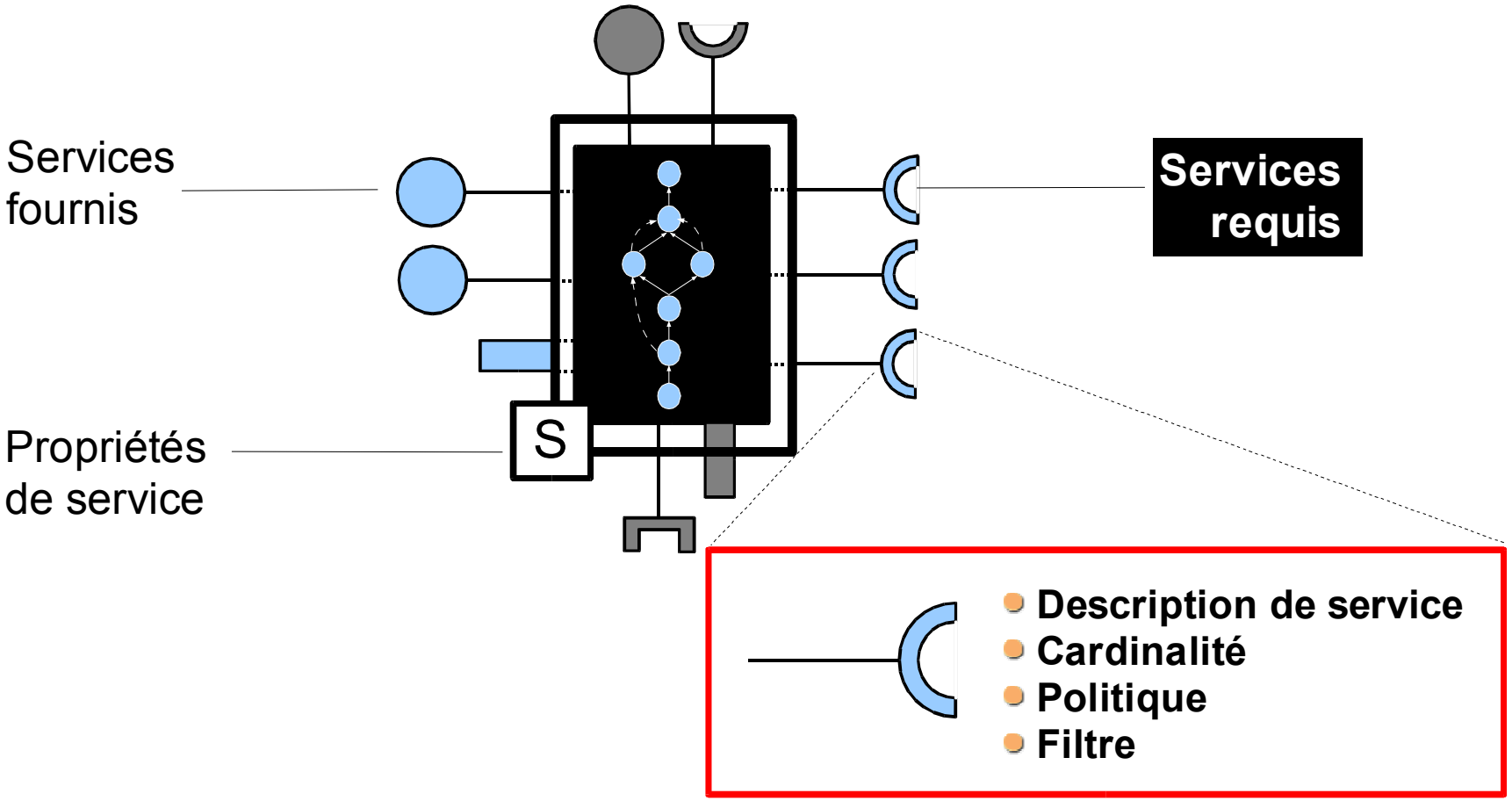
Composant à services



Distinguent fournisseurs d'un même service

- <clé, valeur>

Composant à services



Plan de la thèse

Problématique

Fondations

Modèle à composants orienté services

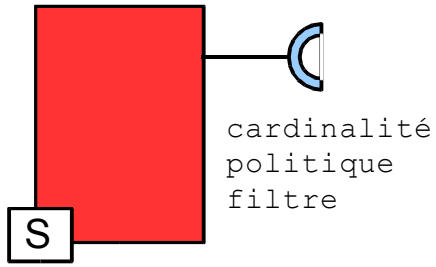
- Composants à services
- Gestion des instances
- Assemblage et évolution d'une application
- Types d'instances
- Imprévisibilité et espaces de résolution
- Gestion de composition
- Descripteur de composition

Réalisation

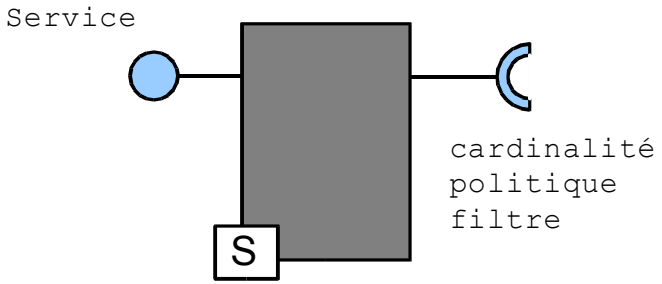
Évaluations

Conclusions et perspectives

Validité des instances



Instance *invalide*



Instance *valide*

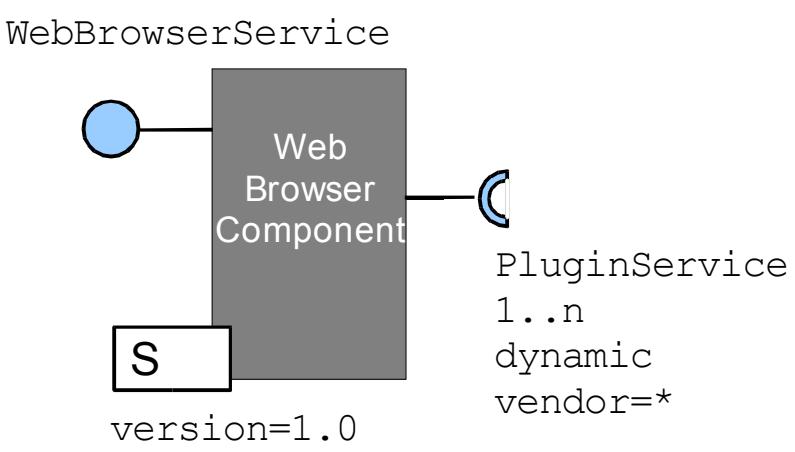
Création intentionnelle

- Création ne garantit pas la validité
- Validité change pendant l'exécution

Validité

- par rapport aux services requis

Services requis



Cardinalité

- Optionnelle, obligatoire
- Simple, multiple

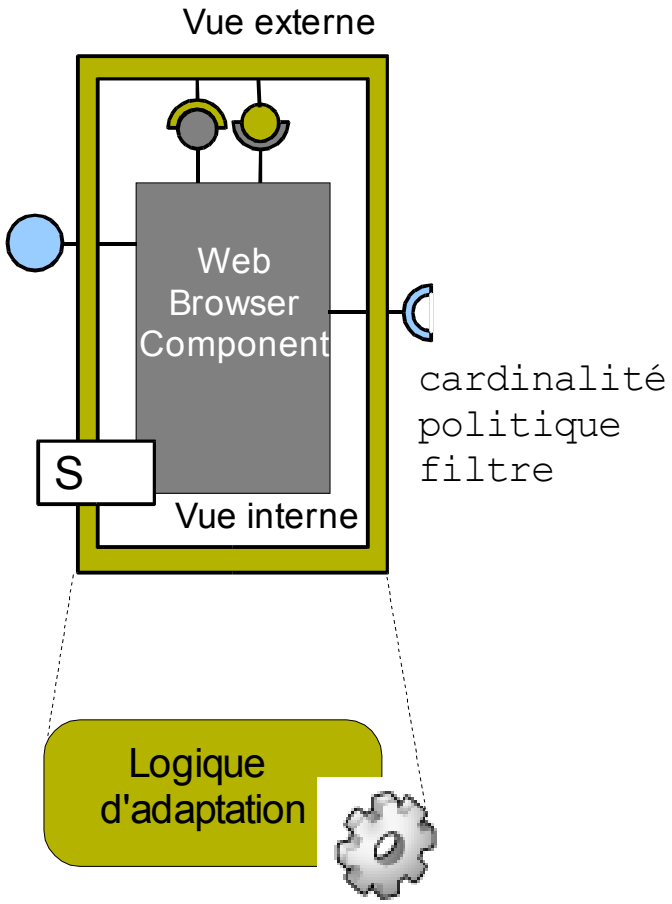
Politique

- Statique, dynamique

Filtre

- Par rapport aux propriétés de service

Gestionnaire d'instance



Conteneur indépendant

- Créé avec instance

Résponsabilités

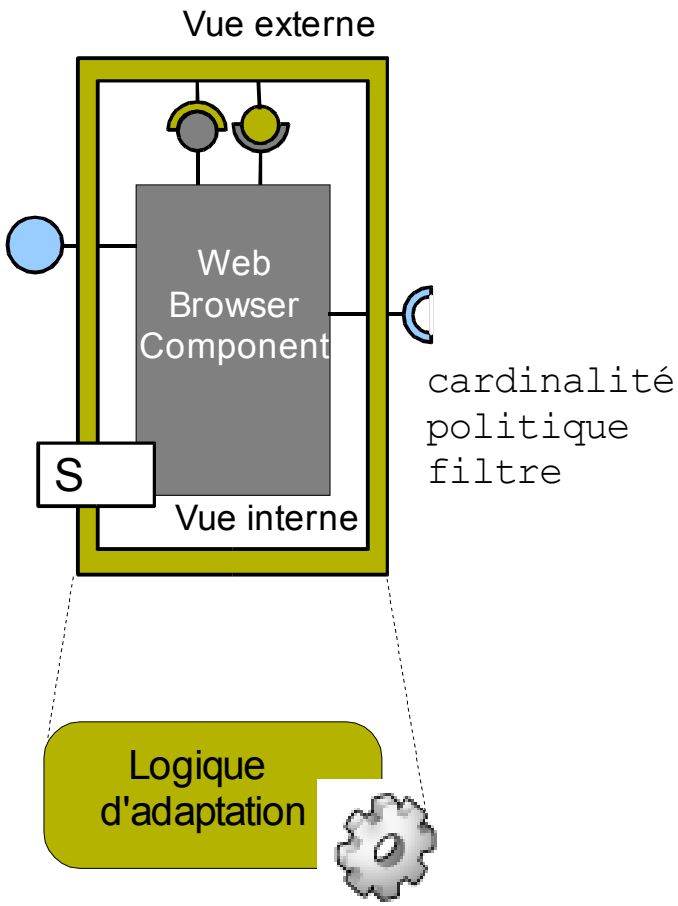
- Publication de services
- Gestion des dépendances
- Maintien de la validité

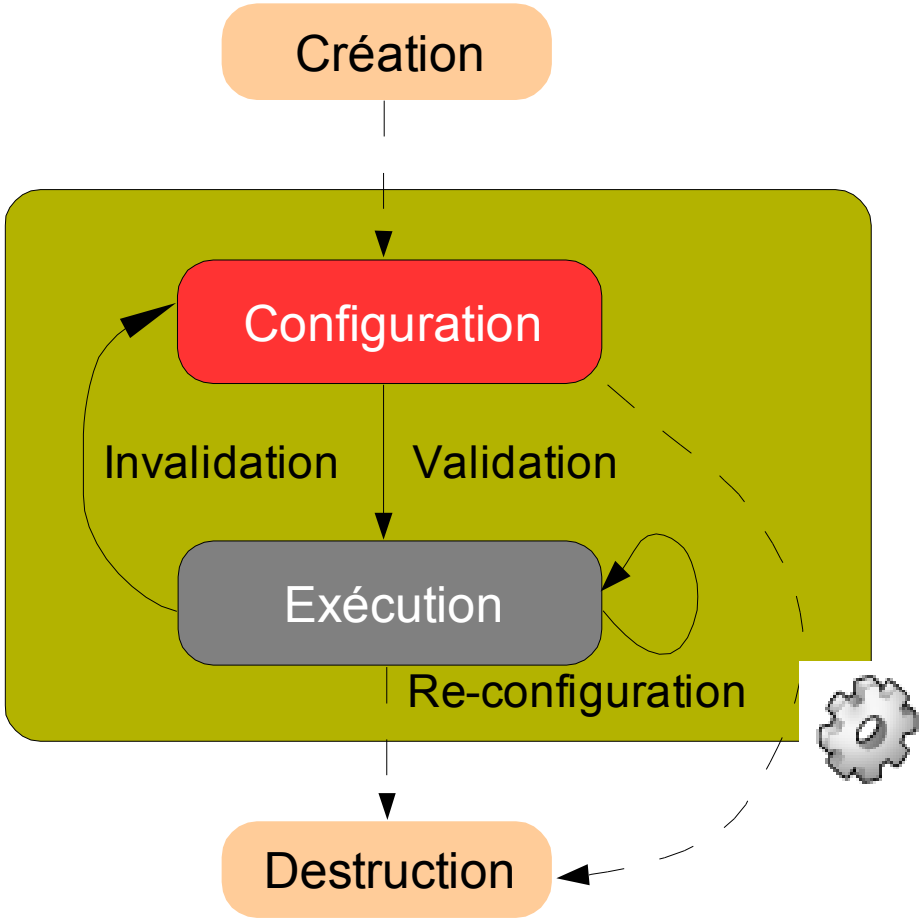
Gestion des dépendances

- Supervision des changements
- Reconfigurations

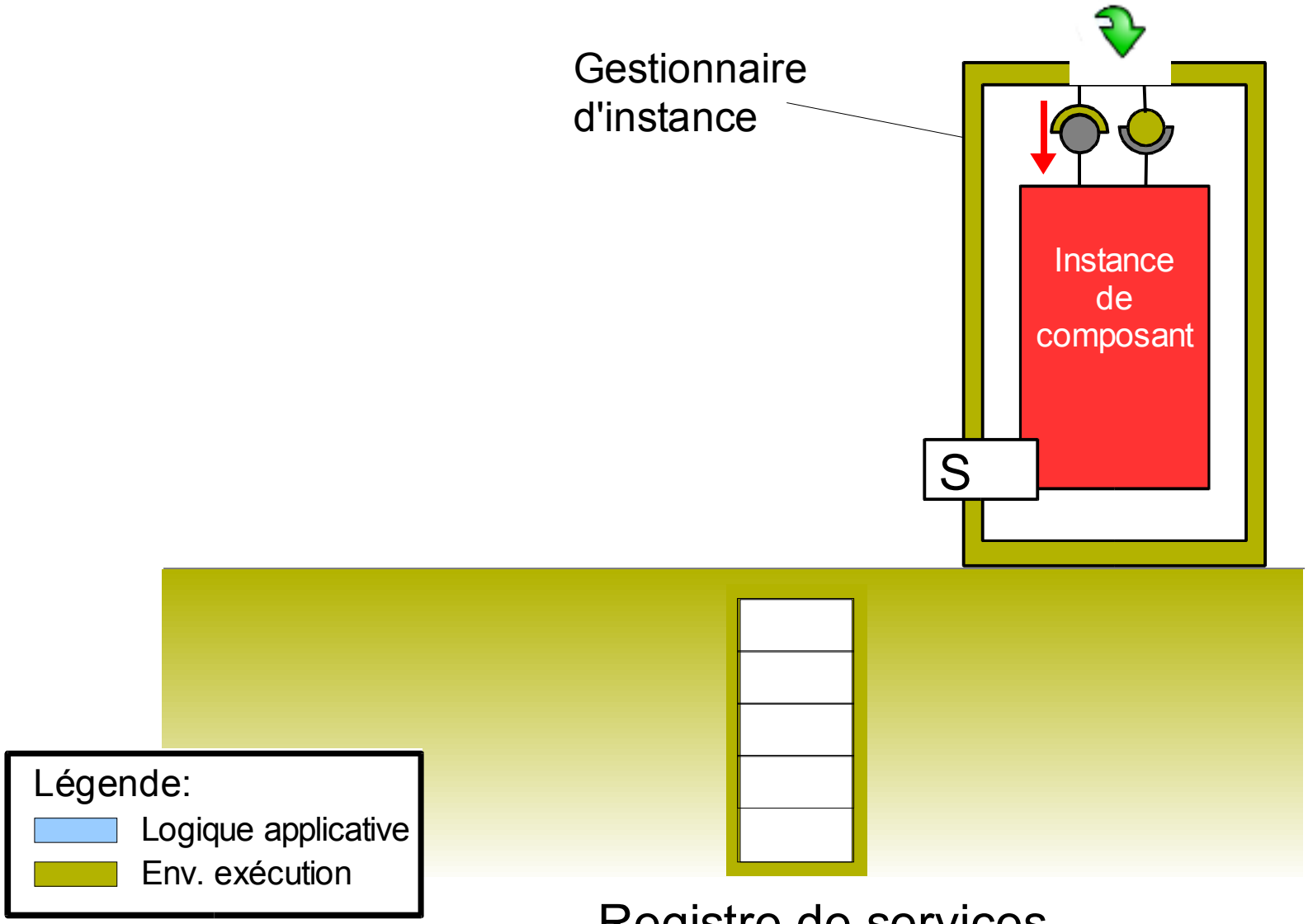
  Opérations `bind` et `unbind`

Gestionnaire d'instance

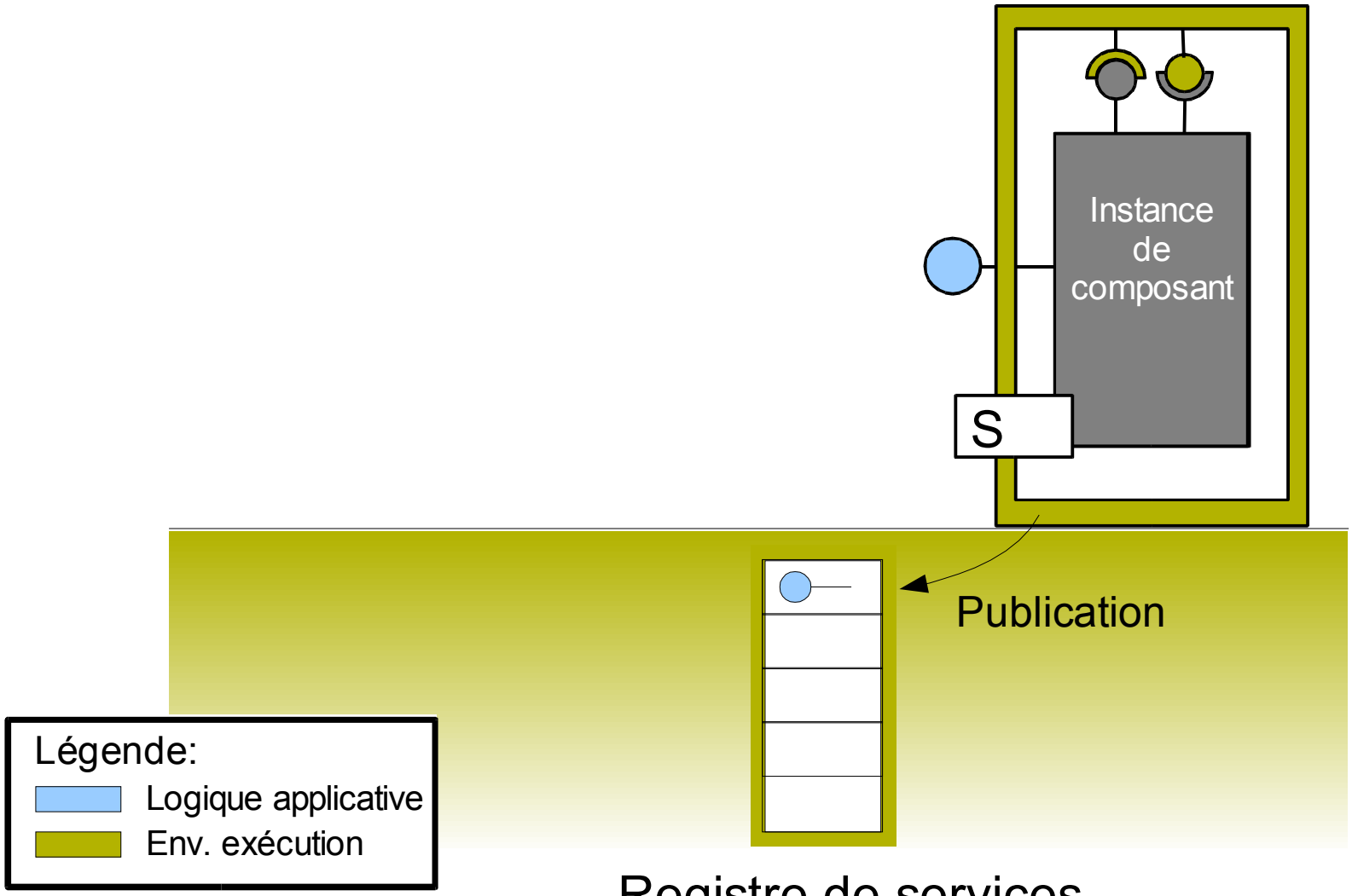




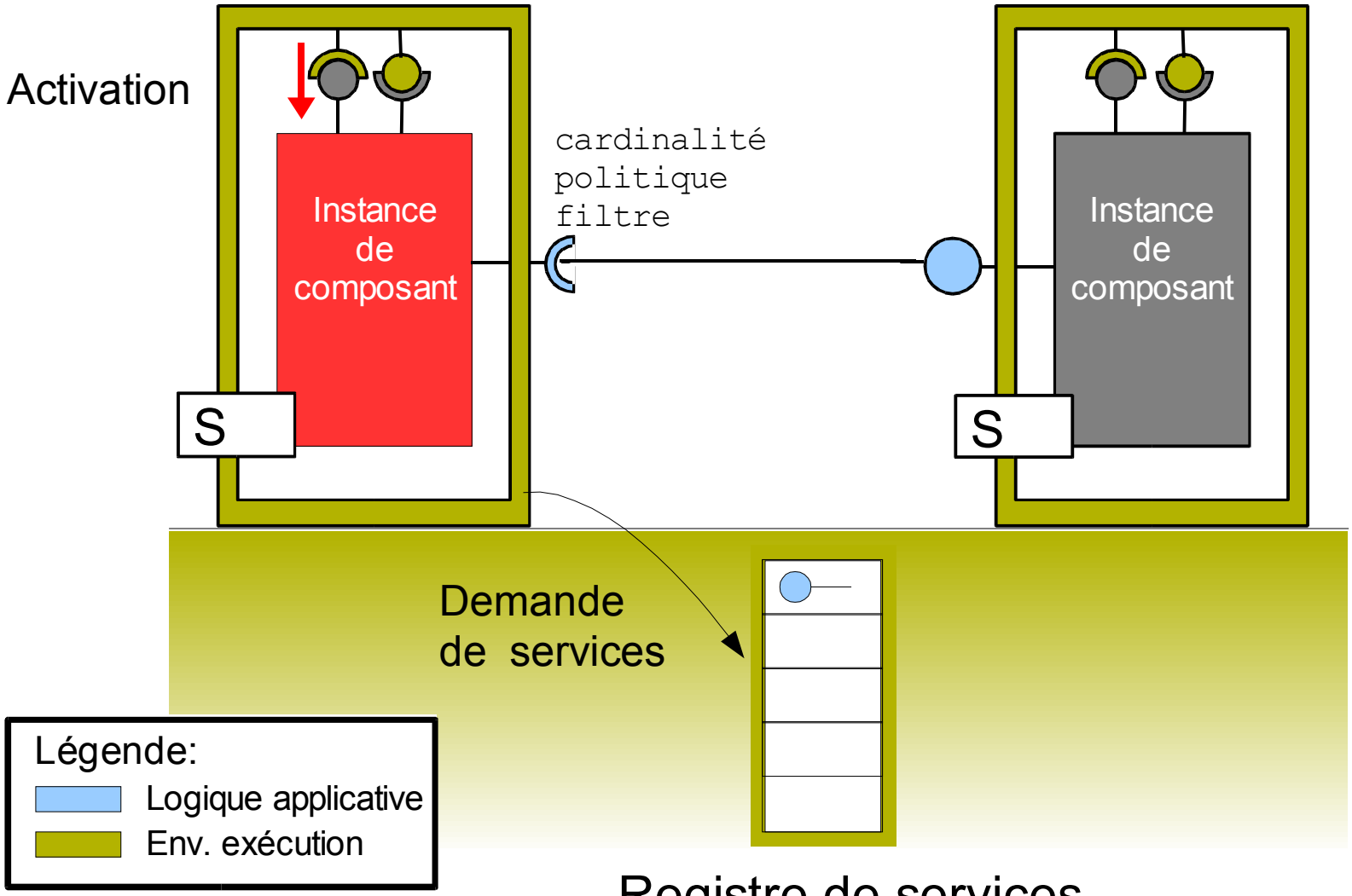
Gestion des instances



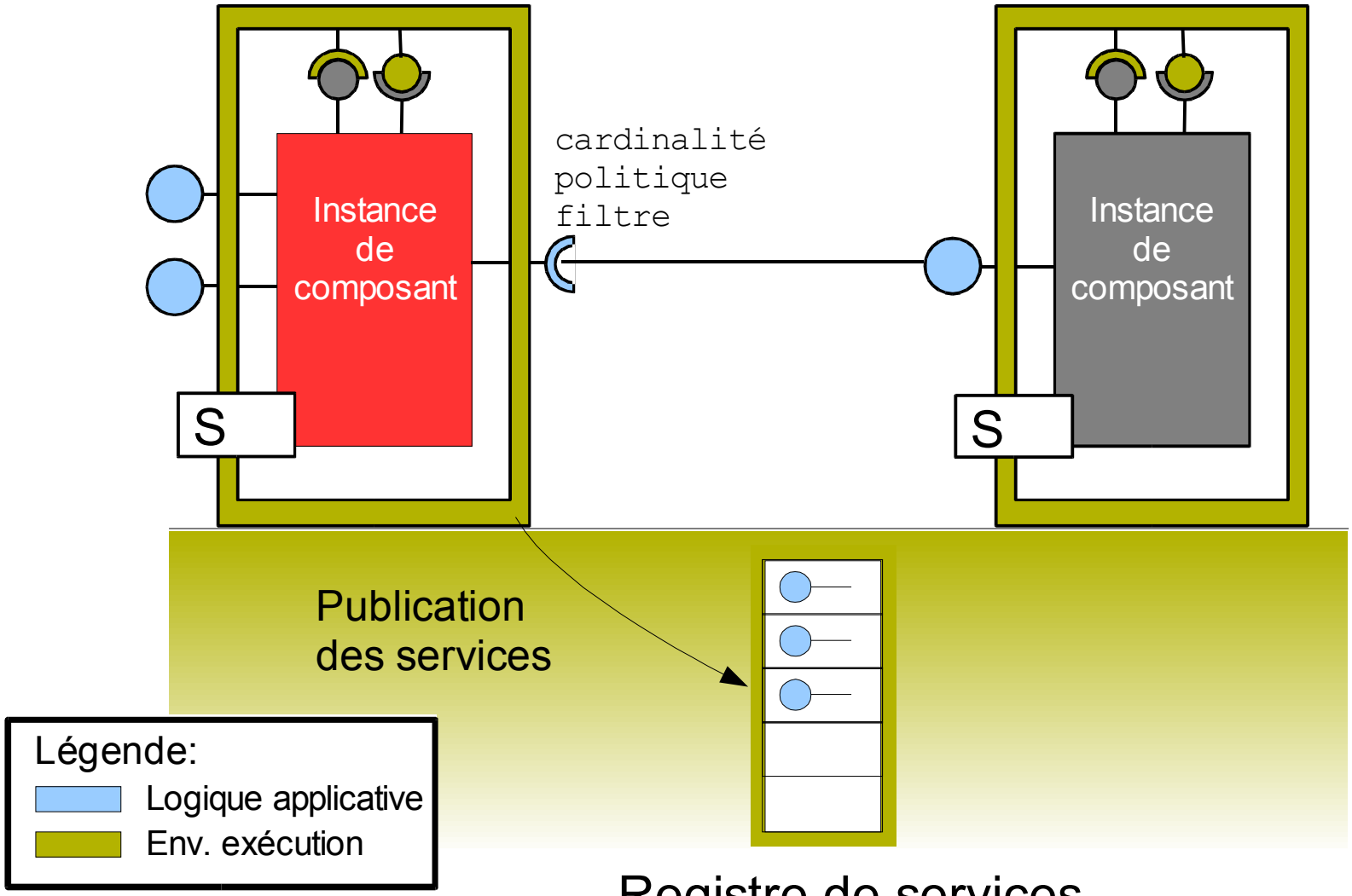
Gestion des instances



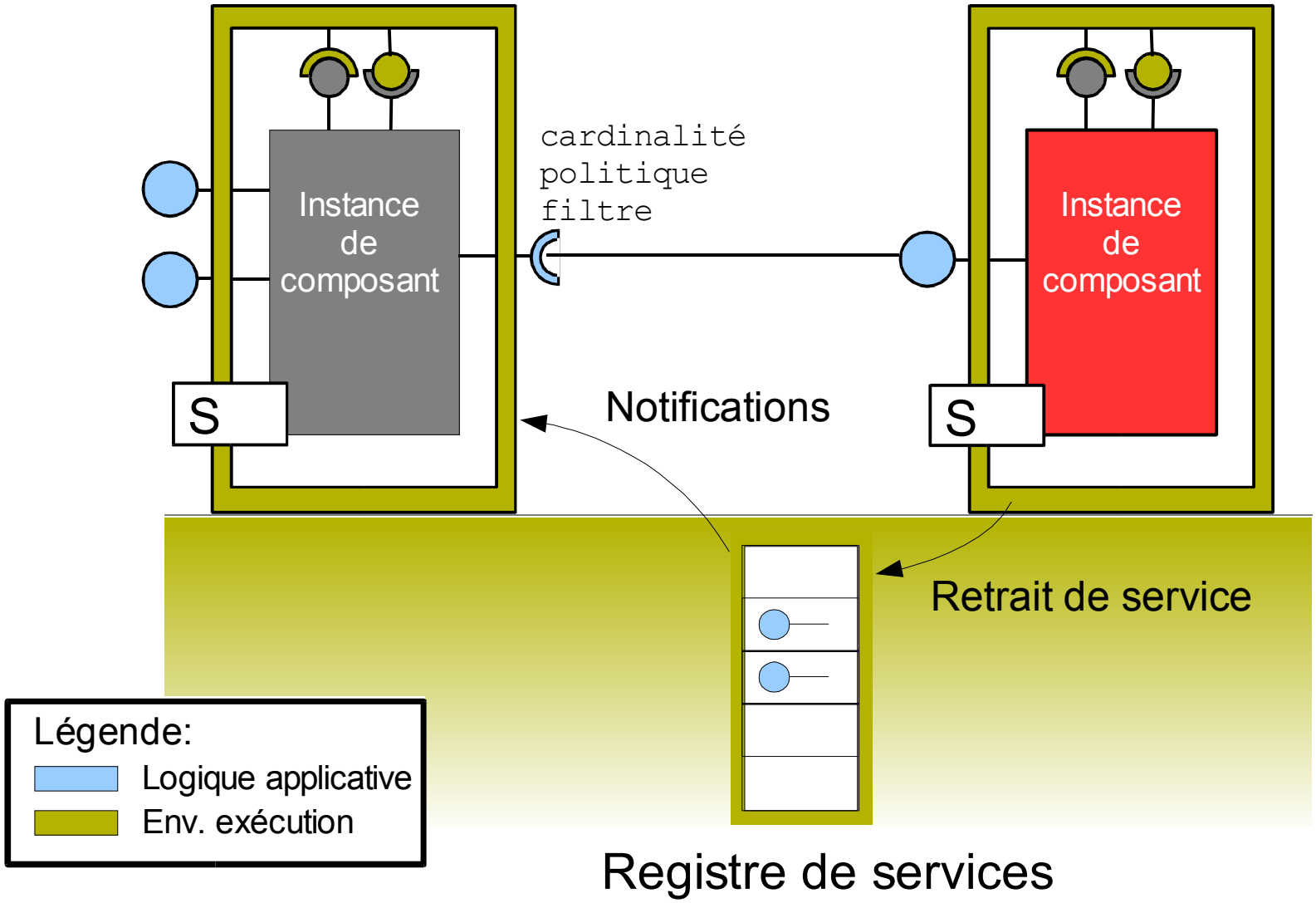
Gestion des instances



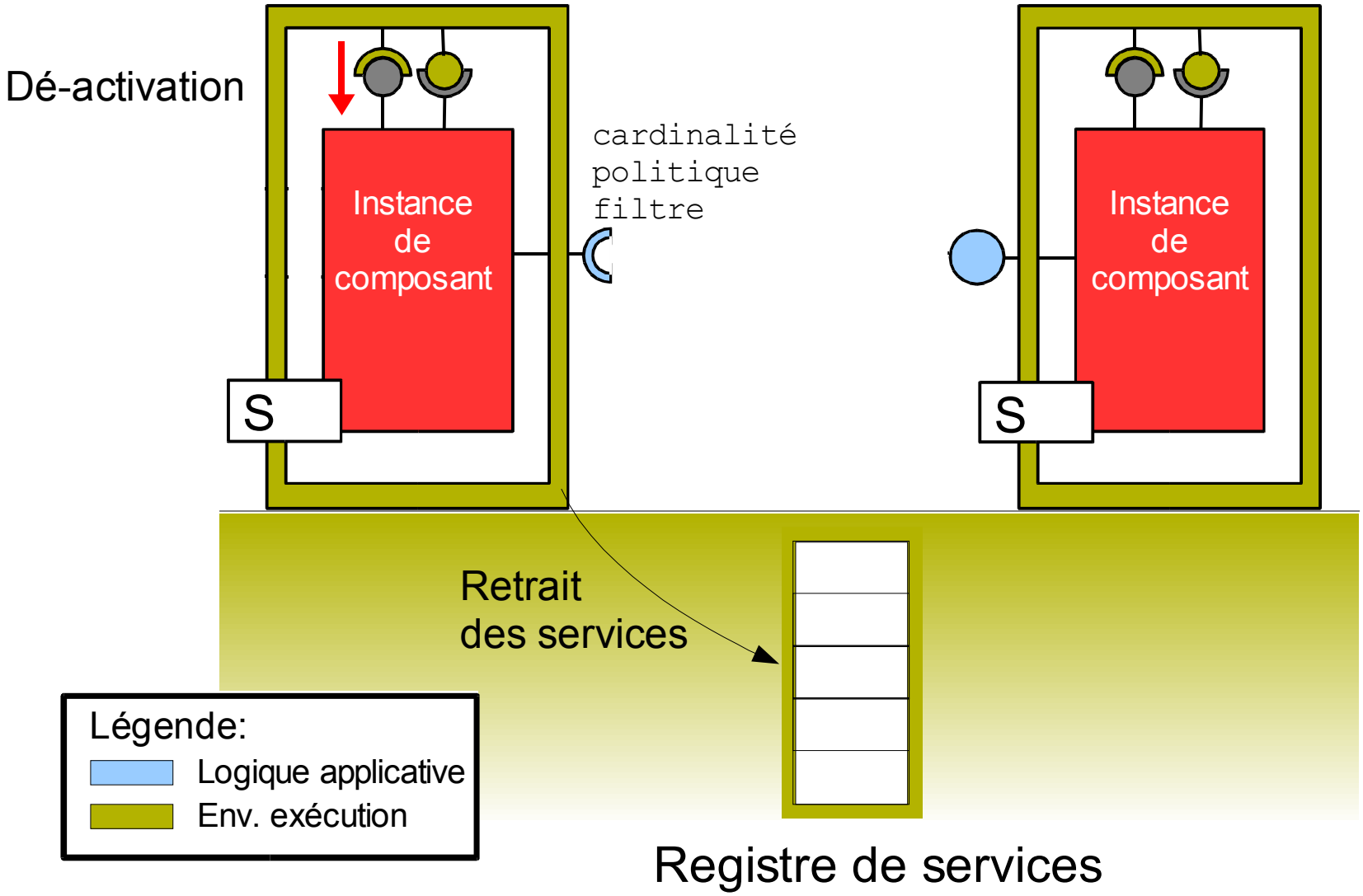
Gestion des instances



Gestion des instances



Gestion des instances



Plan de la thèse

Problématique

Fondations

Modèle à composants orienté services

- Composants à services
- Gestion des instances
- Assemblage et évolution d'une application
- Types d'instances
- Imprévisibilité et espaces de résolution
- Gestion de composition
- Descripteur de composition

Réalisation

Évaluations

Conclusions et perspectives

Applications

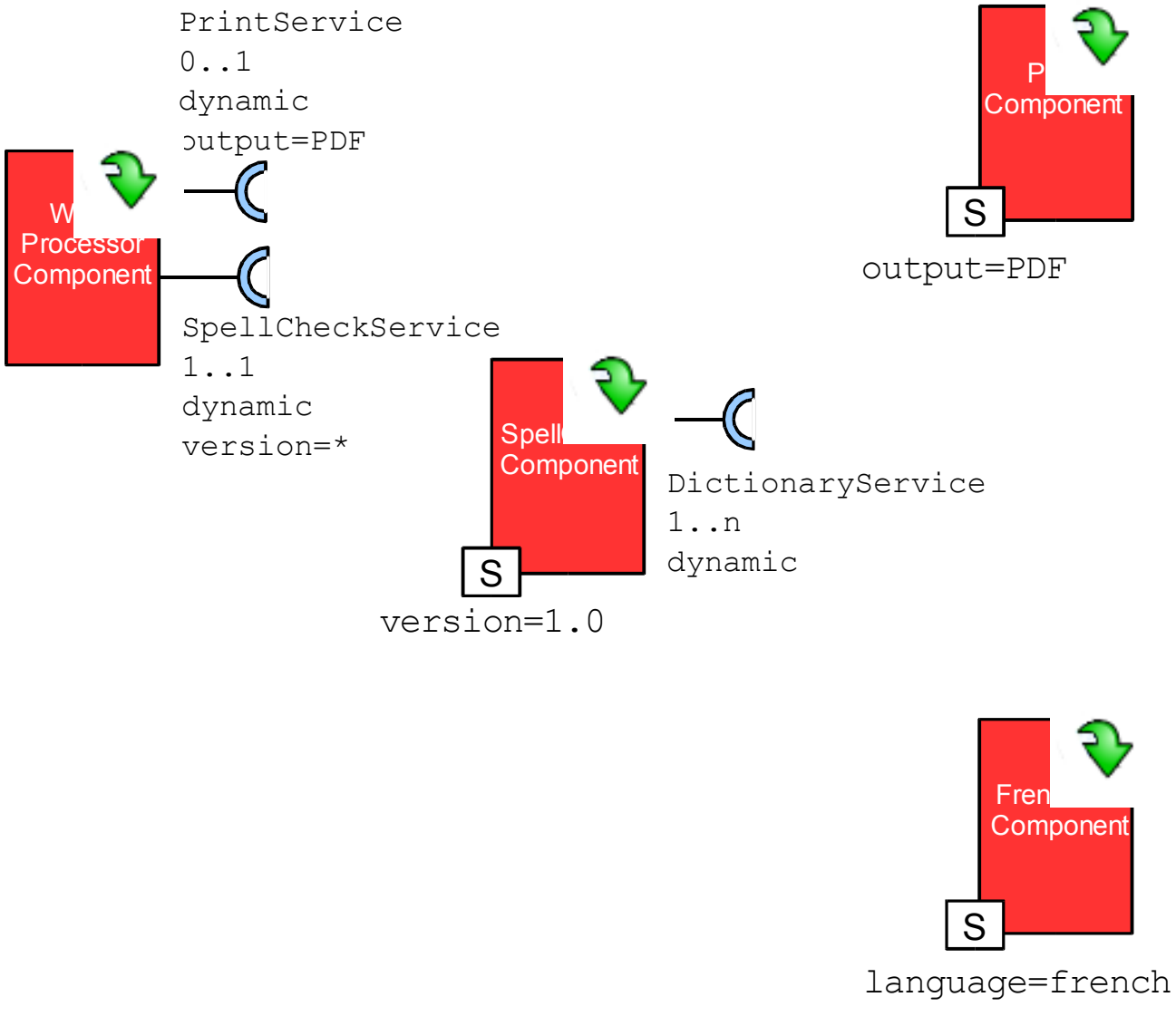
Application

- Créée à partir d'ensemble d'instances introduites dans un environnement d'exécution
- Un "noyau" extensible

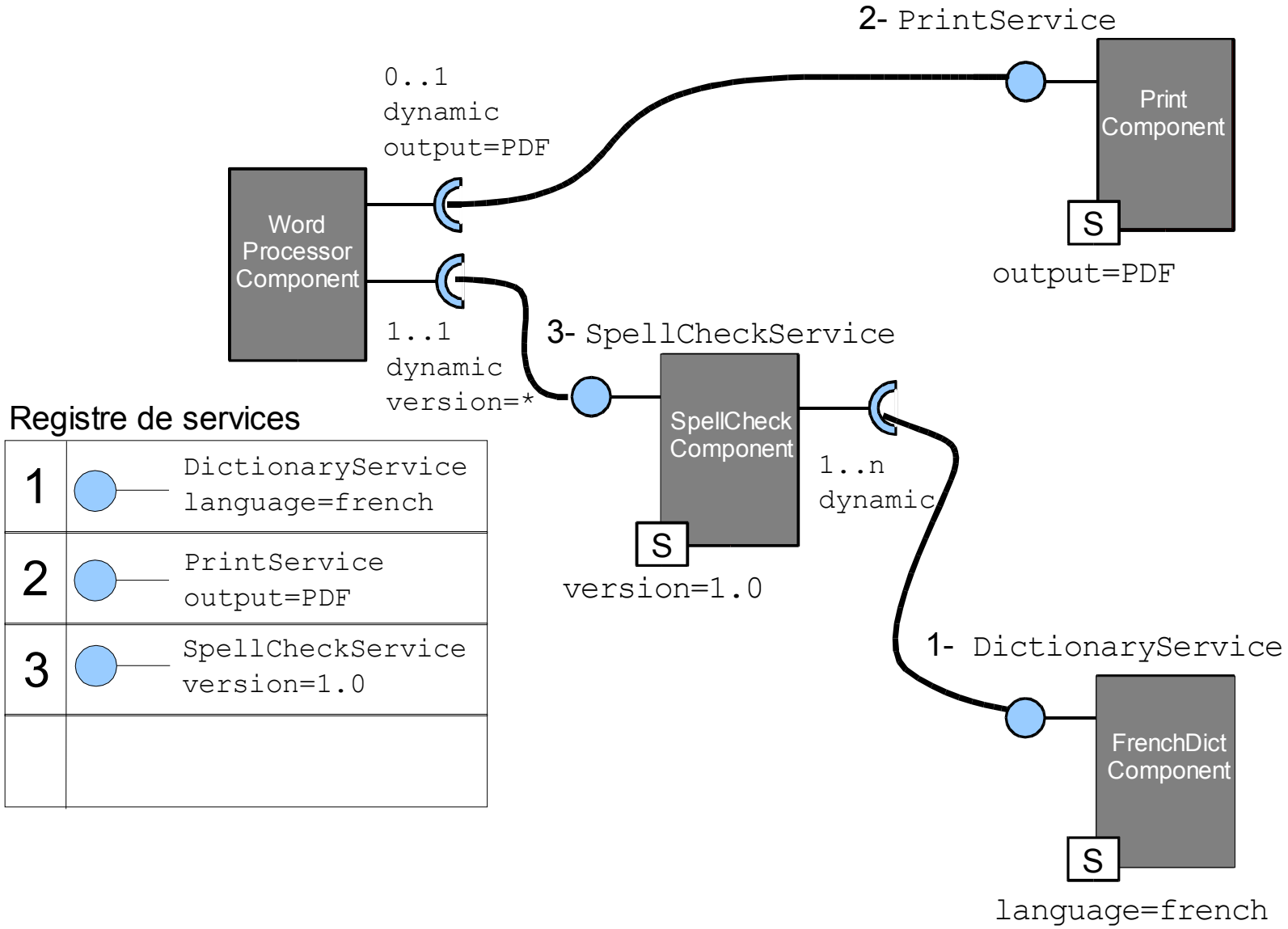
Gestion résulte dans applications capables

- D'auto-assemblage progressif
- D'incorporer des fonctionnalités
- De libérer des fonctionnalités
- De substituer des fonctionnalités

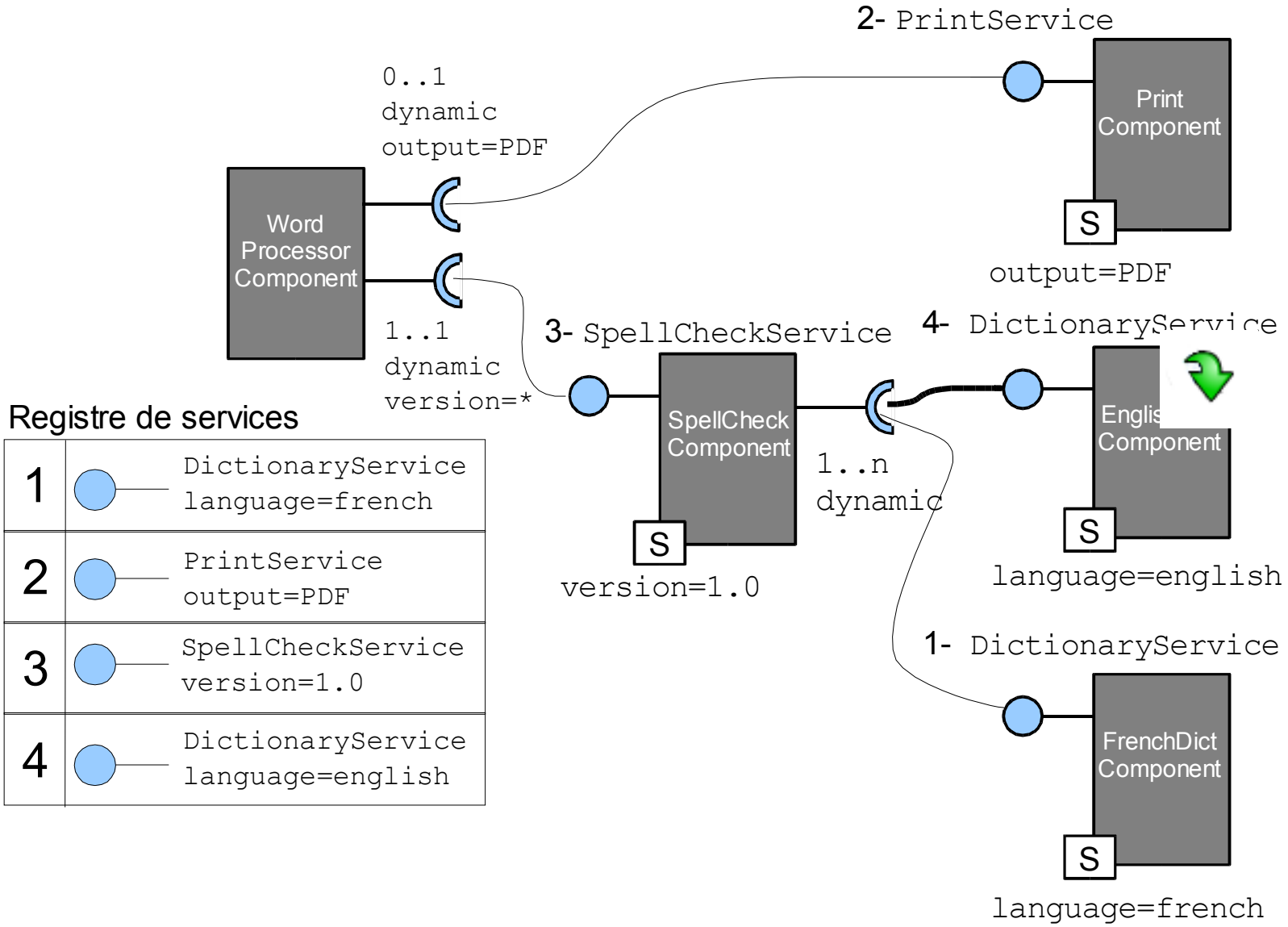
Assemblage


PrintService
0..1
dynamic
output=PDF
W
Processor
Component
SpellCheckService
1..1
dynamic
version=*
P
Component
S
output=PDF
Spell
Component
S
version=1.0
DictionaryService
1..n
dynamic
Fren
Component
S
language=french

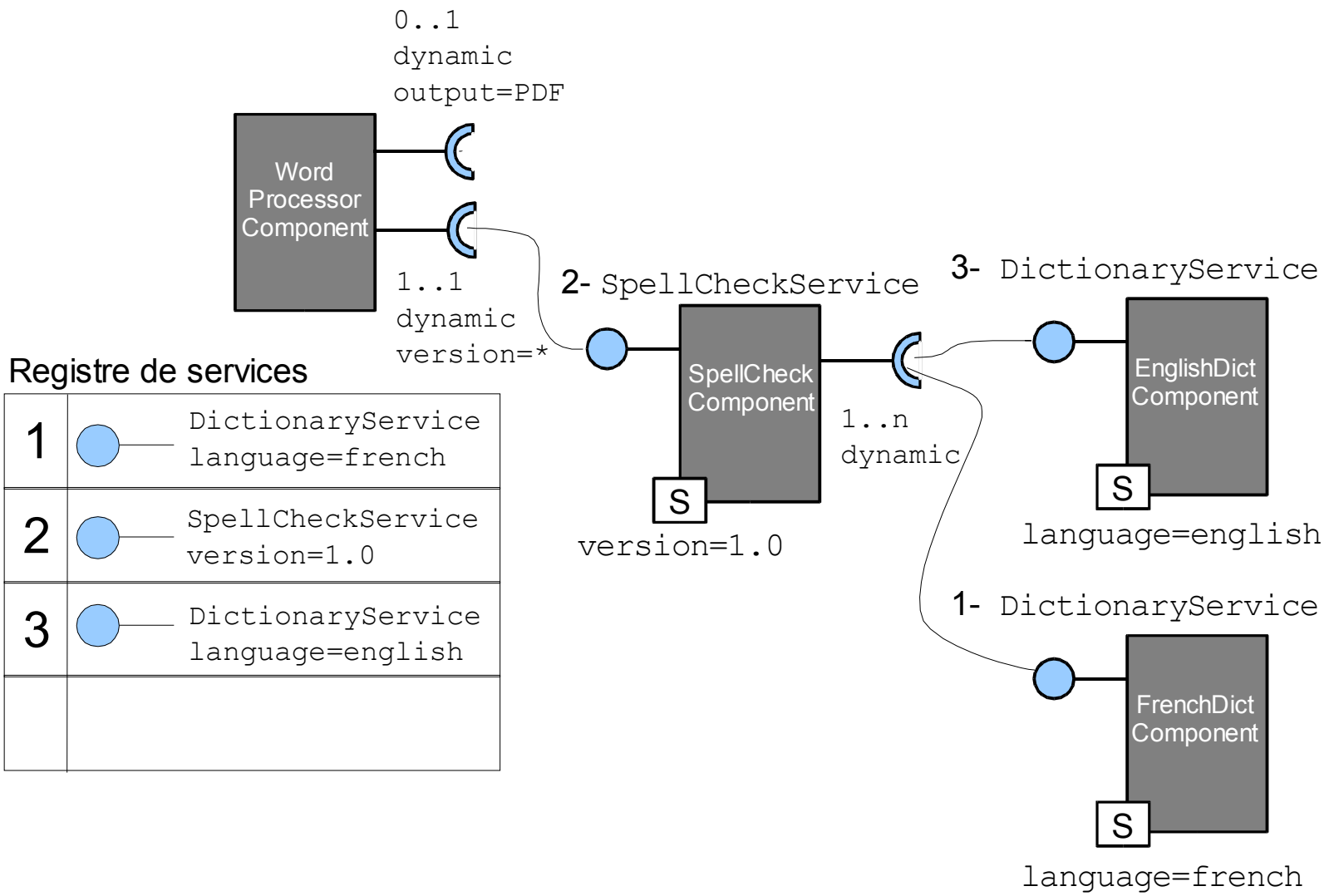
Assemblage


2- PrintService
0..1
dynamic
output=PDF
Print Component
S
output=PDF
Word Processor Component
1..1
dynamic
version=*
3- SpellCheckService
SpellCheck Component
S
version=1.0
1..n
dynamic
1- DictionaryService
FrenchDict Component
S
language=french
Registre de services
1
DictionaryService
language=french
2
PrintService
output=PDF
3
SpellCheckService
version=1.0

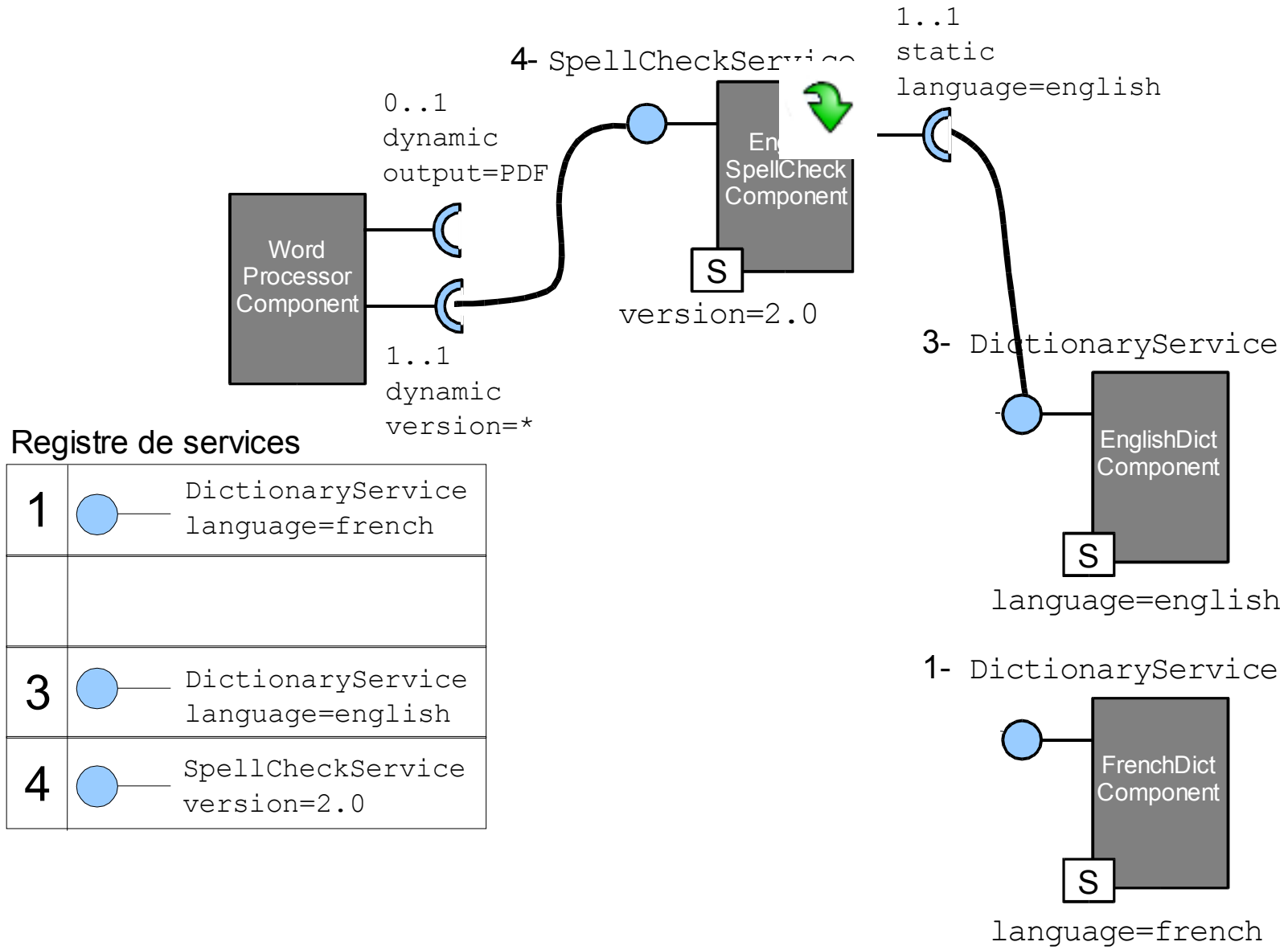
Incorporation de fonctionnalité



Départ de fonctionnalité



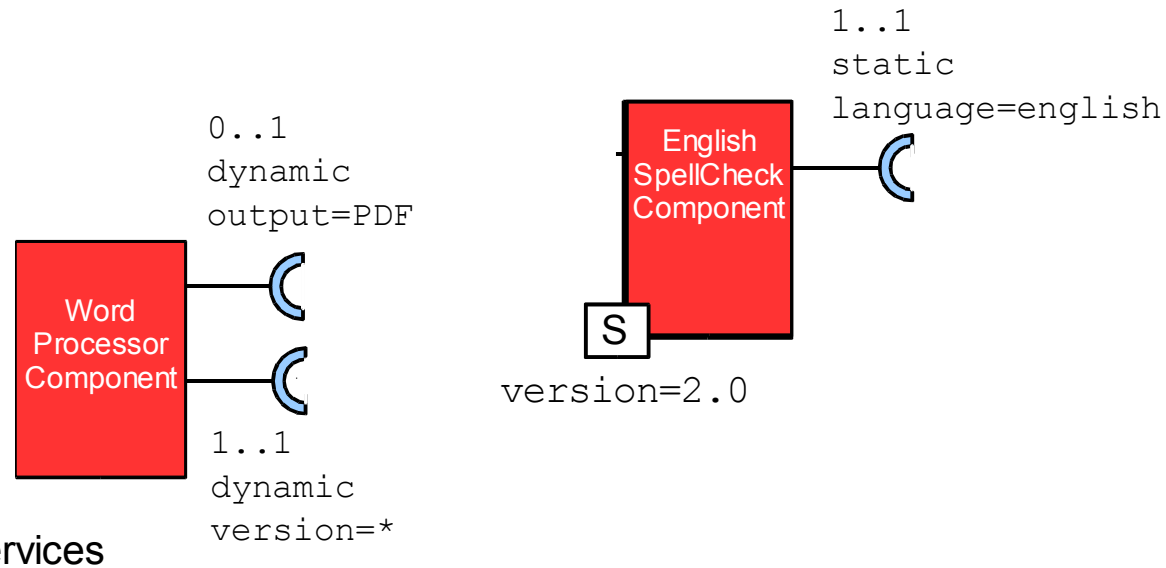
Substitution de fonctionnalité



Registre de services

1	DictionaryService language=french
3	DictionaryService language=english
4	SpellCheckService version=2.0

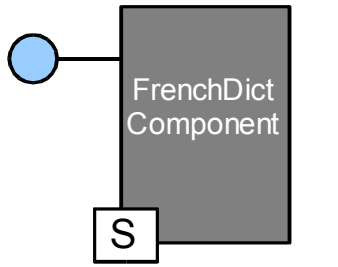
Invalidation « en chaine »



Registre de services

1	DictionaryService language=french

1- DictionaryService



Plan de la thèse

Problématique

Fondations

Modèle à composants orienté services

- Composants à services
- Gestion des instances
- Assemblage et évolution d'une application
- Types d'instances
- Imprévisibilité et espaces de résolution
- Gestion de composition
- Descripteur de composition

Réalisation

Évaluations

Conclusions et perspectives

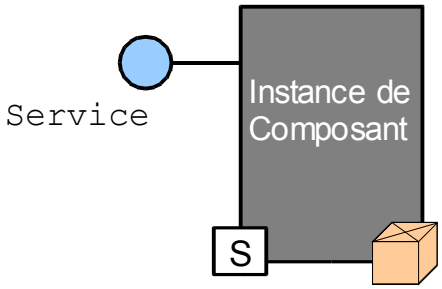
Types d'instances

Instances *de déploiement*

- Instances uniques (singleton)
- Cycle de vie lié aux activités de déploiement

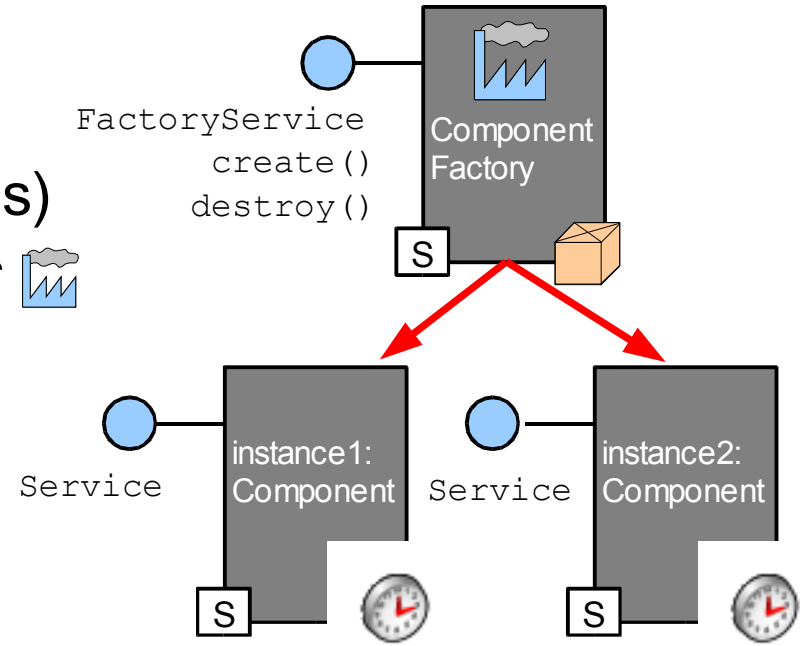
  `create()` = installation

  `destroy()` = retrait

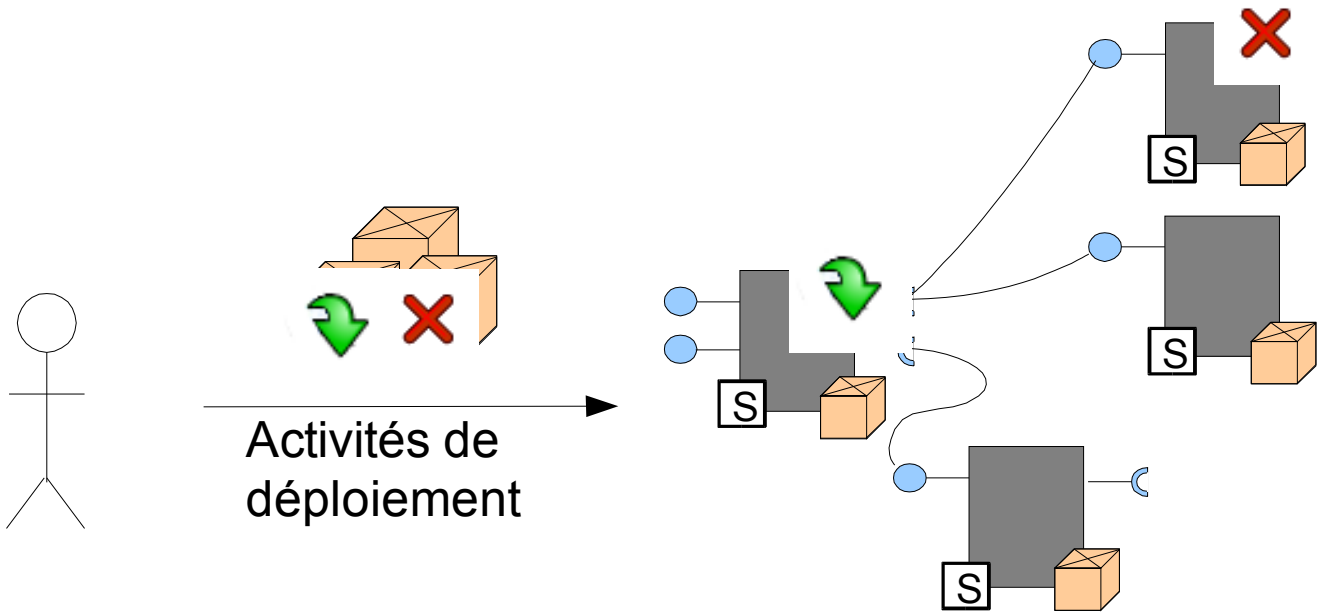


Instances *dynamiques*

- Instances multiples (nommées)
- Création à partir de *fabriques*
  - Instances de déploiement
  - Disponibilité dynamique



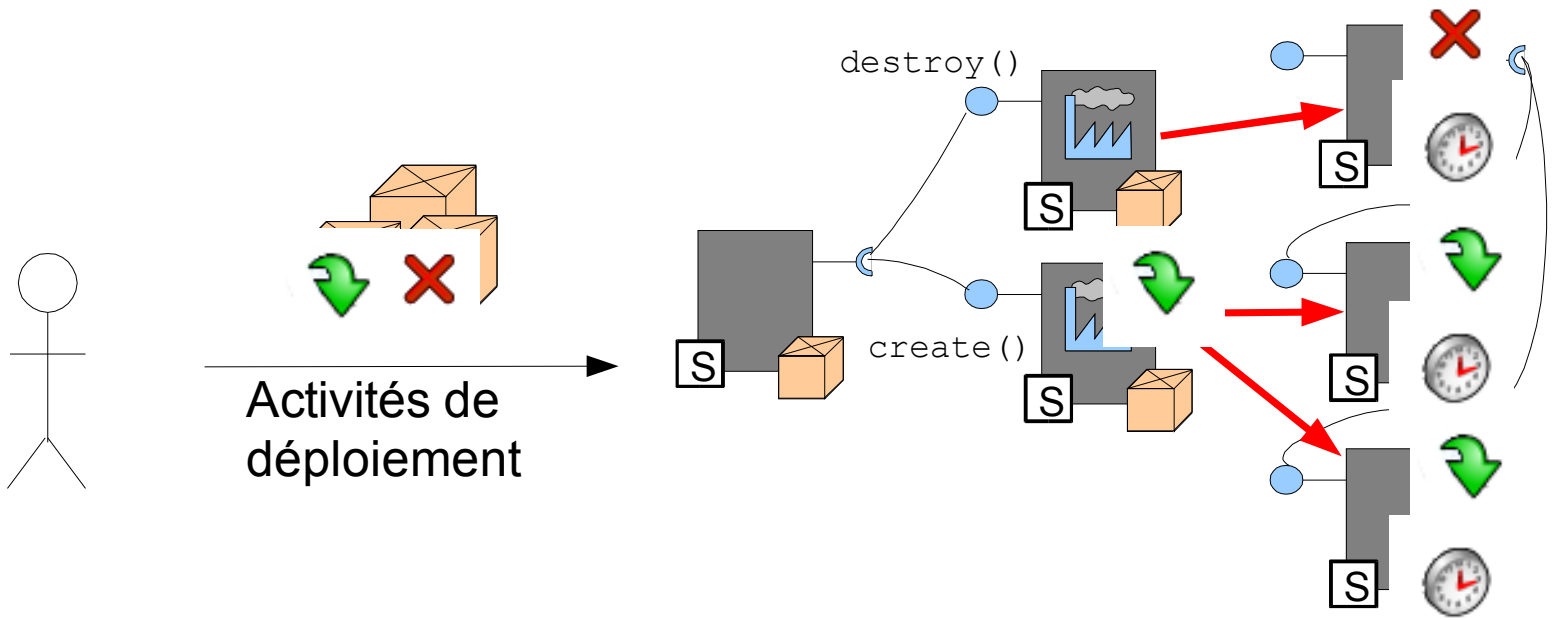
Apps. avec instances de déploiement



Assemblage et évolution d'une application

- Par rapport aux activités de déploiement

Applications avec instances dynamiques



Assemblage et évolution d'une application

- Par rapport aux activités de déploiement
- Par rapport aux interactions avec les fabriques à l'intérieur du système

Plan de la thèse

Problématique

Fondations

Modèle à composants orienté services

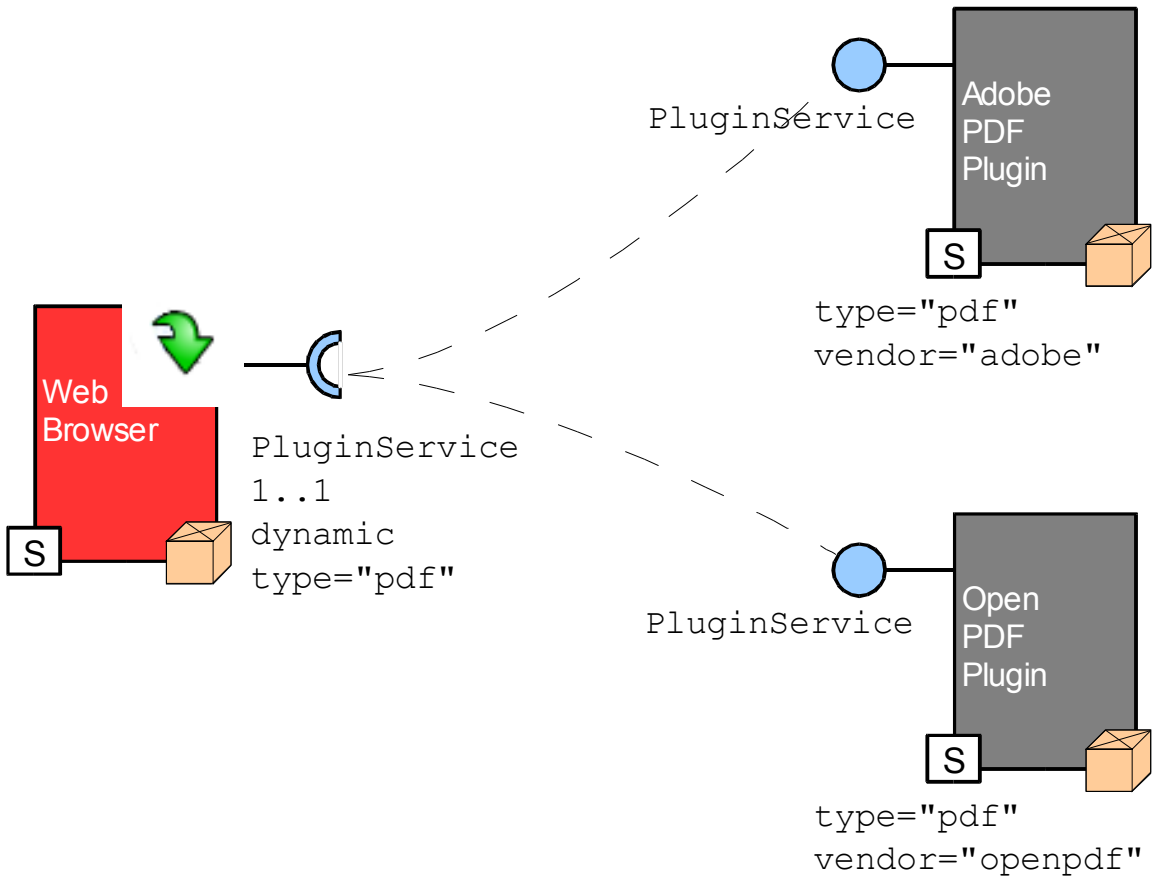
- Composants à services
- Gestion des instances
- Assemblage et évolution d'une application
- Types d'instances
- Imprévisibilité et espaces de résolution
- Gestion de composition
- Descripteur de composition

Réalisation

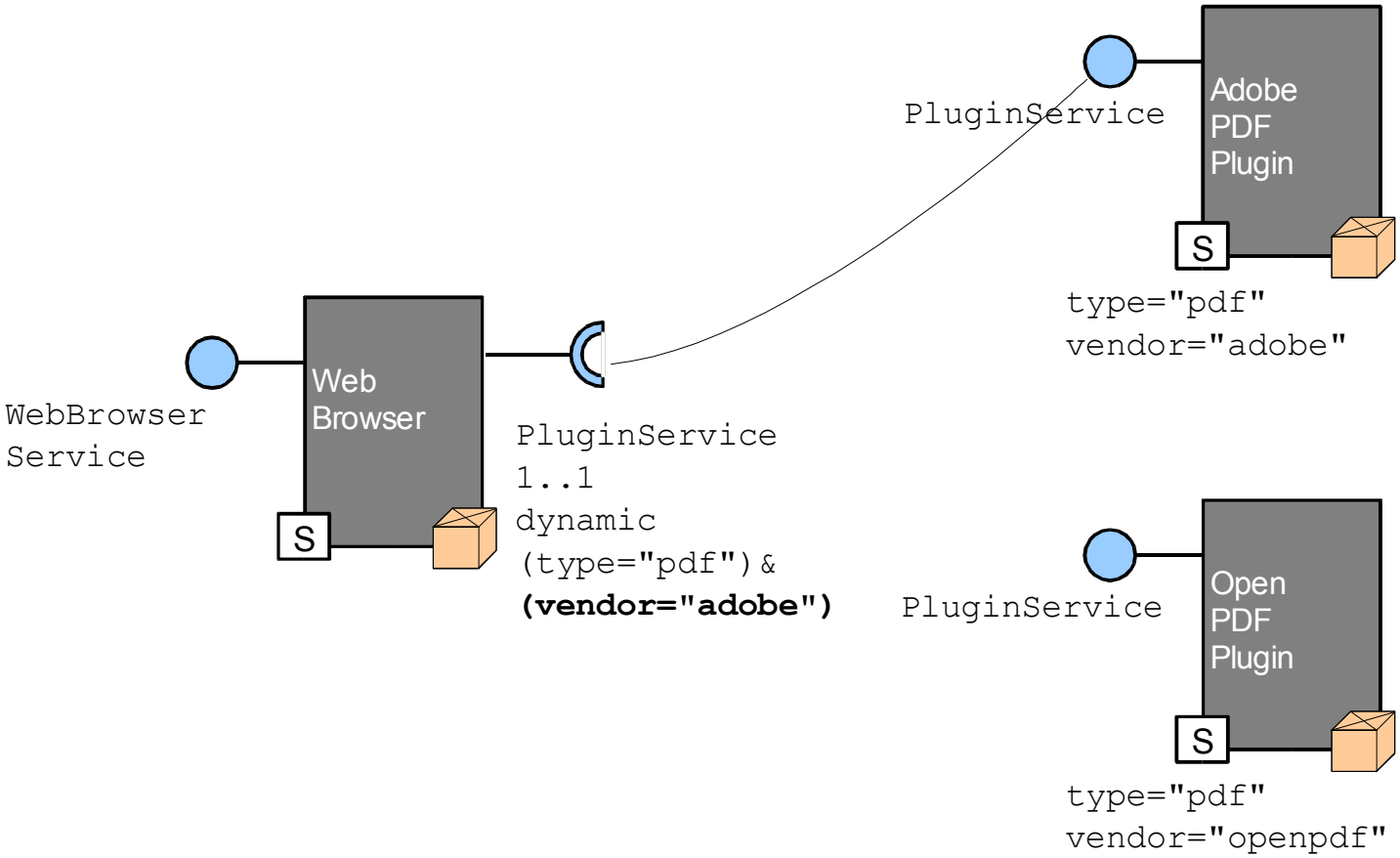
Évaluations

Conclusions et perspectives

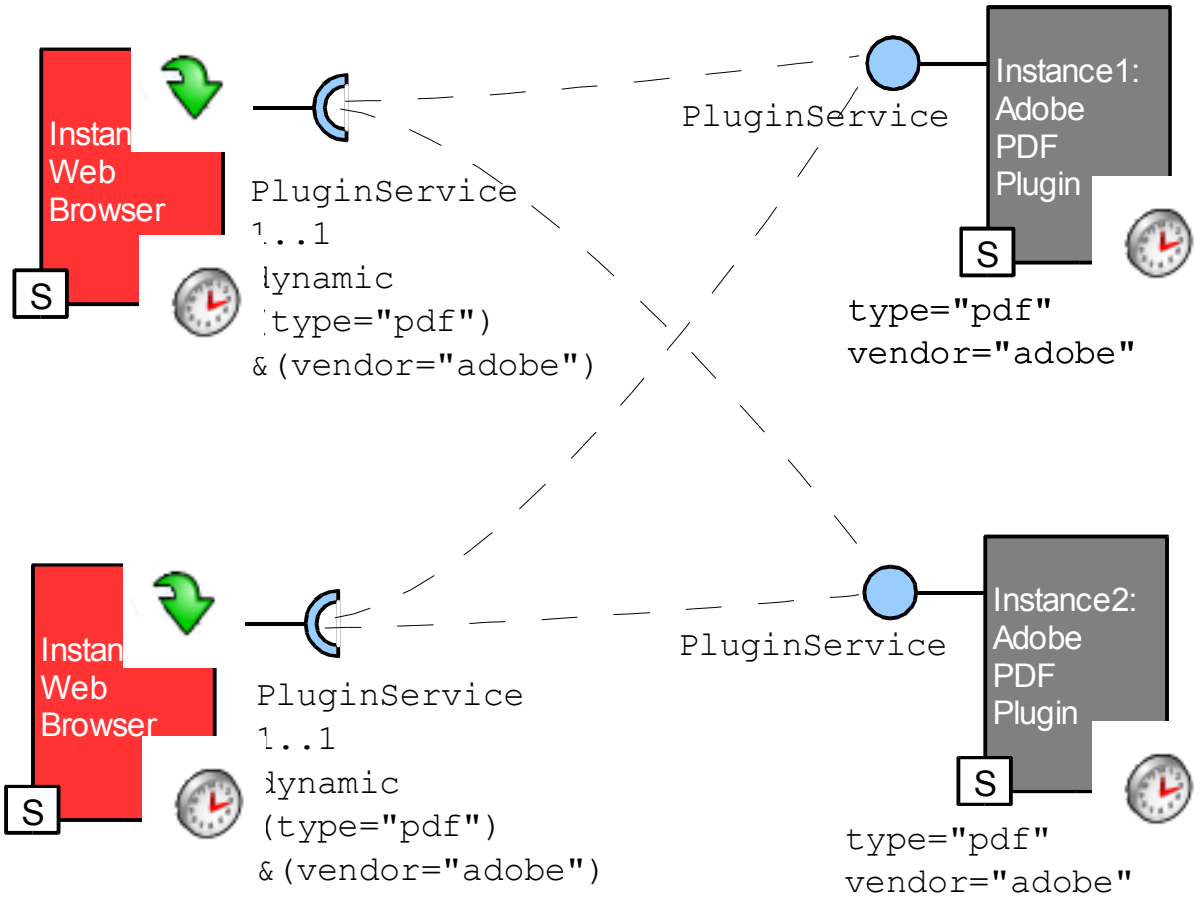
Imprévisibilité



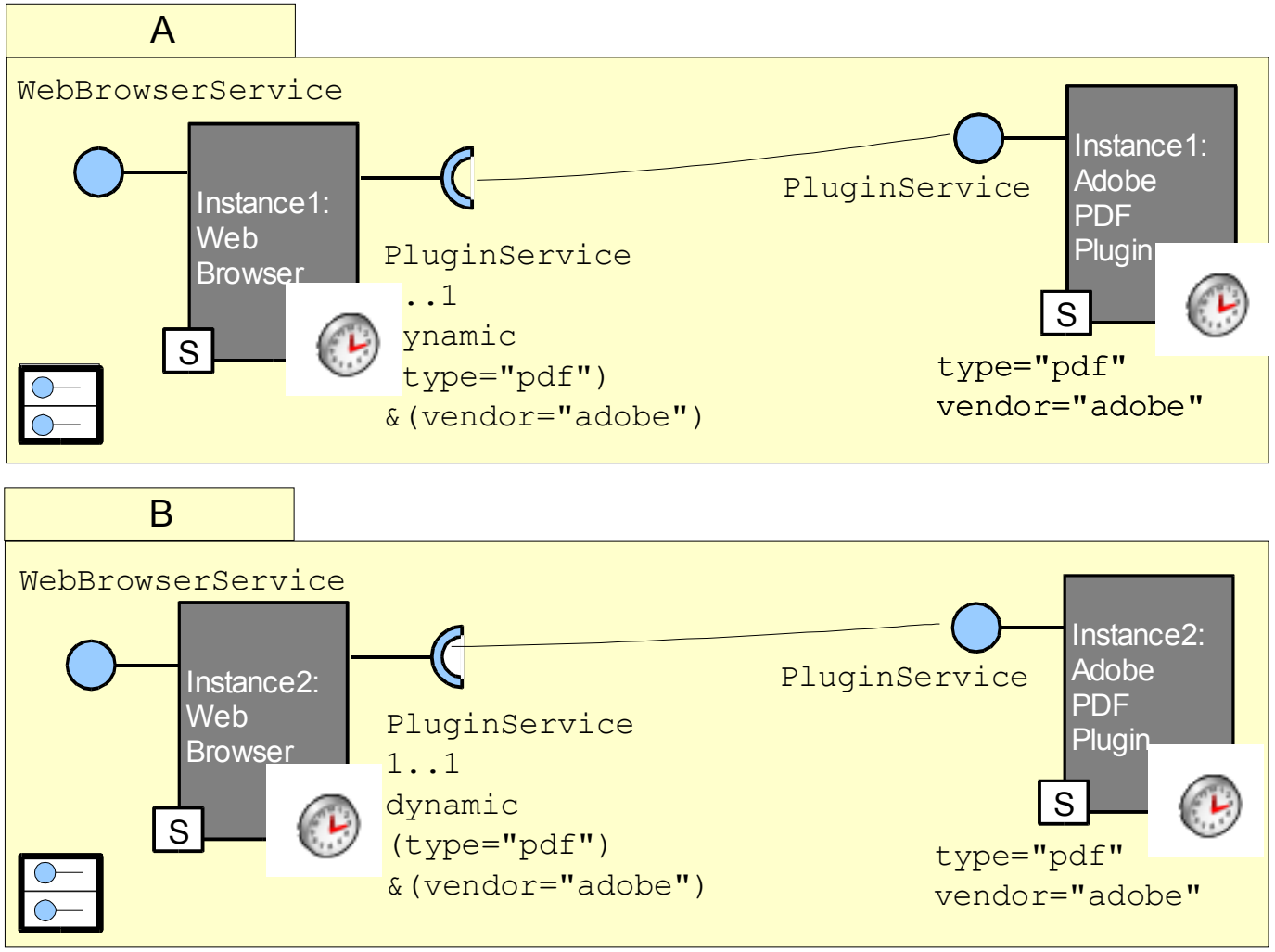
Imprévisibilité et instances déploiement



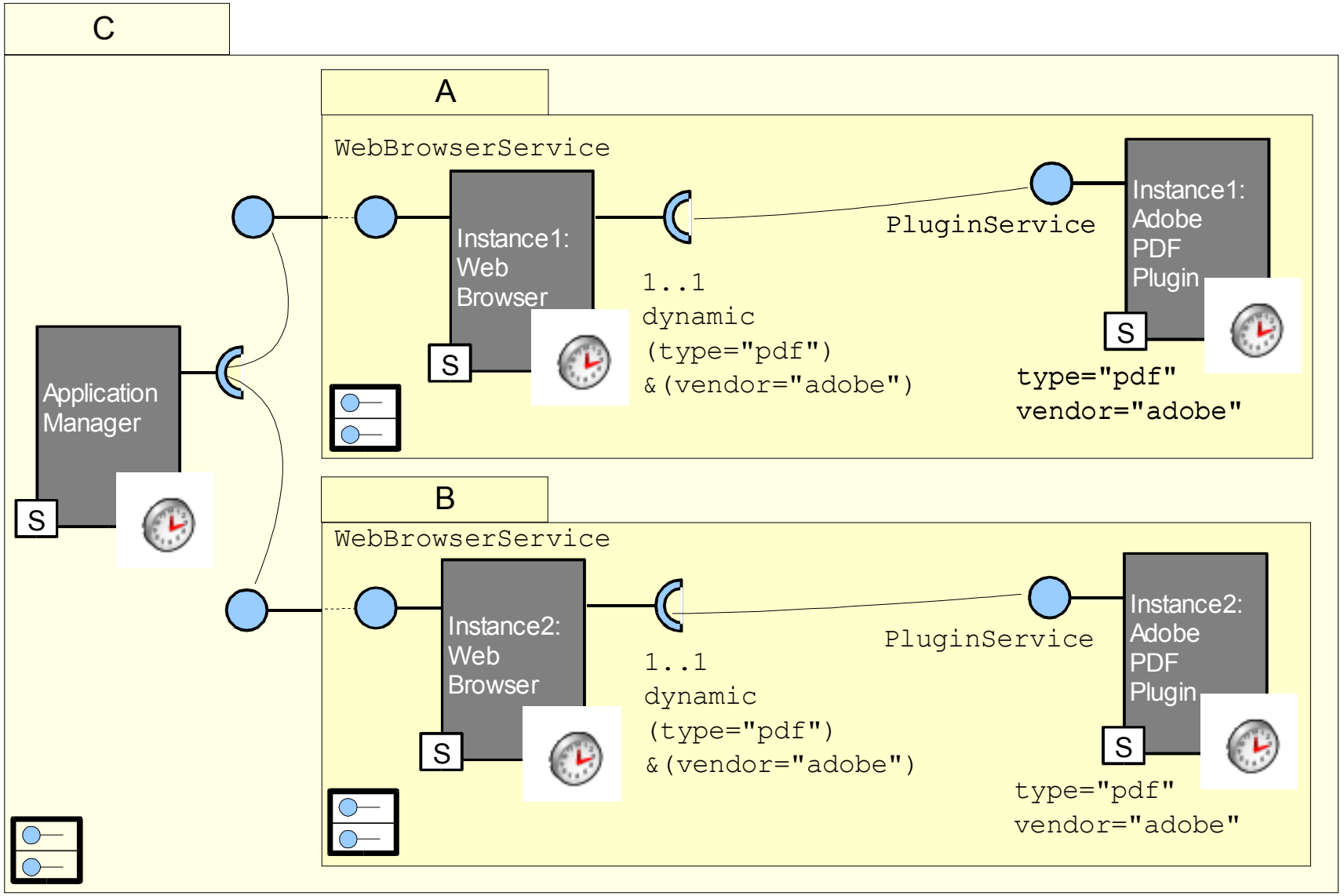
Imprévisibilité et instances dynamiques



Espaces de résolution



Espaces de résolution



Plan de la thèse

Problématique

Fondations

Modèle à composants orienté services

- Composants à services
- Gestion des instances
- Assemblage et évolution d'une application
- Types d'instances
- Imprévisibilité et espaces de résolution
- Gestion de composition
- Descripteur de composition

Réalisation

Évaluations

Conclusions et perspectives

Composition d'instances

Composition

- Ensemble instances dans un espace de résolution
- Peut être hiérarchique

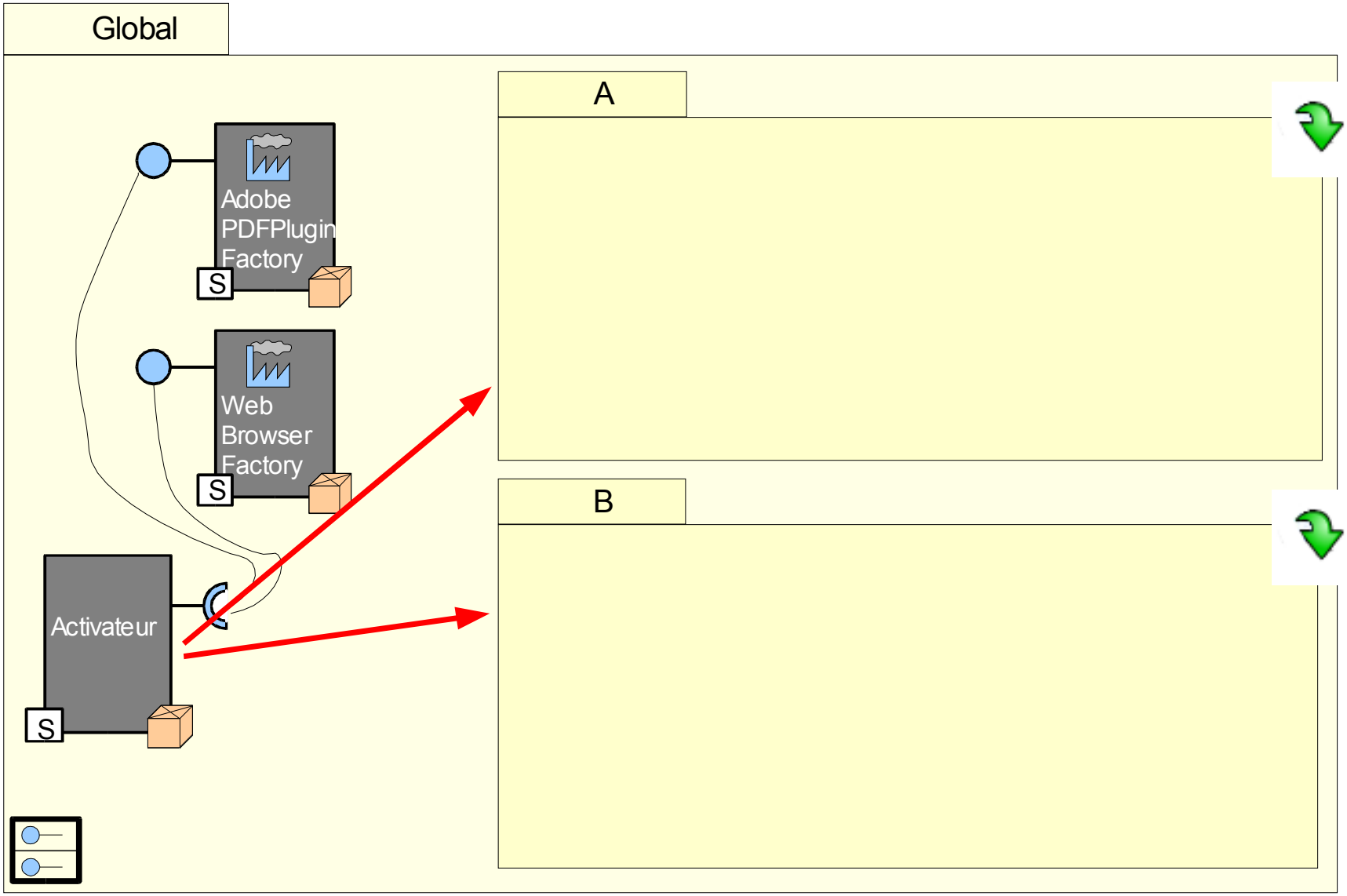
Création de composition

- Liaison avec fabriques
- Création d'espaces de résolution
- Création des instances

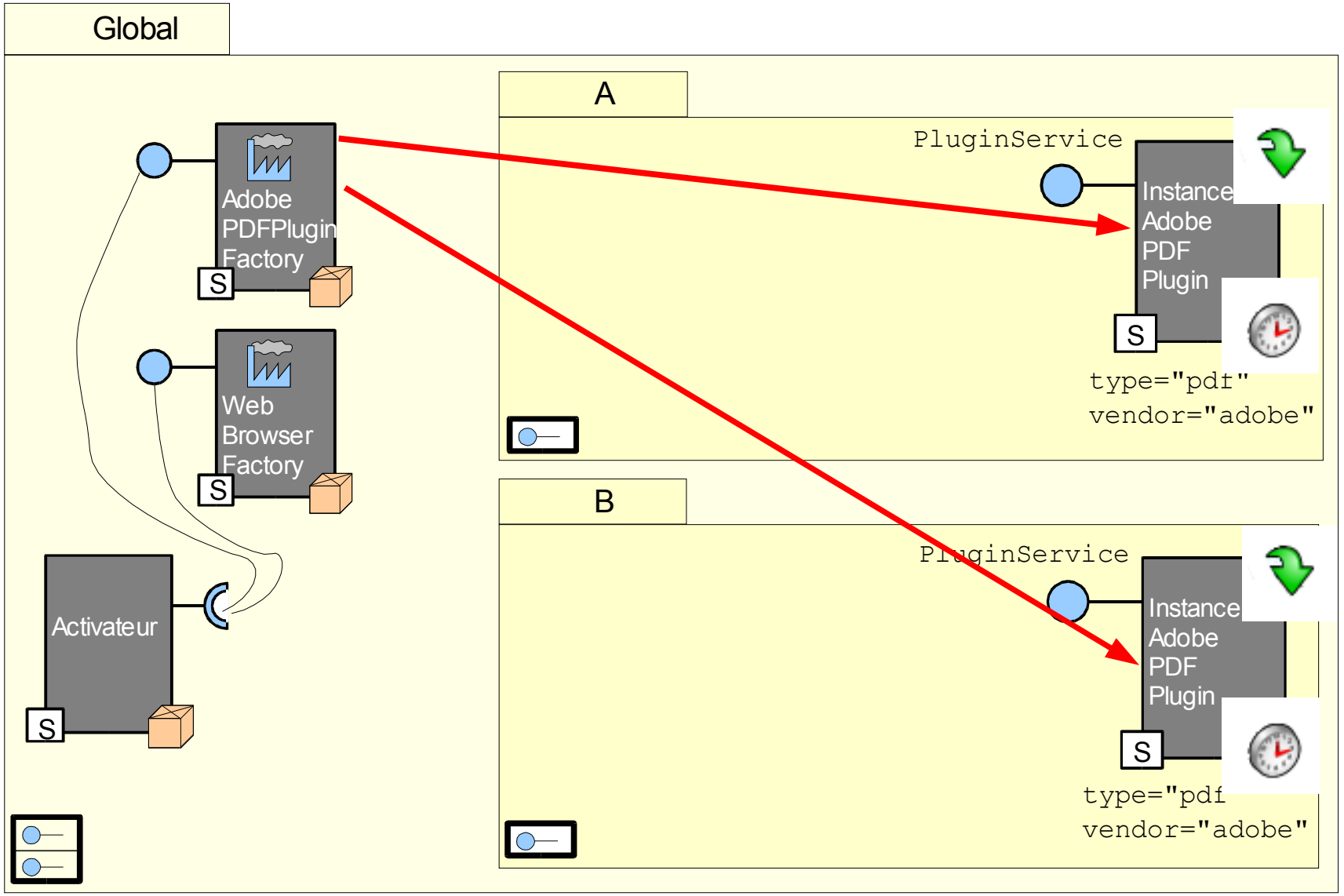
Gestion à l'exécution

- Arrivée et départ de fabriques

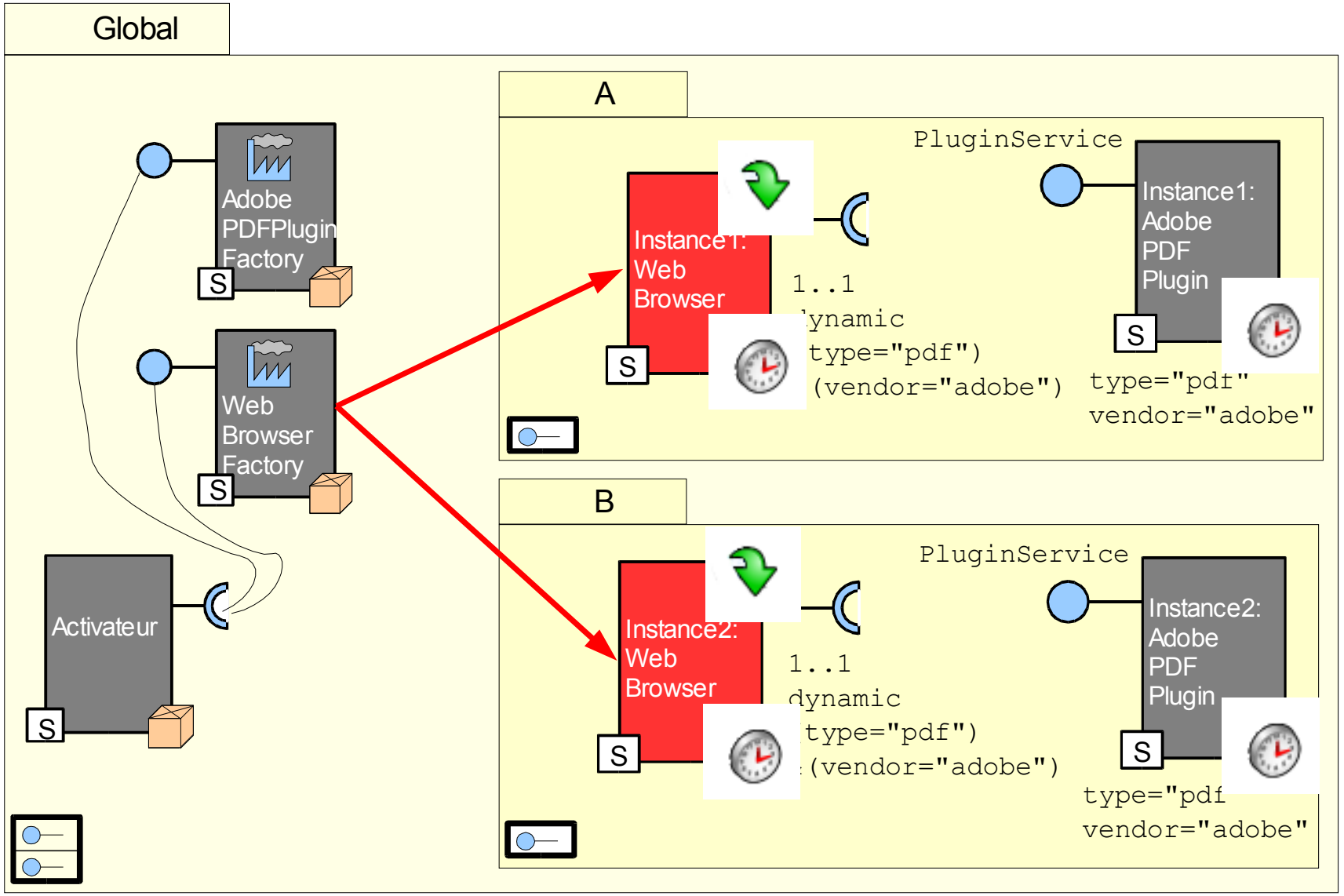
Création de composition



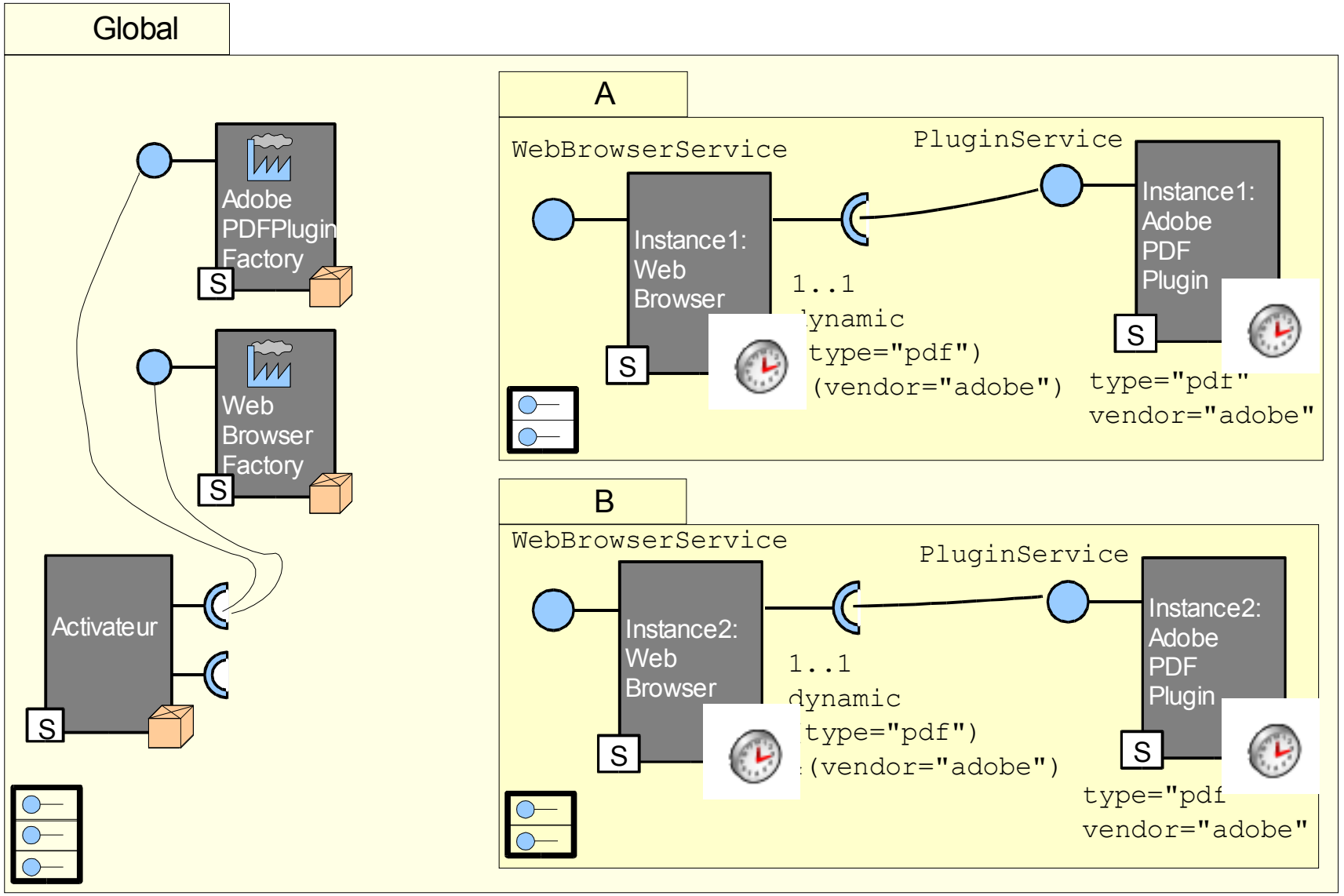
Création de composition



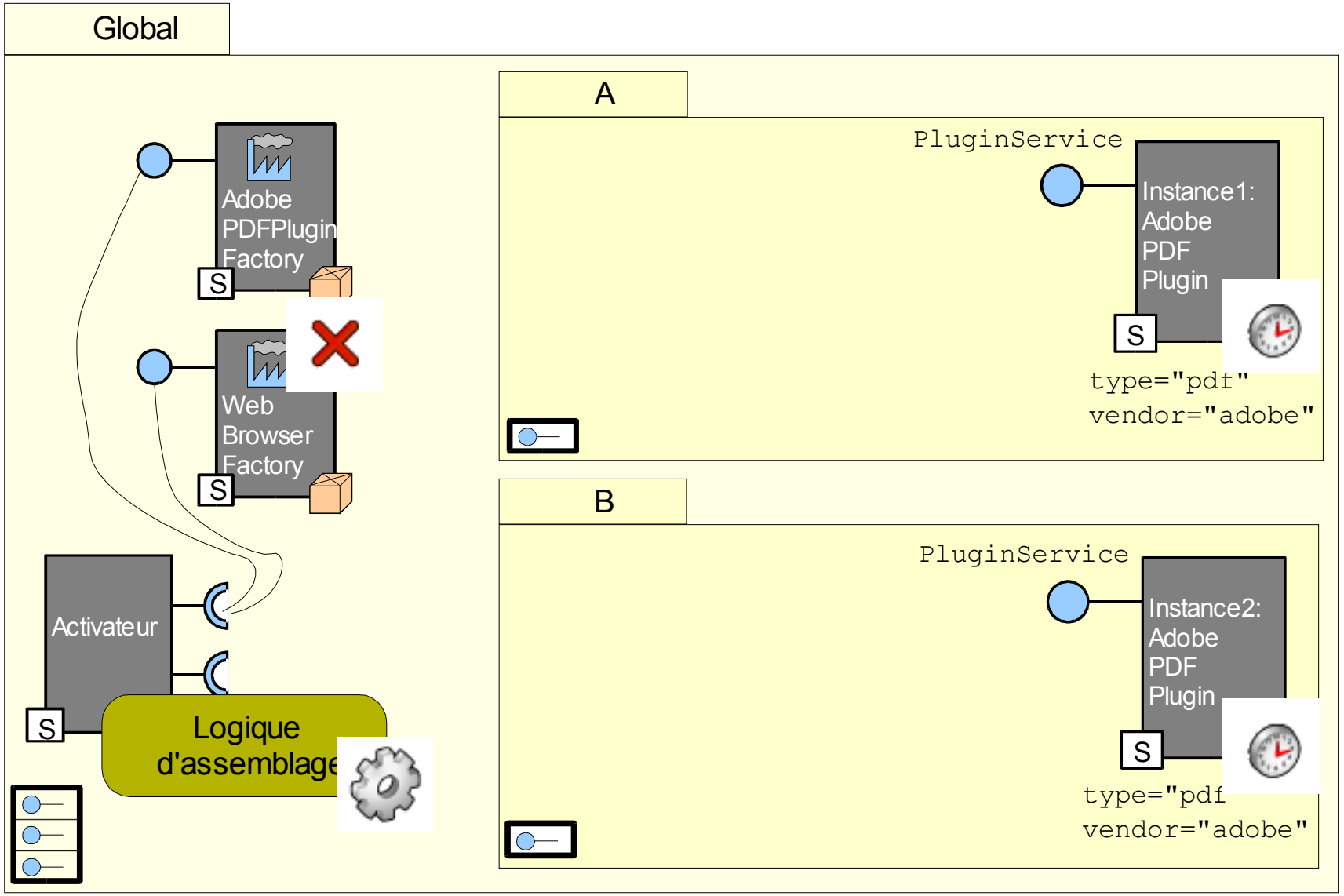
Création de composition



Création de composition



Création de composition



Plan de la thèse

Problématique

Fondations

Modèle à composants orienté services

- Composants à services
- Gestion des instances
- Assemblage et évolution d'une application
- Types d'instances
- Imprévisibilité et espaces de résolution
- Gestion de composition
- Descripteur de composition

Réalisation

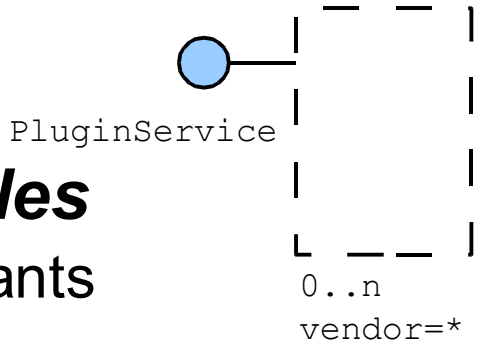
Évaluations

Conclusions et perspectives

Descripteur de composition

Décrit une composition de façon abstraite

- Extraire logique d'assemblage



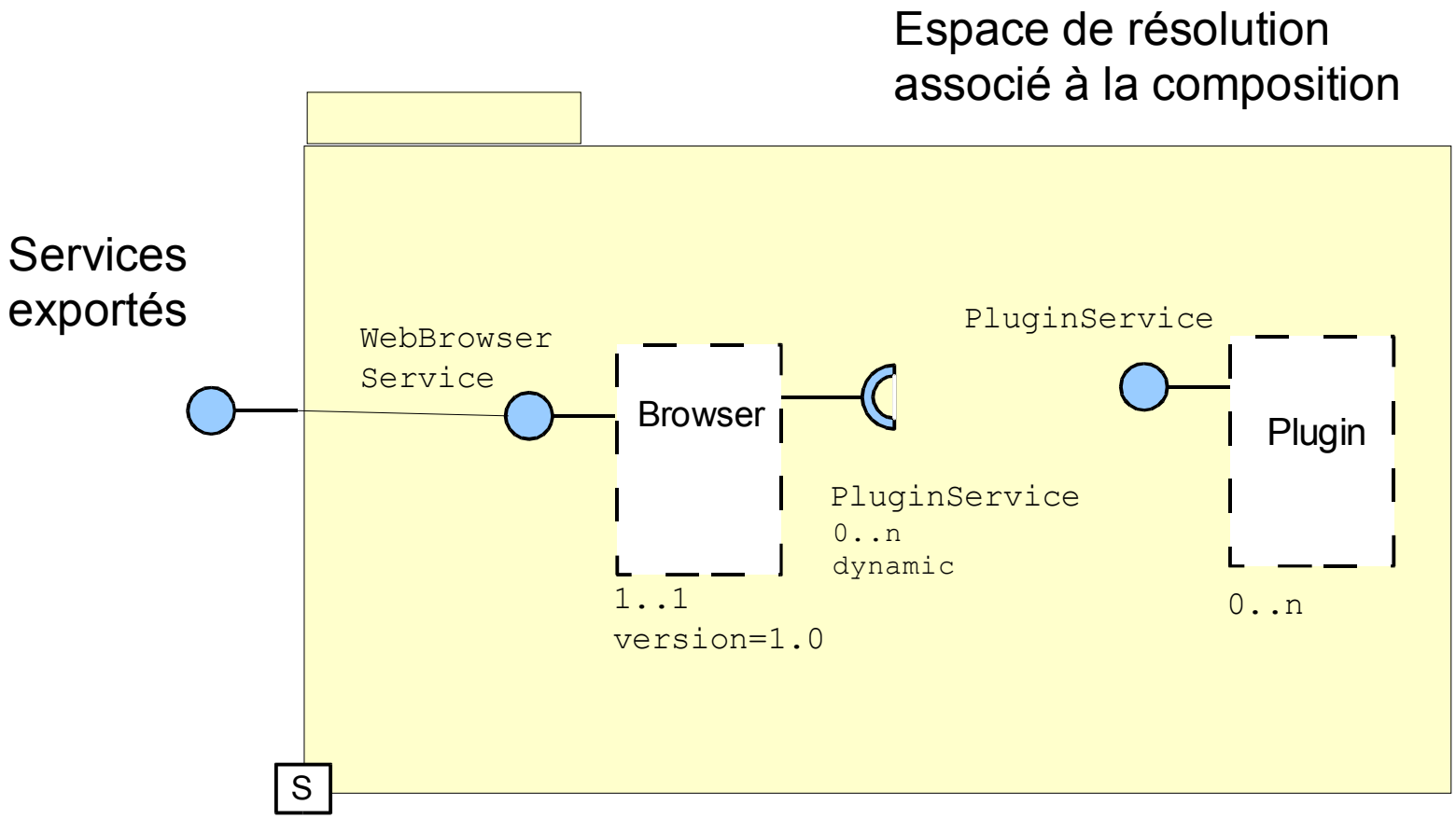
Composition décrite en terme de *socles*

- Nombre variable d'instances de composants
- Vue externe d'un composant à services
- Connexions non-explicites

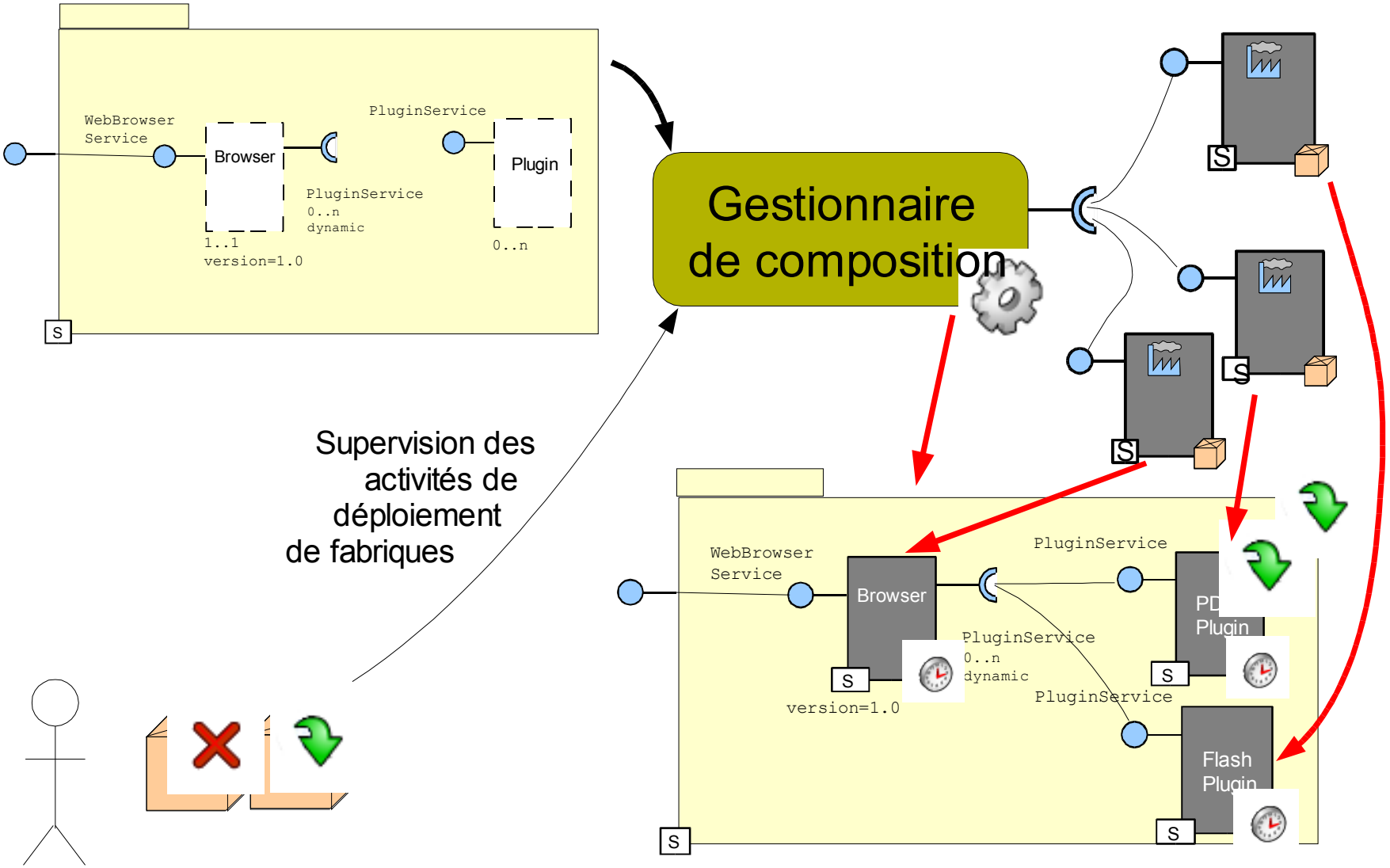
Intégration à l'exécution

- Informations sur vue externe associées aux fabriques
- Tout composant à service qui implémente vue externe est candidat à être intégré
- Services fournis additionnels permis
- Service requis additionnels permis si optionnels

Descripteur de composition



Gestionnaire de composition



Plan de la thèse

Problématique
Fondations
Modèle à composants orienté services
Réalisation

- Service Binder
- Composition Manager
- Service d'introspection

Évaluations

Conclusions et perspectives

Contraintes

Description de services uniquement syntaxique

- Interfaces Java

Pas de gestion du transfert d'état

- Lors de substitutions
- Pas de traitement des propriétés de configuration

Réalisation au dessus de OSGi

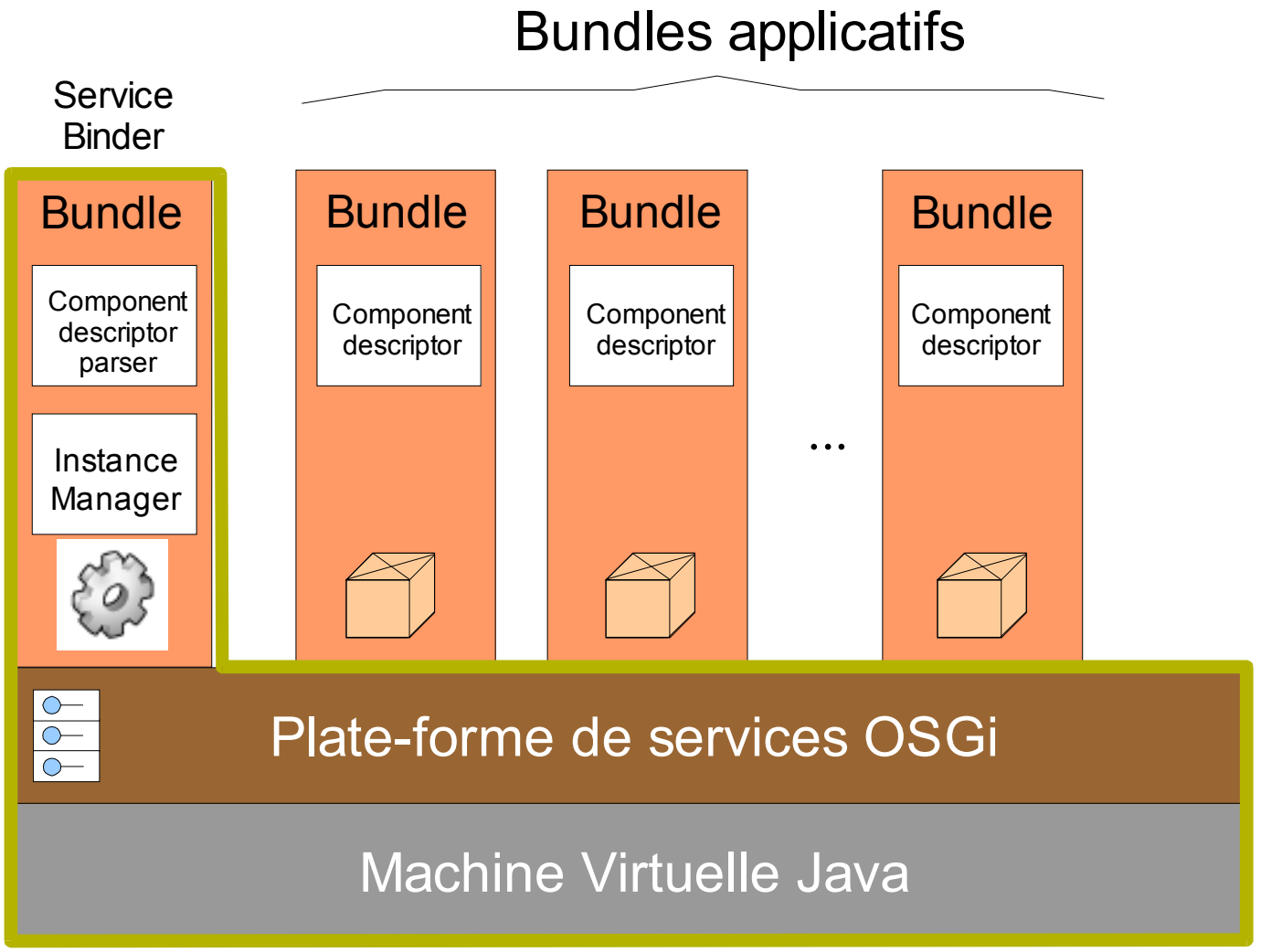
- Registre de services avec mécanismes d'intérrogation (LDAP)
- Mécanismes de notifications
- Gestion des aspects liés au déploiement (bundle)
- OSCAR

Service Binder

Service Binder

- Parseur de descripteur de composants
- Gestionnaire d'instances
- Fabriques
- Espaces de résolution

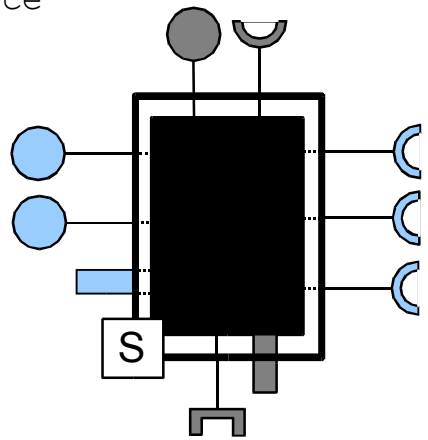
Service Binder



Descripteur de composant

- Descripteur composant

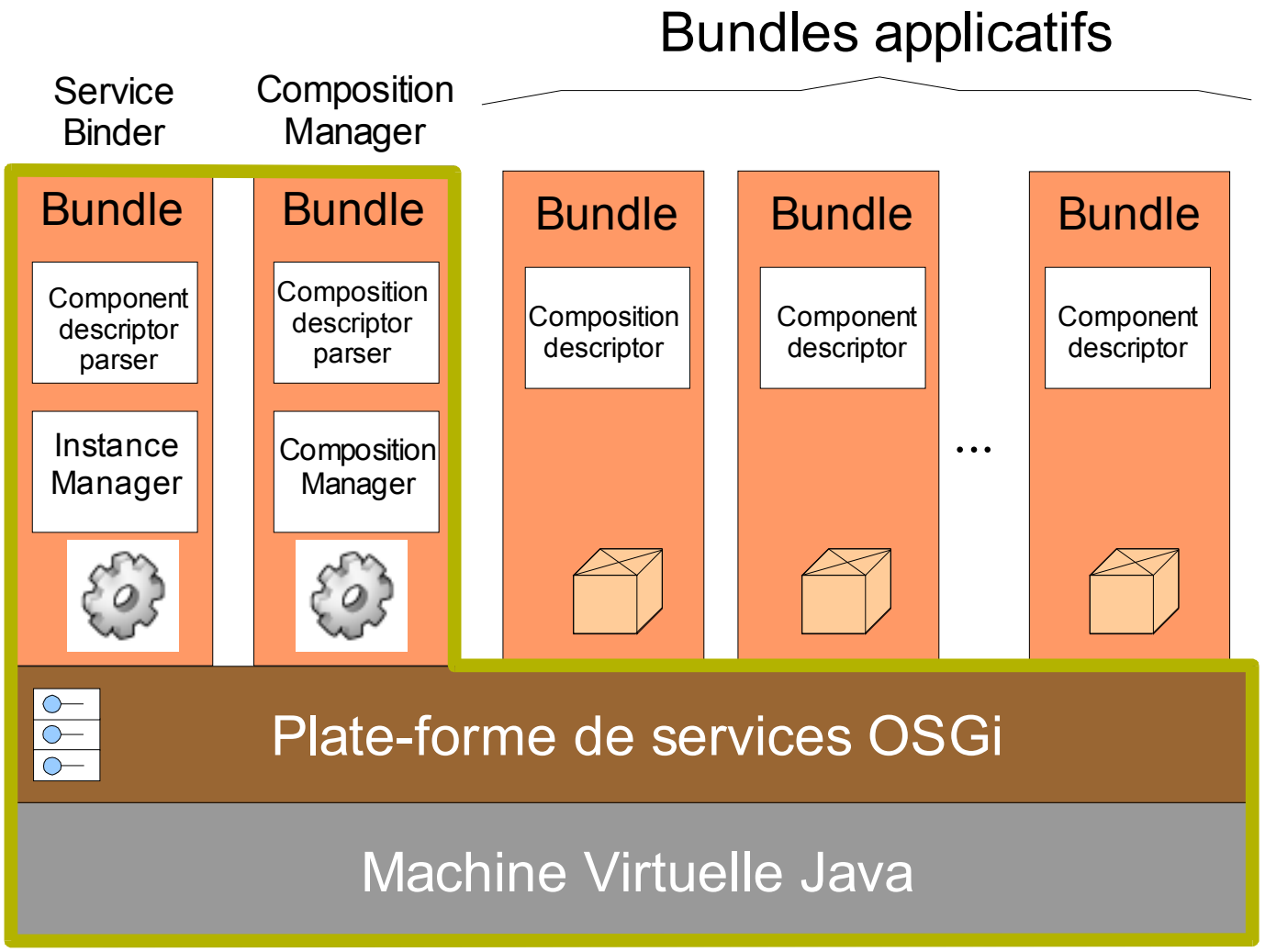
```
<bundle>
  <component class="org.examples.impl.SpellCheckComponent" factory="no">
    <provides service="org.examples.interfaces.SpellCheckService"/>
    <property name="version" value="1.0" type="string"/>
    <requires
        service="org.examples.interfaces.DictionaryService"
        filter="(language=*)"
        cardinality="1..n"
        policy="dynamic"
        bind-method="addDictionary"
        unbind-method="removeDictionary"
    />
  </component>
</bundle>
```



- Extension du Manifest

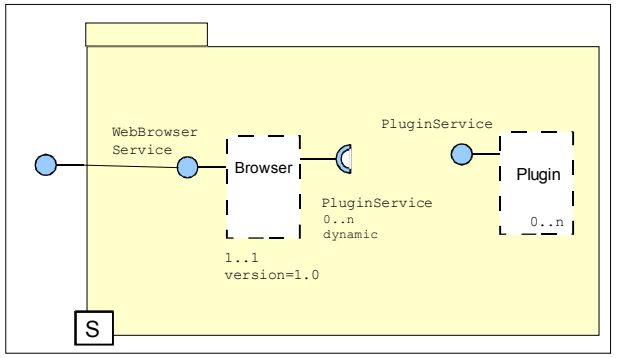
```
Bundle-Activator: org.examples.impl.Activator
Import-Package:
 org.ungoverned.gravity.servicebinder; specification-version="1.1",
Bundle-Classpath: ., somelib.jar
Bundle-Version: 1.0.0
Metadata-Location: org/examples/impl/res/metadata.xml
```

Composition manager

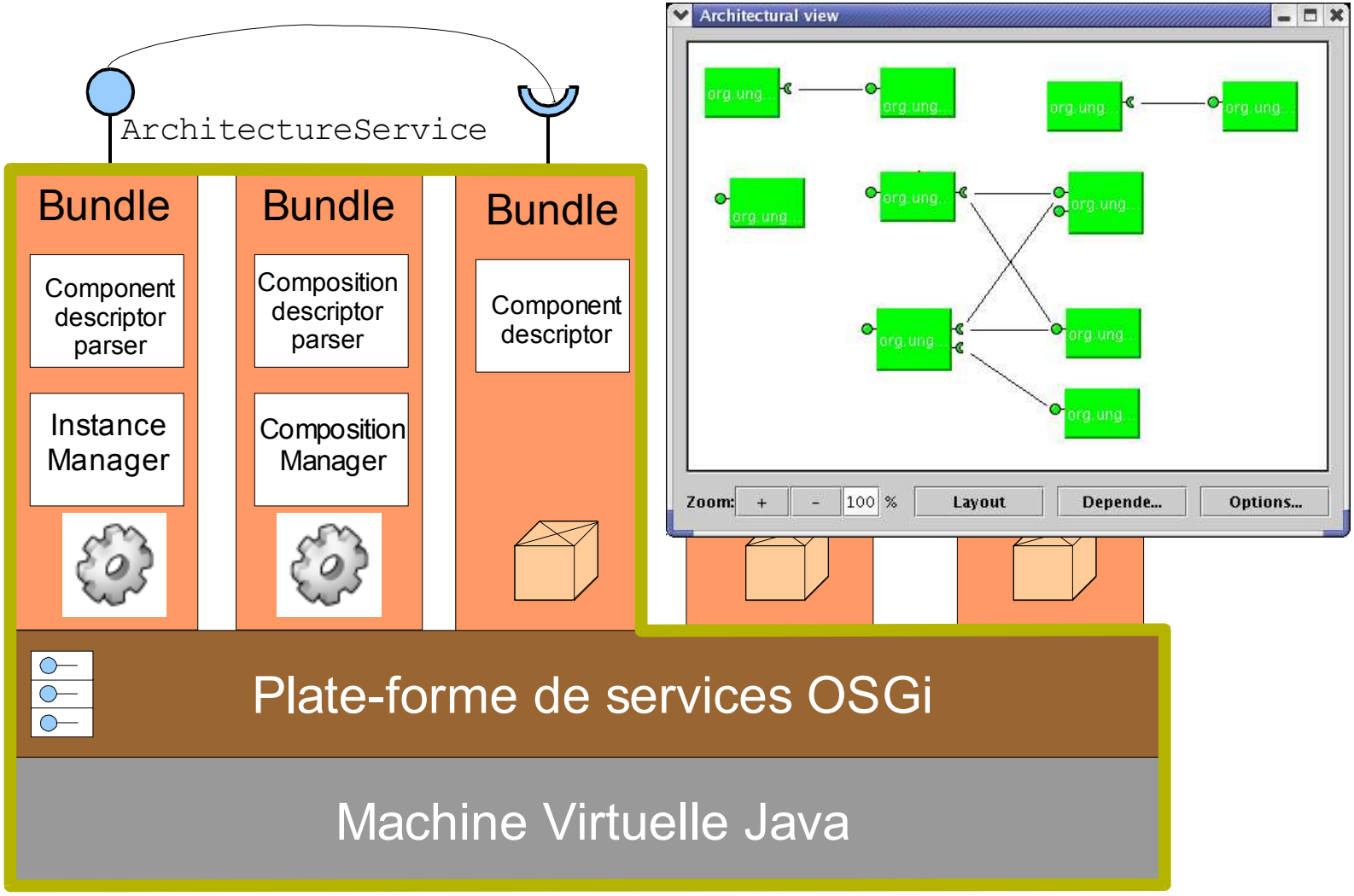


Descripteur de composition

```
<composition factory="yes">
  <exports from="core" service="org.gravity.services.Application" type="provided"/>
  <exports from="core" service="org.gravity.services.WindowManager" type="required"/>

  <property name="appName" value="WebBrowser" type="string"/>

  <placeholder id="core" filter="" cardinality="1..1">
    <provides service="org.gravity.services.Application"/>
    <provides service="org.gravity.services.WebBrowser"/>
    <requires service="org.gravity.services.WindowManager"
      filter=""
      cardinality="1..1"
      policy="dynamic"/>
    <requires service="org.gravity.services.BrowserPlugin"
      filter=""
      cardinality="0..n"
      policy="dynamic"/>
  </placeholder>

  <placeholder id="plug" filter=""
               cardinality="0..n">
    <provides
       service="org.gravity.services.BrowserPlugin"/>
  </placeholder>

</composition>
```



Service d'introspection



Plan de la thèse

Problématique

Fondations

Modèle à composants orienté services

Evaluations

- Service Binder et OSGi
- Schneider Electric
- Ascert
- Lanceur d'applications
- Gravity

Conclusions et perspectives

Service Binder et OSGi

Simplifie la construction d'applications OSGi

- Concepts additionnels compatibles
- Taille réduite (70k), surcout raisonnable:

Task	Standard OSGi (in bytes)	Service Binder (in bytes)	Delta
Registering one service	156	537	244%
Registering one service and requesting one service	864	1344	55%
Additionnal requested services	768	1032	34%

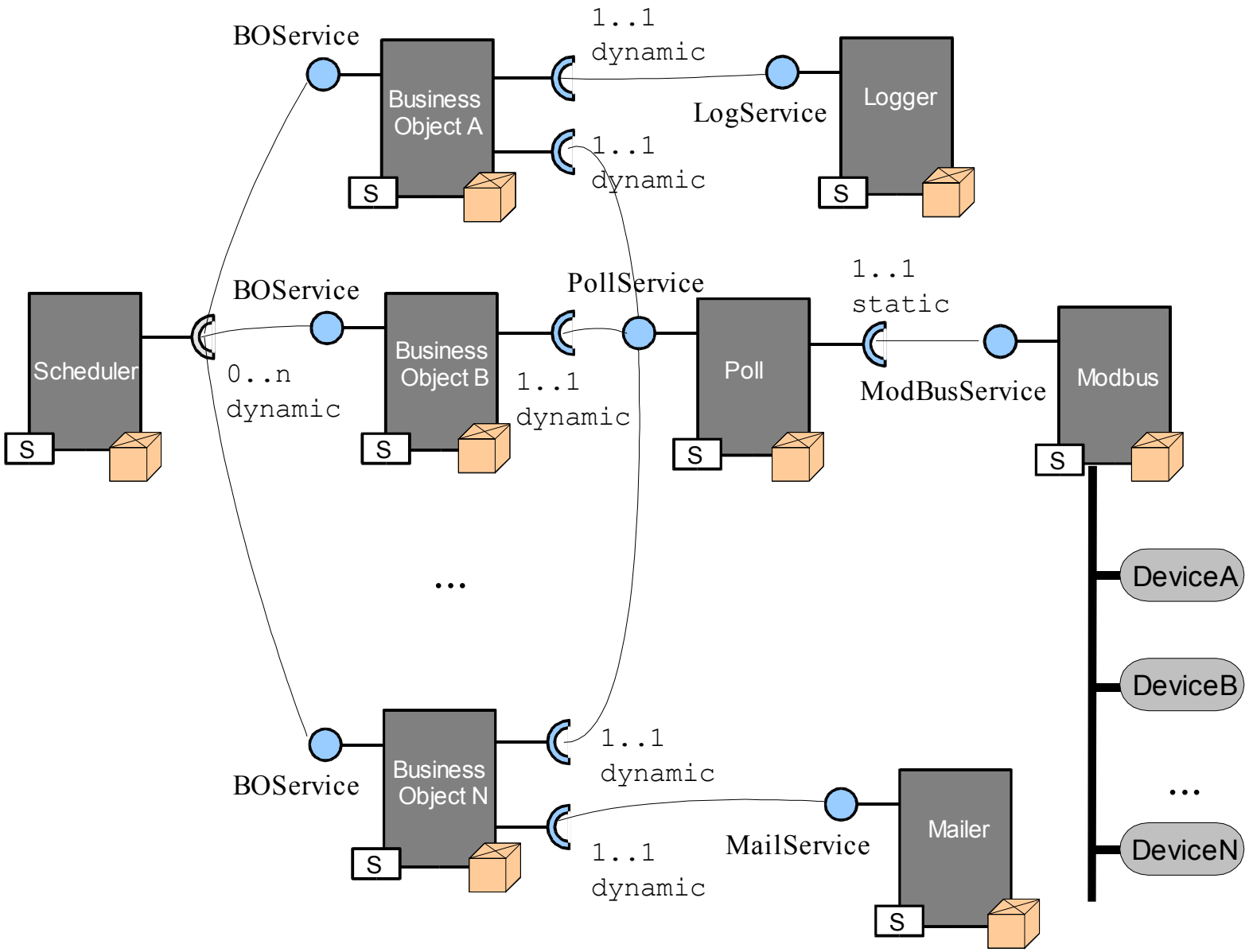
Comité d'OSGi intéressé par ces idées

- Incorporation dans future spécification ?

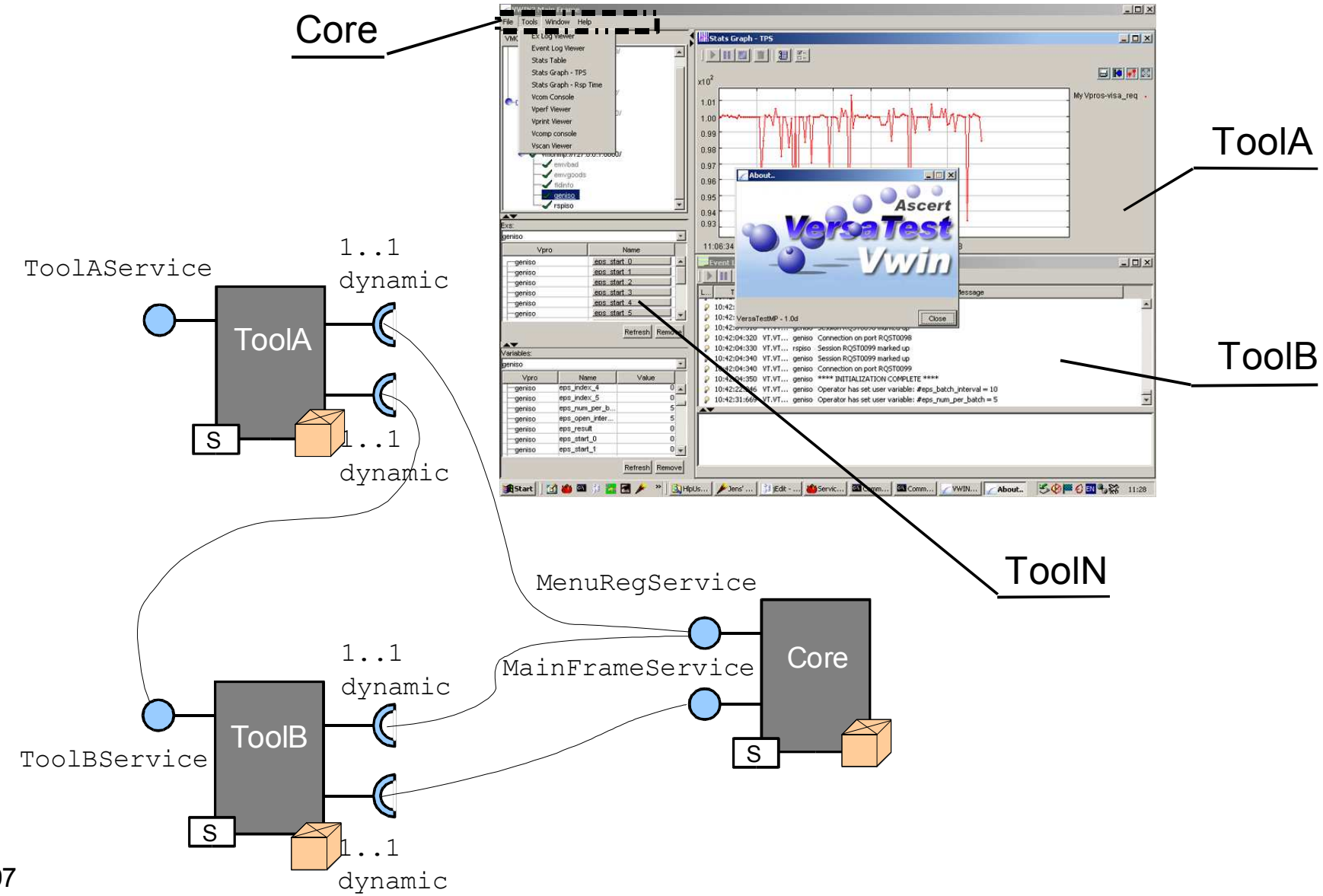
Projet source libre

`http://gravity.sf.net/servicebinder`

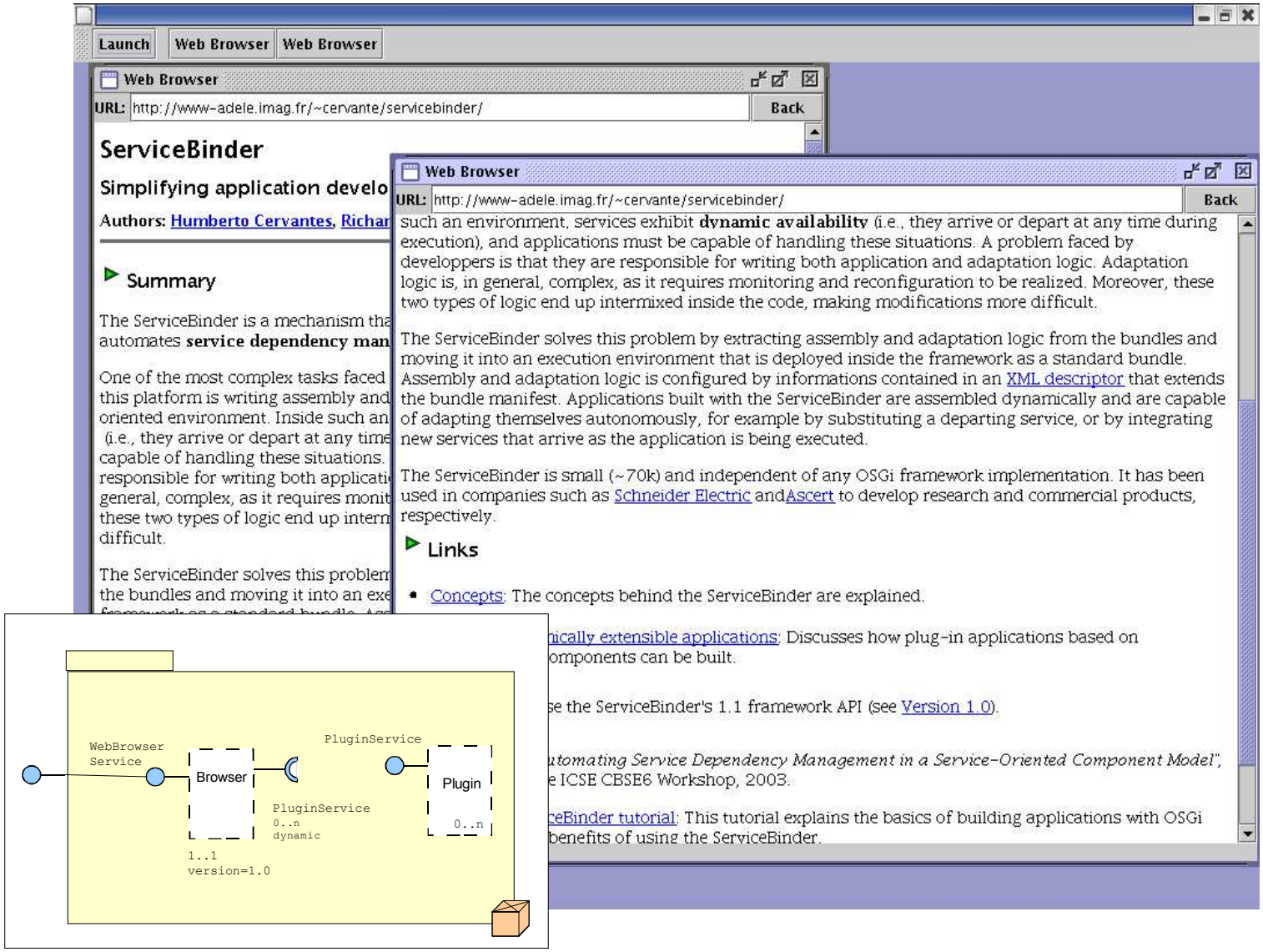
Schneider Electric



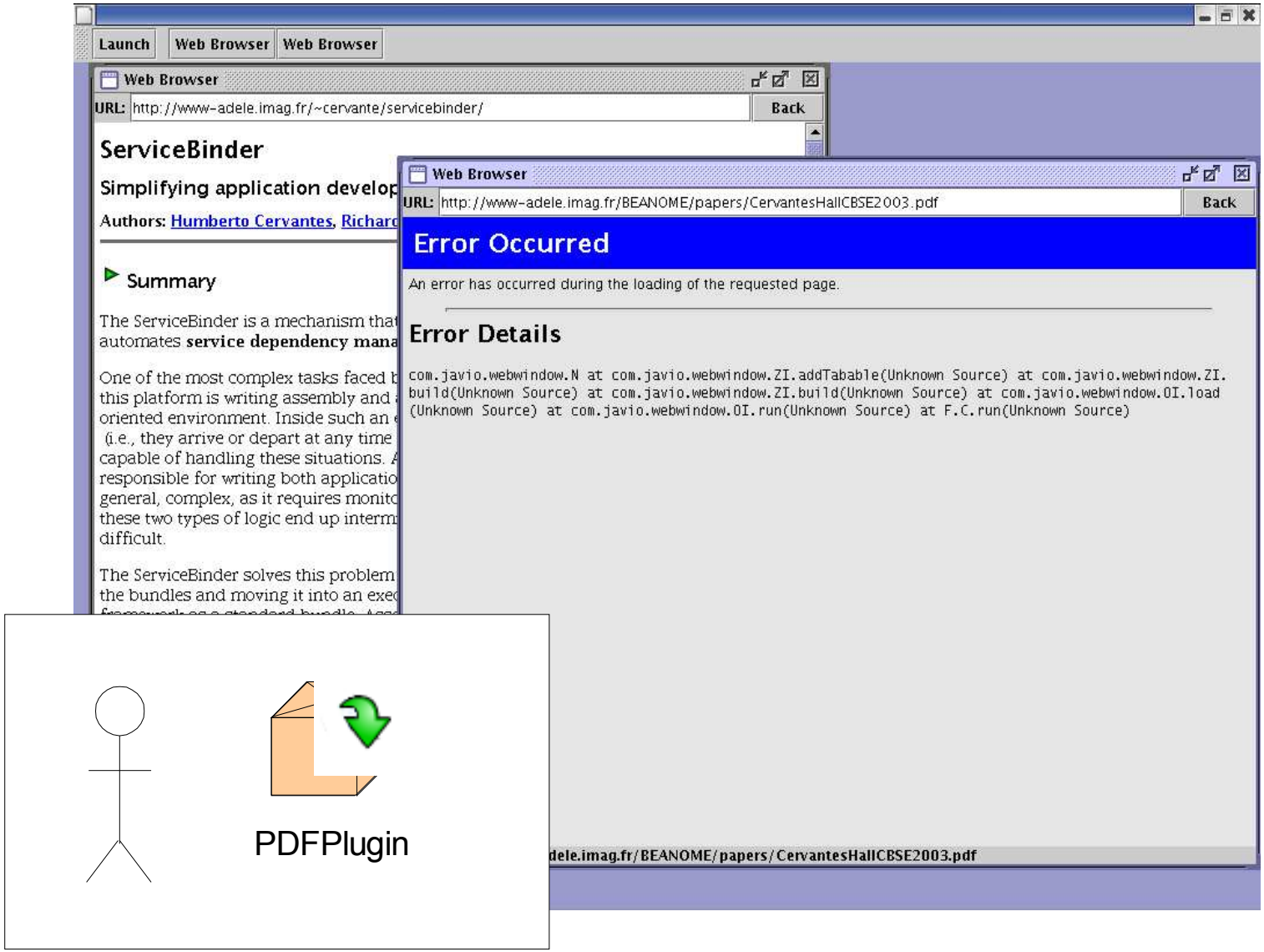
Client VersaTest



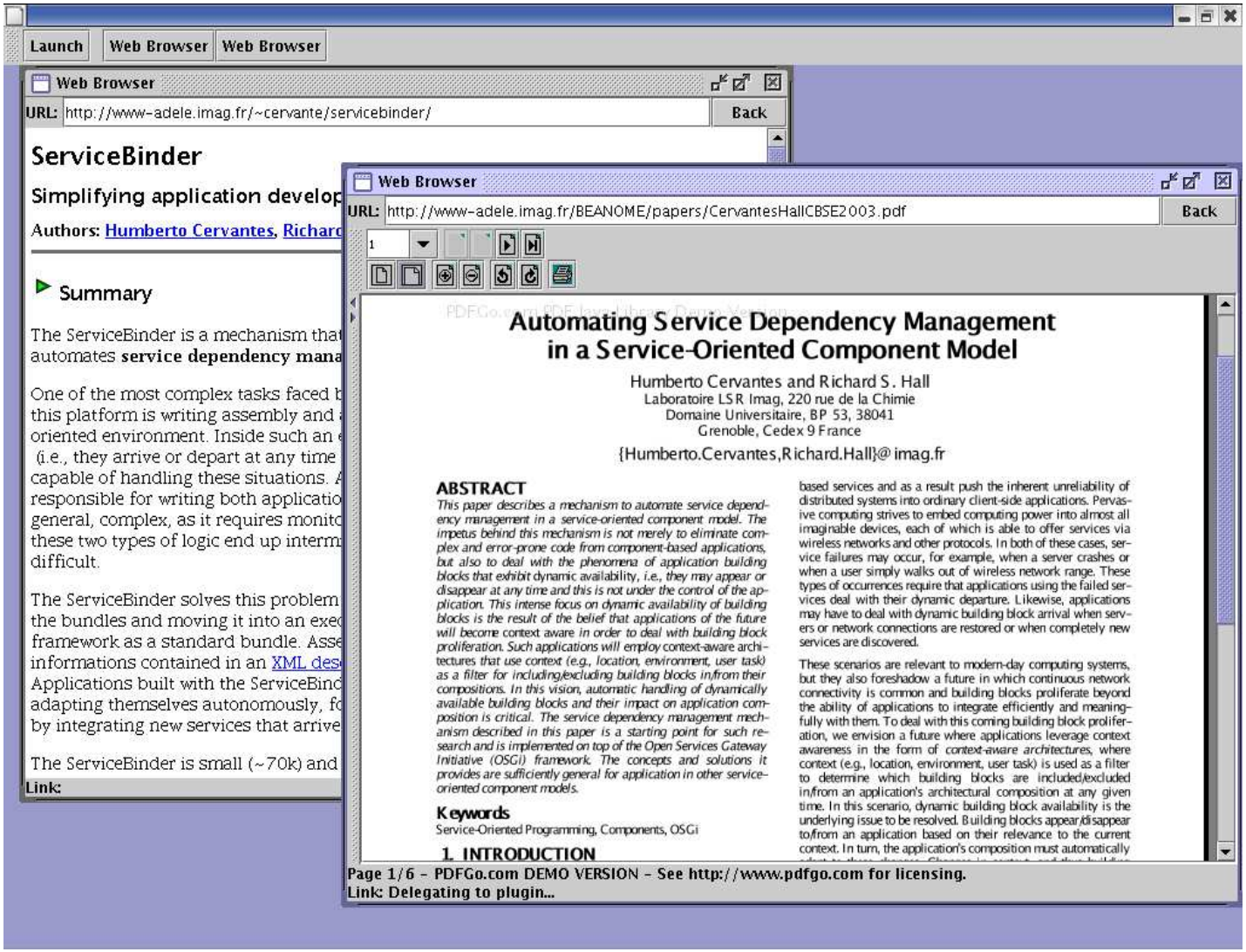
Gestionnaire de composition



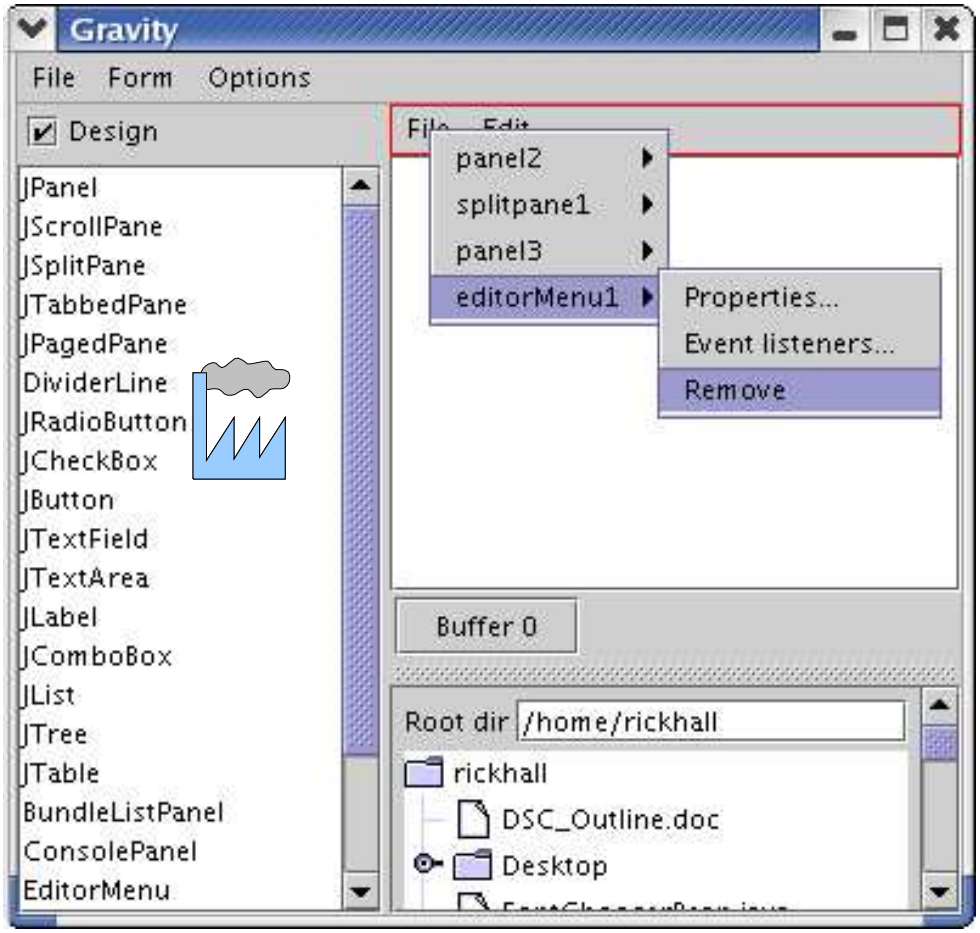
Gestionnaire de composition

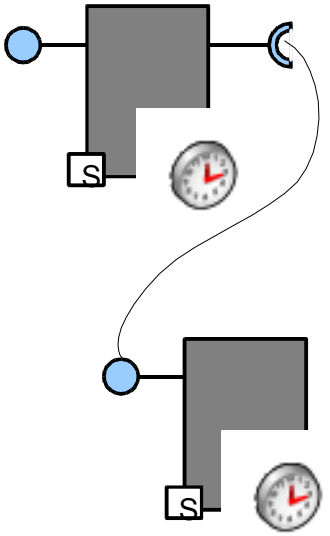


Gestionnaire de composition



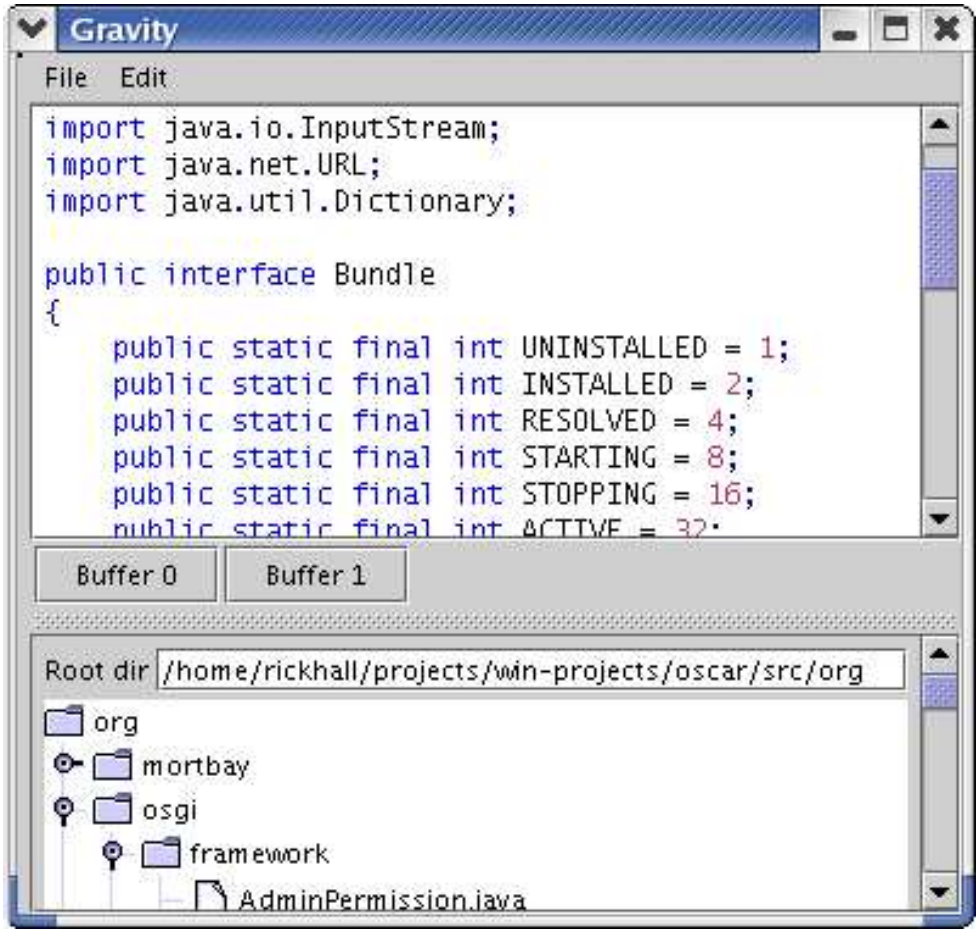
Gravity

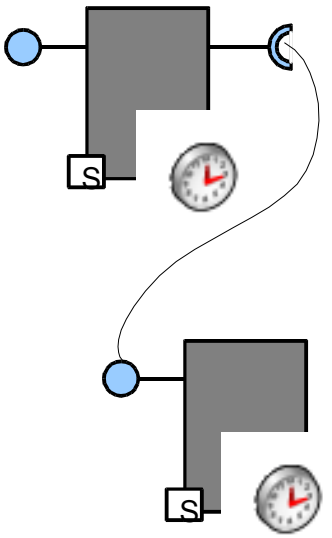




Assemblage

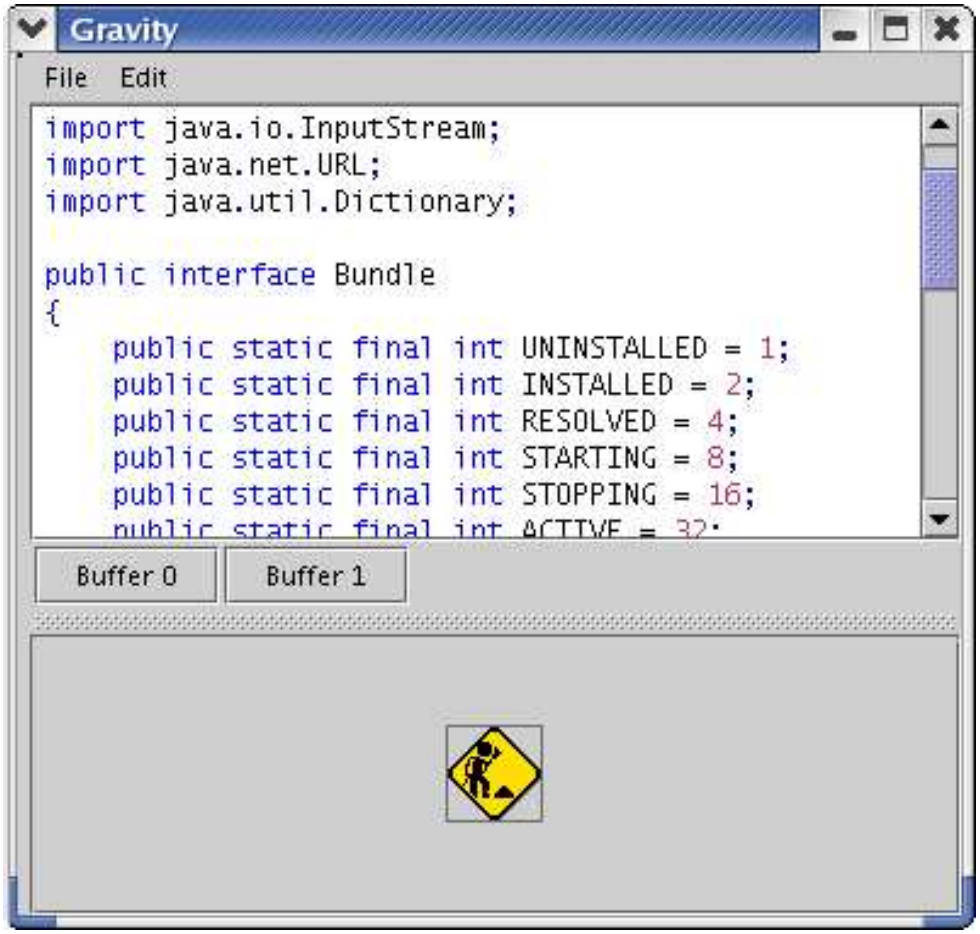
Gravity

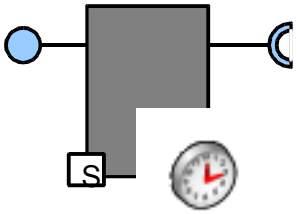


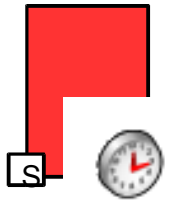


Exécution

Gravity





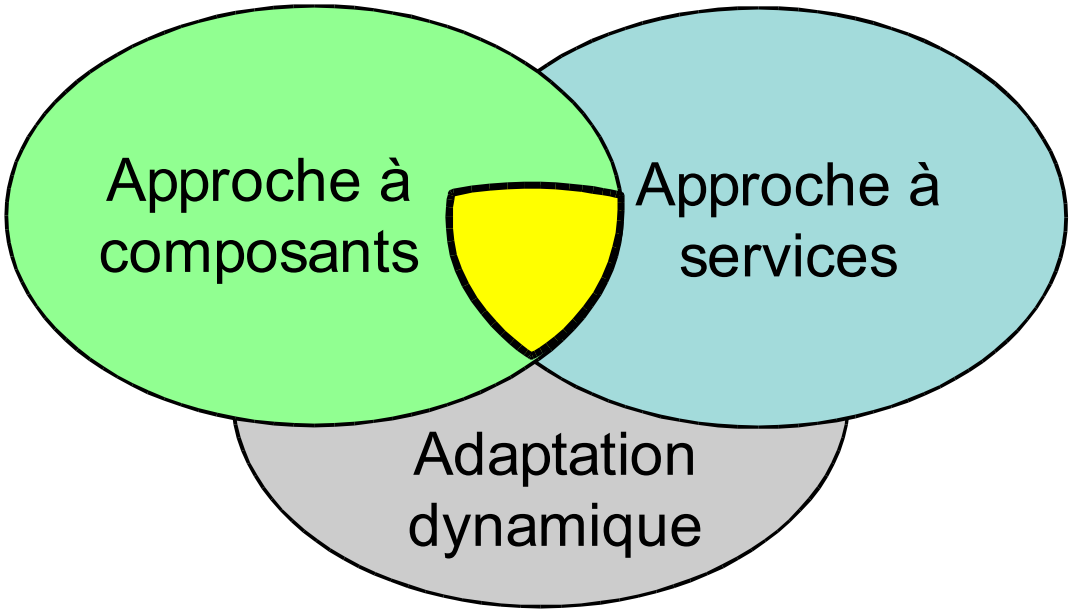


Exécution en mode dégradé

Plan de la thèse

Problématique
Fondations
Modèle à composants orienté services
Réalisation
Évaluations
Conclusion et perspectives

Synthèse



Modèle à composants orienté services

- Construire des applications supportant la disponibilité dynamique
- Support d'exécution de ces applications

Synthèse

Concepts du modèle à composants orienté services

- Composants à services
- Instances valides et invalides
- Gestionnaires d'instances
- Instances de déploiement et dynamiques
- Espaces de résolution
- Déscripteur de composition

Applications capables

- D'auto-assemblage
- D'auto-adaptation à travers l'incorporation, retrait et substitution de fonctionnalités
- De supporter le déploiement continu

Conclusions

Approche se révèle très adéquate

- « *Le tout plus grand que la somme des parties* »
- Nouveaux défis comme l'imprévisibilité

Approche indépendante de modèle à composants et de plate-forme de services

- Autres implémentations possibles

Idées réutilisables dans la programmation orientée services en général

- Services != Services Web

Conclusions

Réalisation

- OSGi est une base adéquate pour construire un modèle à composants
- Rajout de concepts à la plate-forme OSGi, tout en restant compatibles
- Regles d'adaptation simples facilitent adoption
- Retours sur l'utilisation du Service Binder très encourageants

Environnement d'exécution extensible

- Construction d'un noyau orienté services?
- Déploiement continu

Perspectives

Mécanismes de description et de découverte de services plus complexes

- Ajout d'informations sémantiques

Construction d'applications sensibles au contexte

Questions ?

TEMPLATE