Adaptation fonctionnelle de composants gros-grain avec JBOSS/AOP

Olivier Caron, Bernard Carré, Alexis Muller et Gilles Vanwormhoudt

Equipe COCOA,
Laboratoire d'Informatique Fondamentale de Lille,
Université des Sciences et Technologies de Lille,
59655 Villeneuve d'Ascq cedex - France
TELECOM Lille I

journée thématique du groupe COSMAL du 27 janvier 2009

Contexte, point de départ
Travaux existants sur la décomposition fonctionnelle

- De nombreuses approches sur la décomposition fonctionnelle : Catalysis, Subject-Oriented Design, Role-Oriented Design, View-Oriented Design.
- Apport MDA puis MDE :
  - Concrétisation par l'emploi de modèles
  - Réutilisation par le recours à des techniques de paramétrage

Contexte, point de départ
Travaux existants sur la décomposition fonctionnelle

- De nombreuses approches sur la décomposition fonctionnelle : Catalysis, Subject-Oriented Design, Role-Oriented Design, View-Oriented Design.
- Apport MDA puis MDE :
  - Concrétisation par l'emploi de modèles
  - Réutilisation par le recours à des techniques de paramétrage

Contexte, point de départ
Travaux existants sur la décomposition fonctionnelle

- De nombreuses approches sur la décomposition fonctionnelle : Catalysis, Subject-Oriented Design, Role-Oriented Design, View-Oriented Design.
- Apport MDA puis MDE :
  - Concrétisation par l'emploi de modèles
  - Réutilisation par le recours à des techniques de paramétrage

Contexte, point de départ
Travaux existants sur la décomposition fonctionnelle

- De nombreuses approches sur la décomposition fonctionnelle : Catalysis, Subject-Oriented Design, Role-Oriented Design, View-Oriented Design.
- Apport MDA puis MDE :
  - Concrétisation par l'emploi de modèles
  - Réutilisation par le recours à des techniques de paramétrage

Nos travaux sur les modèles paramétrés [ECMDA'05]

- Notion de *composants de modèles*
- Représente une préoccupation fonctionnelle
- Exprime une partie requise complexe sous forme d'un modèle de système
- Un opérateur d'application pour l'expression d'assemblages complexes

Nos travaux sur les modèles paramétrés [ECMDA'05]

- Notion de *composants de modèles*
- Représente une préoccupation fonctionnelle
- Exprime une partie requise complexe sous forme d'un modèle de système
- Un opérateur d'application pour l'expression d'assemblages complexes

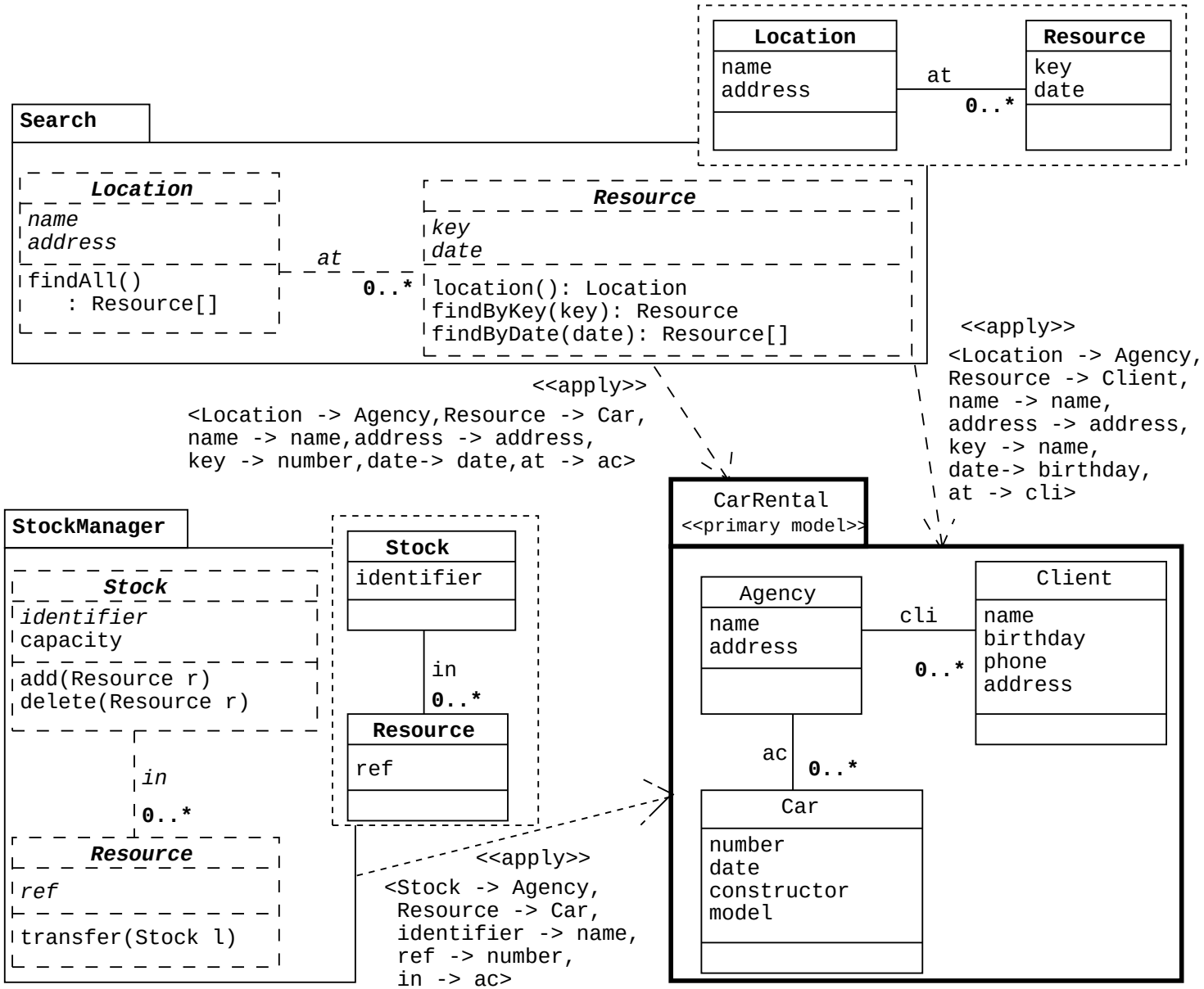
Nos travaux sur les modèles paramétrés [ECMDA'05]

- Notion de *composants de modèles*
- Représente une préoccupation fonctionnelle
- Exprime une partie requise complexe sous forme d'un modèle de système
- Un opérateur d'application pour l'expression d'assemblages complexes

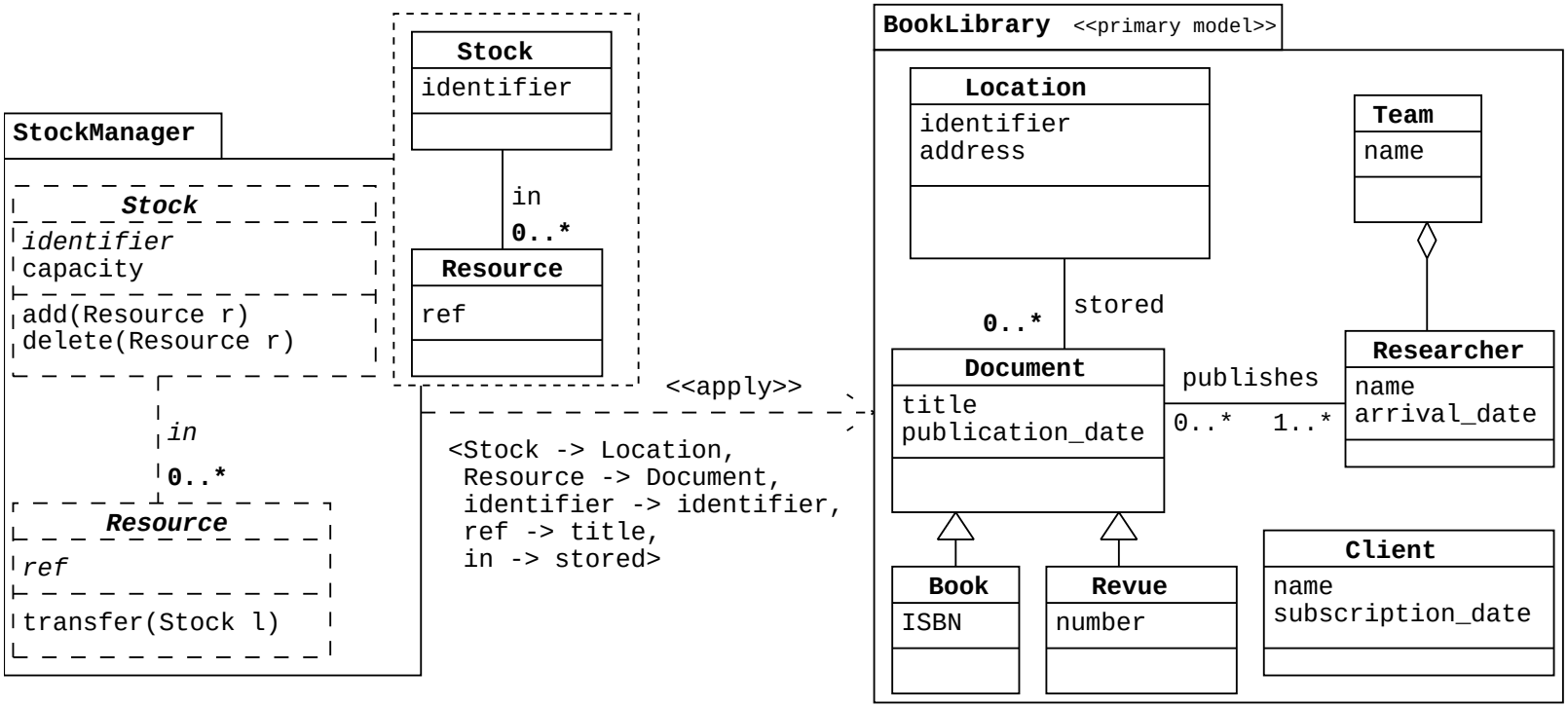
Nos travaux sur les modèles paramétrés [ECMDA'05]

- Notion de *composants de modèles*
- Représente une préoccupation fonctionnelle
- Exprime une partie requise complexe sous forme d'un modèle de système
- Un opérateur d'application pour l'expression d'assemblages complexes

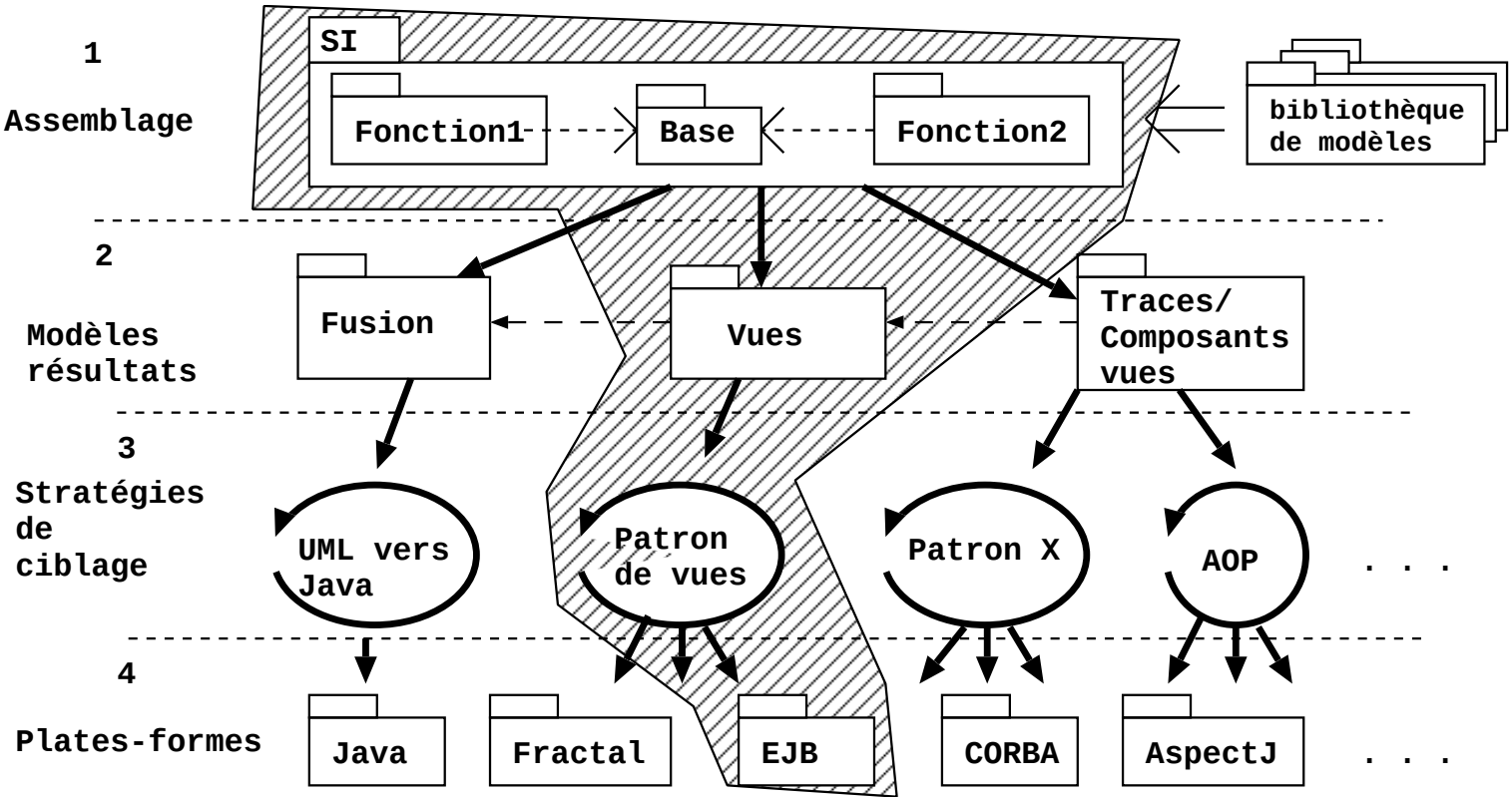
Un exemple d'assemblage



Un exemple de réutilisation



Une voie dans la chaîne de production logicielle flexible [LMO'07]



Objectif

Réaliser une application par assemblage de composants de systèmes d'information (CSI) réutilisables

- Nous considérons les CSI :
  - Un CSI capture une fonction spécifique du SI (découpage fonctionnel)
  - UN CSI est constitué d'un ensemble d'objets ou de composants plus élémentaires (aspect gros-grain)
  - un CSI est réutilisable (dimension composant)

Principale difficulté

La connexion complexe de composants gros-grains

Objectif

Réaliser une application par assemblage de composants de systèmes d'information (CSI) réutilisables

- Nous considérons les CSI :
  - Un CSI capture une fonction spécifique du SI (découpage fonctionnel)
  - UN CSI est constitué d'un ensemble d'objets ou de composants plus élémentaires (aspect gros-grain)
  - un CSI est réutilisable (dimension composant)

Principale difficulté

La connexion complexe de composants gros-grains

Objectif

Réaliser une application par assemblage de composants de systèmes d'information (CSI) réutilisables

- Nous considérons les CSI :
  - Un CSI capture une fonction spécifique du SI (découpage fonctionnel)
  - UN CSI est constitué d'un ensemble d'objets ou de composants plus élémentaires (aspect gros-grain)
  - un CSI est réutilisable (dimension composant)

Principale difficulté

La connexion complexe de composants gros-grains

Objectif

Réaliser une application par assemblage de composants de systèmes d'information (CSI) réutilisables

- Nous considérons les CSI :
  - Un CSI capture une fonction spécifique du SI (découpage fonctionnel)
  - UN CSI est constitué d'un ensemble d'objets ou de composants plus élémentaires (aspect gros-grain)
  - un CSI est réutilisable (dimension composant)

Principale difficulté

La connexion complexe de composants gros-grains

Objectif

Réaliser une application par assemblage de composants de systèmes d'information (CSI) réutilisables

- Nous considérons les CSI :
  - Un CSI capture une fonction spécifique du SI (découpage fonctionnel)
  - UN CSI est constitué d'un ensemble d'objets ou de composants plus élémentaires (aspect gros-grain)
  - un CSI est réutilisable (dimension composant)

Principale difficulté

La connexion complexe de composants gros-grains

Objectif

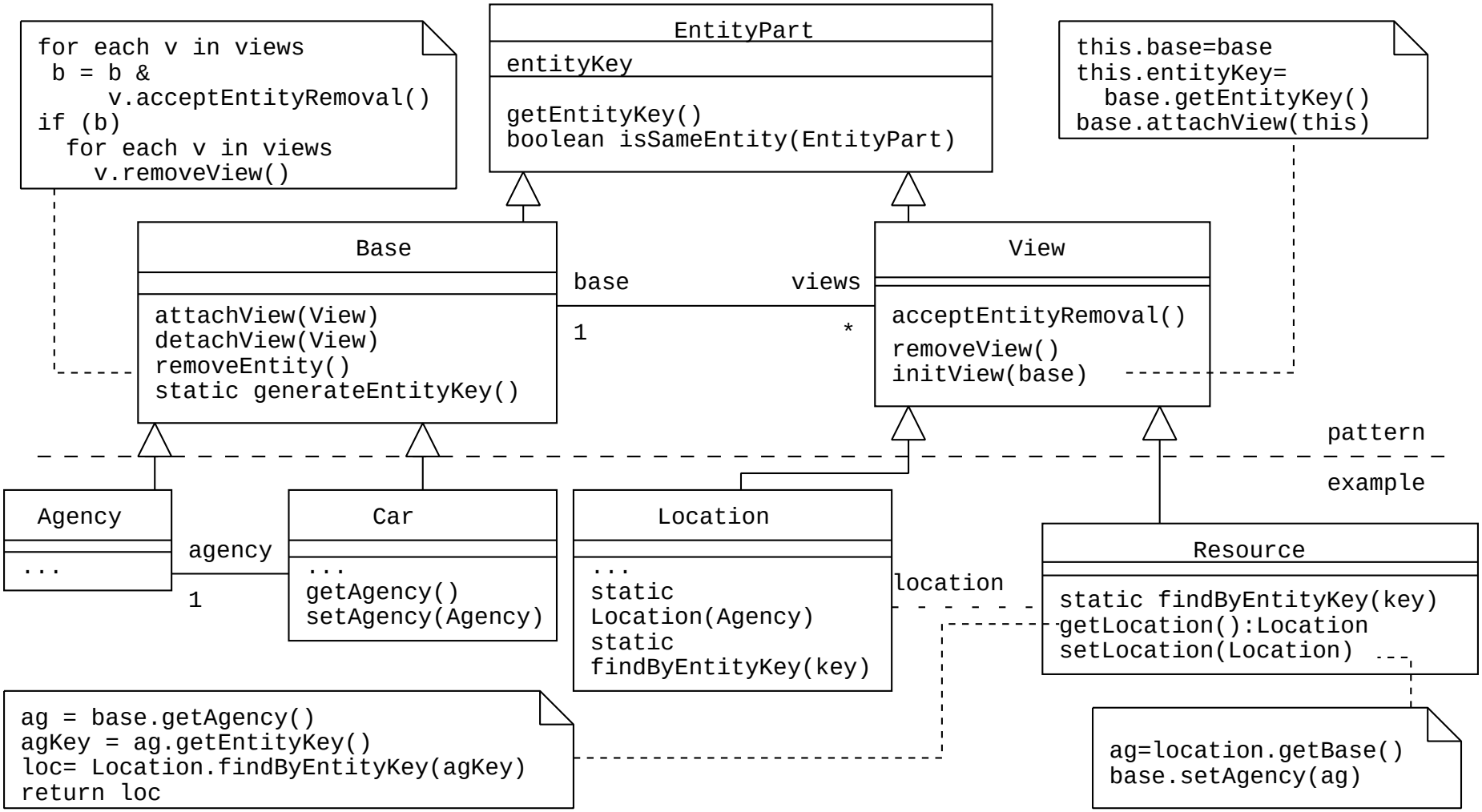
Réaliser une application par assemblage de composants de systèmes d'information (CSI) réutilisables

- Nous considérons les CSI :
  - Un CSI capture une fonction spécifique du SI (découpage fonctionnel)
  - UN CSI est constitué d'un ensemble d'objets ou de composants plus élémentaires (aspect gros-grain)
  - un CSI est réutilisable (dimension composant)

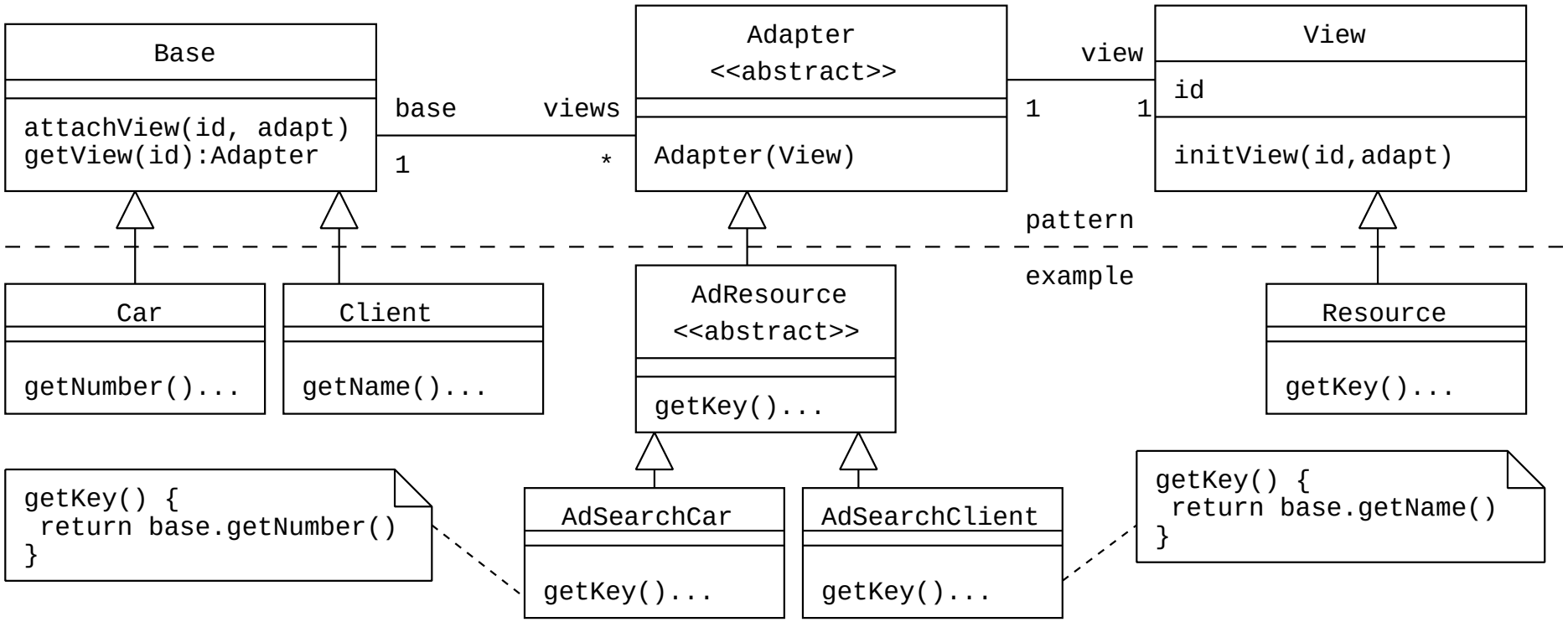
Principale difficulté

La connexion complexe de composants gros-grains

Le patron de représentation éclatée de vues [OOIS'03]

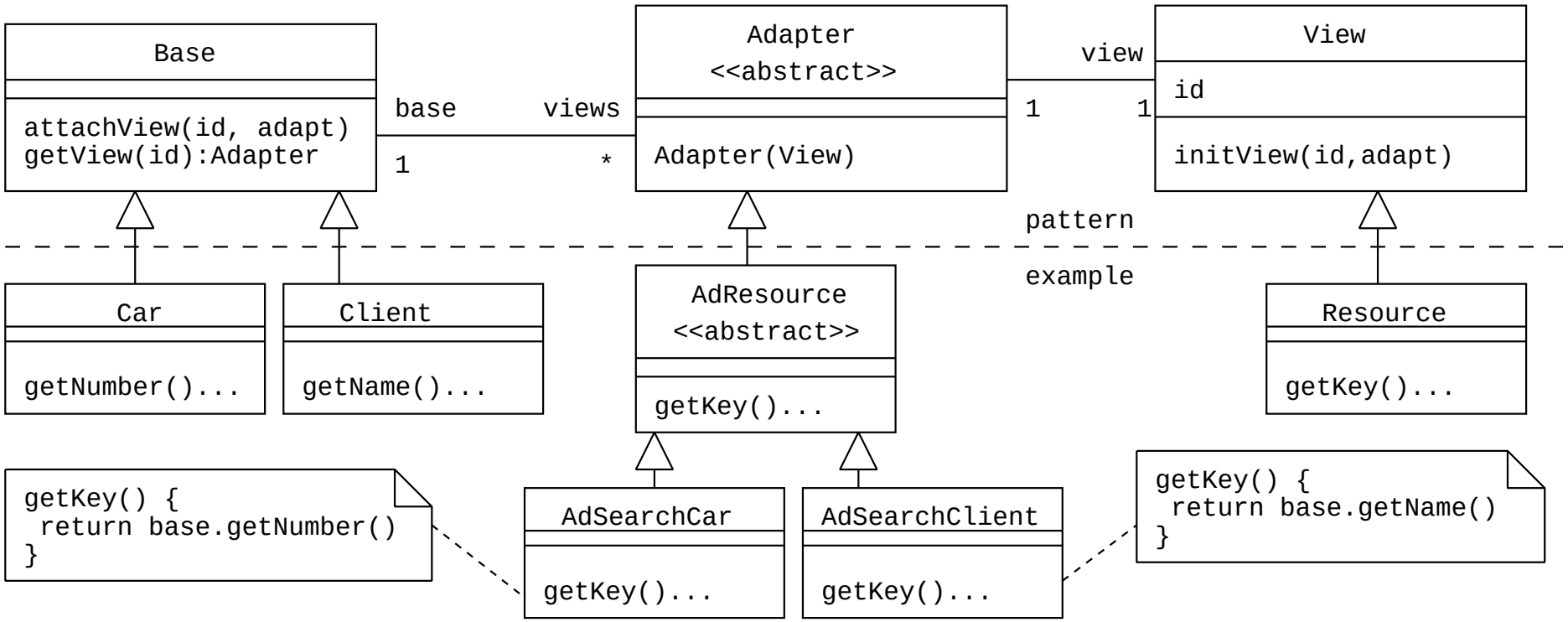


Réutilisation : couplage au patron adaptateur [L'Objet'05]



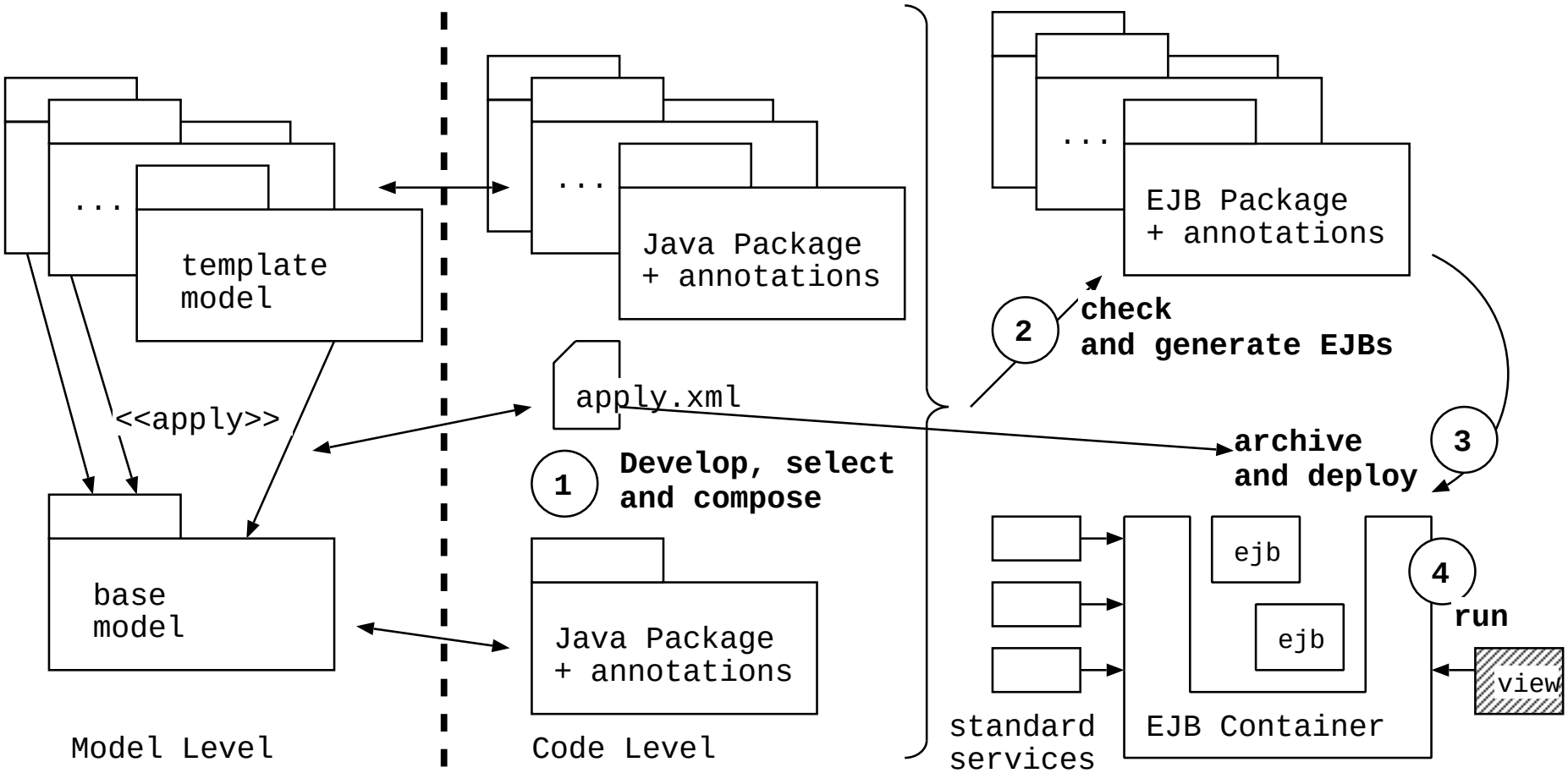
- Déjà une approche générative (CORBA)
- Démarche par interception dynamique dans les conteneurs EJB

Réutilisation : couplage au patron adaptateur [L'Objet'05]



- Déjà une approche générative (CORBA)
- Démarche par interception dynamique dans les conteneurs EJB

Architecture générale du service



Programmation d'un composant CSI Java

- Programmation d'un paquetage Java
- Spécification des éléments requis à l'aide d'un jeu d'annotations :

Element modèle	Annotation	relié à	Signification
package	view.Component	package	composant-vue
class	view.Class	class	classe requise
attribute	view.Attribute	getter/setter methods	attribut requis
operation	view.Method	method	méthode requise
association	view.Role	getter/setter method	association requise

Programmation d'un composant CSI Java

- Programmation d'un paquetage Java
- Spécification des éléments requis à l'aide d'un jeu d'annotations :

Element modèle	**Annotation**	**relié à**	**Signification**
package	view.Component	package	composant-vue
class	view.Class	class	classe requise
attribute	view.Attribute	getter/setter methods	attribut requis
operation	view.Method	method	méthode requise
association	view.Role	getter/setter method	association requise

Exemple du composant `StockManager` (1/2)

```
// package-info.java file
@view.Component
package StockManager ;
```

```
// Resource.java file
package StockManager ;

@view.Class public class Resource  {

  public Resource() {}

  public void transfer(Stock s) {
    this.getStock().delete(this) ; this.setStock(s) ;
    s.add(this) ;
  }

  // Model Required Section
  @view.Attribute
  public String getRef() { return null ; }
  public void setRef(String value) { }

  @view.Role(className="StockManager.Stock", roleName="resources")
  public Stock getStock() { return null ; }
  public void  setStock(Stock s) { }
}
```

Exemple du composant `StockManager` (2/2)

```
// Stock.java file
package StockManager ;
import java.util.Collection ;

@view.Class  public class Stock {
  private int capacity ;

  public Stock() { }

  public int getCapacity() { return this.capacity ;}
  public void setCapacity(int  value) { this.capacity=value ; }

  public void add(Resource r) {
    Collection<Resource> resources=this.getResources();
    if (resources.add(r)) { capacity++ ; this.setResources(resources) ; }
  }

  public void delete(Resource r) {
    Collection<Resource> resources=this.getResources();
    if (resources.remove(r)) { capacity-- ; this.setResources(resources) ; }
  }
  // Model Required Section
  @view.Attribute
  public String getIdentifier() { return null ; }
  public void setIdentifier(String value) {}

  @view.Role(className="StockManager.Resource", roleName="stock")
  public Collection<Resource> getResources() {
    return null ;
  }
  public void setResources(Collection<Resource> value){} ;
}
```

Spécification de l'assemblage de composants

- Décrit les connexions entre les composants CSI et le paquetage de base.
- Utilisation d'un document XML
- Seul élément propre à l'application

Spécification de l'assemblage de composants

- Décrit les connexions entre les composants CSI et le paquetage de base.
- Utilisation d'un document XML
- Seul élément propre à l'application

Spécification de l'assemblage de composants

- Décrit les connexions entre les composants CSI et le paquetage de base.
- Utilisation d'un document XML
- Seul élément propre à l'application

L'assemblage de l'exemple précédent (1/2)

```
<?xml version='1.0' encoding='ISO-8859-1' standalone="no"?>
<!DOCTYPE binding SYSTEM "./dtd/binding.dtd">
<binding>
 <view id="SearchClient">
  <class name="Search.Location" linkedTo="CarRental.Agency">
    <attribute name="name" linkedTo="name" />
    <attribute name="address" linkedTo="address" />
    <role name="resources" linkedTo="clients"  />
  </class>
  <class name="Search.Resource" linkedTo="CarRental.Client">
    <attribute name="key" linkedTo="name" />
    <attribute name="date" linkedTo="birthday" />
    <role name="location" linkedTo="agency"  />
  </class>
 </view>
 <view id="SearchCar">
  <class name="Search.Location" linkedTo="CarRental.Agency">
    <attribute name="name" linkedTo="name" />
    <attribute name="address" linkedTo="address" />
    <role name="resources" linkedTo="cars"  />
  </class>
  <class name="Search.Resource" linkedTo="CarRental.Car">
    <attribute name="key" linkedTo="number" />
    <attribute name="date" linkedTo="date" />
    <role name="location" linkedTo="agency"  />
  </class>
 </view>
```

L'assemblage de l'exemple précédent (2/2)

```
 <view id="StockManagerCarRental">
  <class name="StockManager.Resource" linkedTo="CarRental.Car">
    <attribute name="ref" linkedTo="number" />
    <role name="stock" linkedTo="agency"  />
  </class>
  <class name="StockManager.Stock" linkedTo="CarRental.Agency">
    <attribute name="identifier" linkedTo="name" />
    <role name="resources" linkedTo="cars" />
  </class>
 </view>
</binding>
```

Intégration du service de vues dans les conteneurs EJB

- Objectifs du service :
  - Assurer la gestion des objets conceptuels constitué d'un fragment de base et de ses fragments de vues.
  - Assurer la connexion des composants-vues au composant de base conformément au descripteur d'assemblage
  - Supporter l'application multiple des composants-vues
- Implémentation
  - Intégration dans serveur Jboss/AOP. Le support AOP facilite l'intégration du service de vues (un service = un aspect).
  - Application du patron de représentation éclatée
  - Application du patron Adaptateur par interception dynamique (reflexion Java, XML, invocation dynamique)

Intégration du service de vues dans les conteneurs EJB

- Objectifs du service :
  - Assurer la gestion des objets conceptuels constitué d'un fragment de base et de ses fragments de vues.
  - Assurer la connexion des composants-vues au composant de base conformément au descripteur d'assemblage
  - Supporter l'application multiple des composants-vues
- Implémentation
  - Intégration dans serveur Jboss/AOP. Le support AOP facilite l'intégration du service de vues (un service = un aspect).
  - Application du patron de représentation éclatée
  - Application du patron Adaptateur par interception dynamique (reflexion Java, XML, invocation dynamique)

Intégration du service de vues dans les conteneurs EJB

- Objectifs du service :
  - Assurer la gestion des objets conceptuels constitué d'un fragment de base et de ses fragments de vues.
  - Assurer la connexion des composants-vues au composant de base conformément au descripteur d'assemblage
  - Supporter l'application multiple des composants-vues
- Implémentation
  - Intégration dans serveur Jboss/AOP. Le support AOP facilite l'intégration du service de vues (un service = un aspect).
  - Application du patron de représentation éclatée
  - Application du patron Adaptateur par interception dynamique (reflexion Java, XML, invocation dynamique)

Intégration du service de vues dans les conteneurs EJB

- Objectifs du service :
  - Assurer la gestion des objets conceptuels constitué d'un fragment de base et de ses fragments de vues.
  - Assurer la connexion des composants-vues au composant de base conformément au descripteur d'assemblage
  - Supporter l'application multiple des composants-vues
- Implémentation
  - Intégration dans serveur Jboss/AOP. Le support AOP facilite l'intégration du service de vues (un service = un aspect).
  - Application du patron de représentation éclatée
  - Application du patron Adaptateur par interception dynamique (reflexion Java, XML, invocation dynamique)

Intégration du service de vues dans les conteneurs EJB

- Objectifs du service :
  - Assurer la gestion des objets conceptuels constitué d'un fragment de base et de ses fragments de vues.
  - Assurer la connexion des composants-vues au composant de base conformément au descripteur d'assemblage
  - Supporter l'application multiple des composants-vues
- Implémentation
  - Intégration dans serveur Jboss/AOP. Le support AOP facilite l'intégration du service de vues (un service = un aspect).
  - Application du patron de représentation éclatée
  - Application du patron Adaptateur par interception dynamique (reflexion Java, XML, invocation dynamique)

Intégration du service de vues dans les conteneurs EJB

- Objectifs du service :
  - Assurer la gestion des objets conceptuels constitué d'un fragment de base et de ses fragments de vues.
  - Assurer la connexion des composants-vues au composant de base conformément au descripteur d'assemblage
  - Supporter l'application multiple des composants-vues
- Implémentation
  - Intégration dans serveur Jboss/AOP. Le support AOP facilite l'intégration du service de vues (un service = un aspect).
  - Application du patron de représentation éclatée
  - Application du patron Adaptateur par interception dynamique (reflexion Java, XML, invocation dynamique)

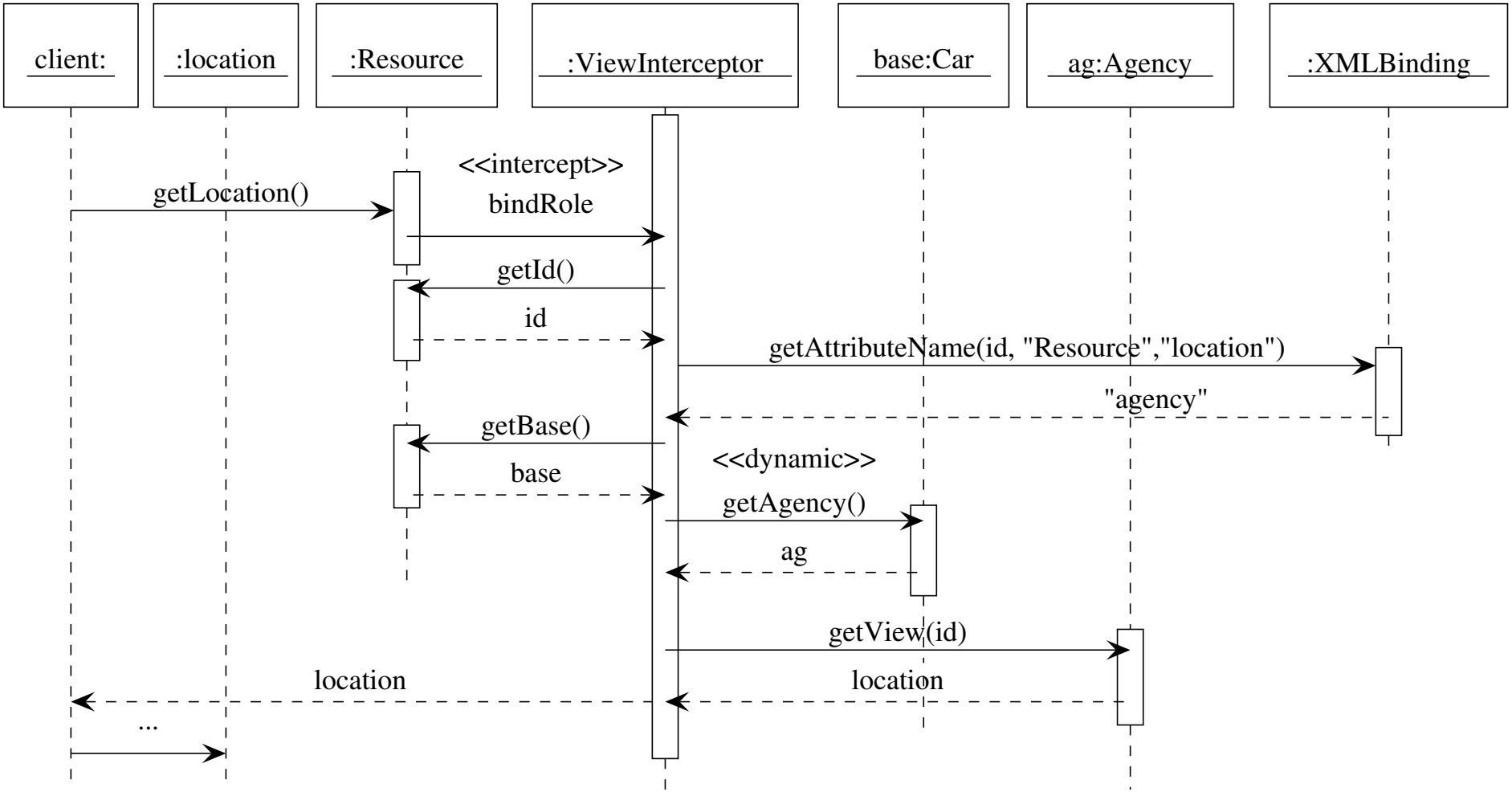
Intégration du service de vues dans les conteneurs EJB

- Objectifs du service :
  - Assurer la gestion des objets conceptuels constitué d'un fragment de base et de ses fragments de vues.
  - Assurer la connexion des composants-vues au composant de base conformément au descripteur d'assemblage
  - Supporter l'application multiple des composants-vues
- Implémentation
  - Intégration dans serveur Jboss/AOP. Le support AOP facilite l'intégration du service de vues (un service = un aspect).
  - Application du patron de représentation éclatée
  - Application du patron Adaptateur par interception dynamique (reflexion Java, XML, invocation dynamique)

Intégration du service de vues dans les conteneurs EJB

- Objectifs du service :
  - Assurer la gestion des objets conceptuels constitué d'un fragment de base et de ses fragments de vues.
  - Assurer la connexion des composants-vues au composant de base conformément au descripteur d'assemblage
  - Supporter l'application multiple des composants-vues
- Implémentation
  - Intégration dans serveur Jboss/AOP. Le support AOP facilite l'intégration du service de vues (un service = un aspect).
  - Application du patron de représentation éclatée
  - Application du patron Adaptateur par interception dynamique (reflexion Java, XML, invocation dynamique)

Exemple d'une interception dynamique d'une association



Exemple d'utilisation des composants-vues

```
 Agency ag1 = (Agency) em.find(Agency.class,"key-ag1") ;
 Search.Location location
     = (Search.Location) ag1.getView("SearchCar") ;
 Collection<Search.Resource> resources = location.findAll() ;
 // resources are view fragments of cars

 location =(Search.Location) ag1.getView("SearchClient") ;
 resources = location.findAll() ;
 // resources are view fragments of clients
 ( (Search.Resource) resources.toArray()[0] ).getLocation() ;

Car newCar=new Car() ; // => creation of view fragments ;
entityManager.persist(aCar) ; // the entity manager now
                              // manages automatically
                              // view fragments
StockManager.Stock stock = ag1.getView("StockManagerCarRental") ;
StockManager.Resource r = newCar.getView("StockManagerCarRental") ;
stock.add(r) ; // now ag1 manages newCar
```

Conclusion

- Des composants CSI gros-grain réutilisables dans la technologie EJB :
  - Les composants sont EJB-compatibles
  - Reprise des pratiques usuelles des développeurs EJB (Java + annotations+ xml)
  - Compatible avec les services EJB standards (persistance, sécurité, etc)
- Le langage d'annotations est utilisable isolément.
- Approche par annotations généralisable à d'autres technologies Java.

Conclusion

- Des composants CSI gros-grain réutilisables dans la technologie EJB :
  - Les composants sont EJB-compatibles
  - Reprise des pratiques usuelles des développeurs EJB (Java + annotations+ xml)
  - Compatible avec les services EJB standards (persistance, sécurité, etc)
- Le langage d'annotations est utilisable isolément.
- Approche par annotations généralisable à d'autres technologies Java.

Conclusion

- Des composants CSI gros-grain réutilisables dans la technologie EJB :
  - Les composants sont EJB-compatibles
  - Reprise des pratiques usuelles des développeurs EJB (Java + annotations+ xml)
  - Compatible avec les services EJB standards (persistance, sécurité, etc)
- Le langage d'annotations est utilisable isolément.
- Approche par annotations généralisable à d'autres technologies Java.

Conclusion

- Des composants CSI gros-grain réutilisables dans la technologie EJB :
  - Les composants sont EJB-compatibles
  - Reprise des pratiques usuelles des développeurs EJB (Java + annotations+ xml)
  - Compatible avec les services EJB standards (persistance, sécurité, etc)
- Le langage d'annotations est utilisable isolément.
- Approche par annotations généralisable à d'autres technologies Java.

Conclusion

- Des composants CSI gros-grain réutilisables dans la technologie EJB :
  - Les composants sont EJB-compatibles
  - Reprise des pratiques usuelles des développeurs EJB (Java + annotations+ xml)
  - Compatible avec les services EJB standards (persistance, sécurité, etc)
- Le langage d'annotations est utilisable isolément.
- Approche par annotations généralisable à d'autres technologies Java.

Conclusion

- Des composants CSI gros-grain réutilisables dans la technologie EJB :
  - Les composants sont EJB-compatibles
  - Reprise des pratiques usuelles des développeurs EJB (Java + annotations+ xml)
  - Compatible avec les services EJB standards (persistance, sécurité, etc)
- Le langage d'annotations est utilisable isolément.
- Approche par annotations généralisable à d'autres technologies Java.