Aspect-Component Unification Approach [Pessemier'07] - Discussion

Hakim Hannousse

ECOLE DES MINES DE NANTES

DEPARTMENT OF COMPUTER SCIENCE

Aspect Models - Comparison

	AspectJ	**Filtres de Composition**	**JAC / HyperJ**
Symétrie d'élément	non	non	oui
Symétrie de relation Placement	non Aspect	oui Tout	oui Tout
Symétrie de relation Portée	non Binaire	oui Totale	oui Totale
Séparation des préoccupations transverses	oui	oui	oui

Component Models - Comparison

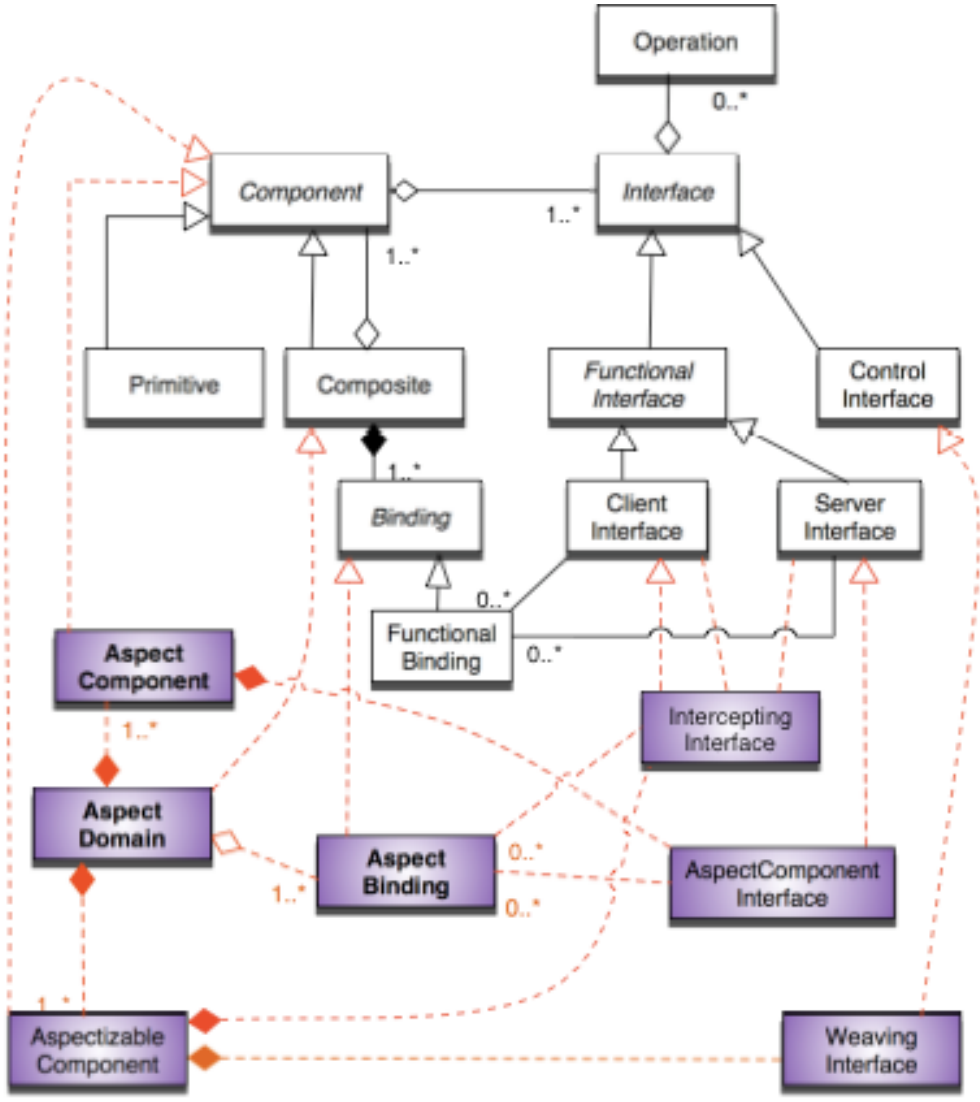
	EJB	Spring	OpenCOM	Fractal
Séparation des préoccupations fonctionnelles	oui	oui	oui	oui
Séparation des préoccupations techniques	non	oui	oui	oui
Général	non	non	oui	oui
Réflexif	non	non	oui	oui
Hiérarchique	non	non	oui	oui
Extensible	non	non	non	oui
Séparation des préoccupations transverses	non	non	non	non

Aspect-Component Based Models - Comparison

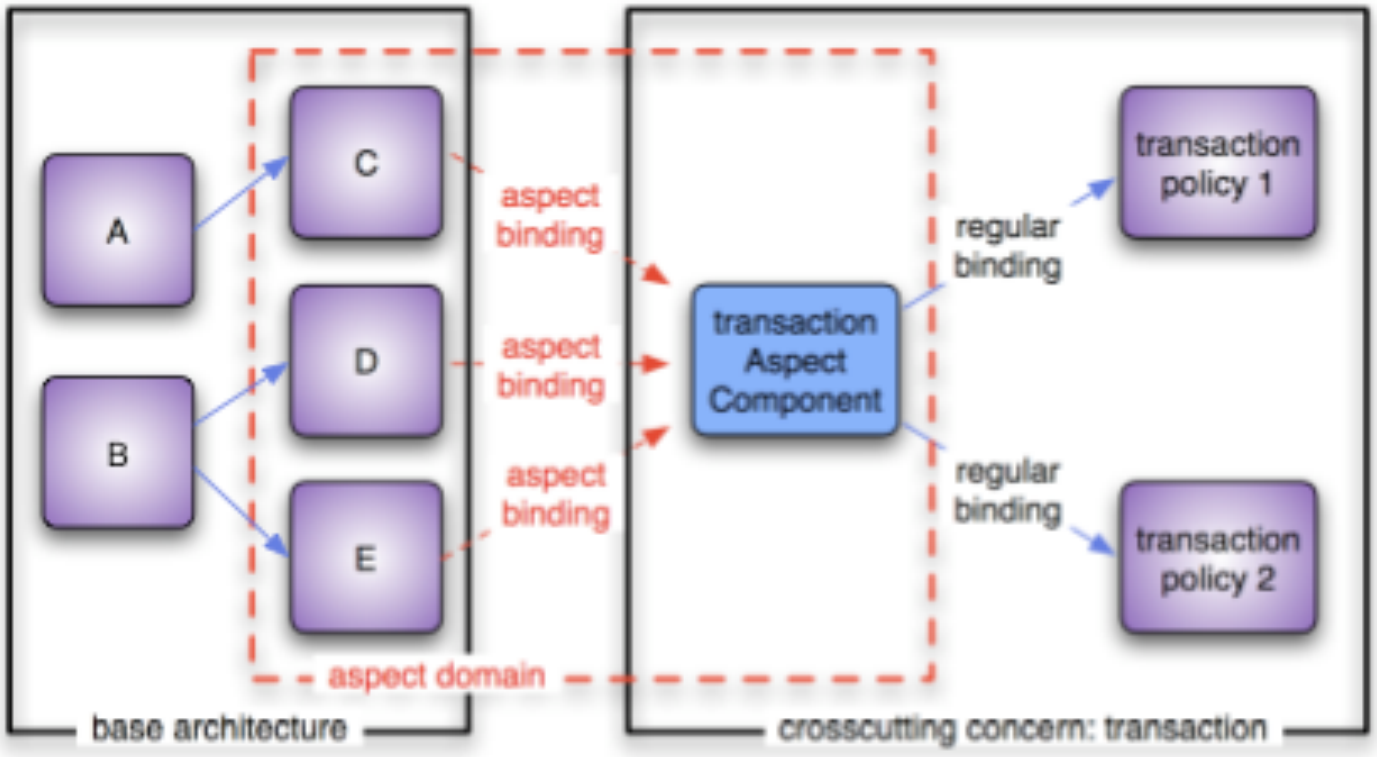
	JBoss AOP	Spring AOP	CAM/DAOP	FuseJ
Critères du point de vue composant et architecture				
ADL	non	non	oui	oui*
Général	non	non	oui	non
Réflexif	non	non	non	non
Extensible	non	oui	non	non
Hiérarchique	non	non	non	oui**
Critères du point de vue aspect				
Symétrie d'élément	non	oui	non	oui
Symétrie de placement	oui	oui	oui	oui
Symétrie de portée	non	non	non	non

** Le langage de composition et de spécification de FuseJ*
*** La hiérarchie est possible, mais chaque composite ne compose que deux composants à la fois.*

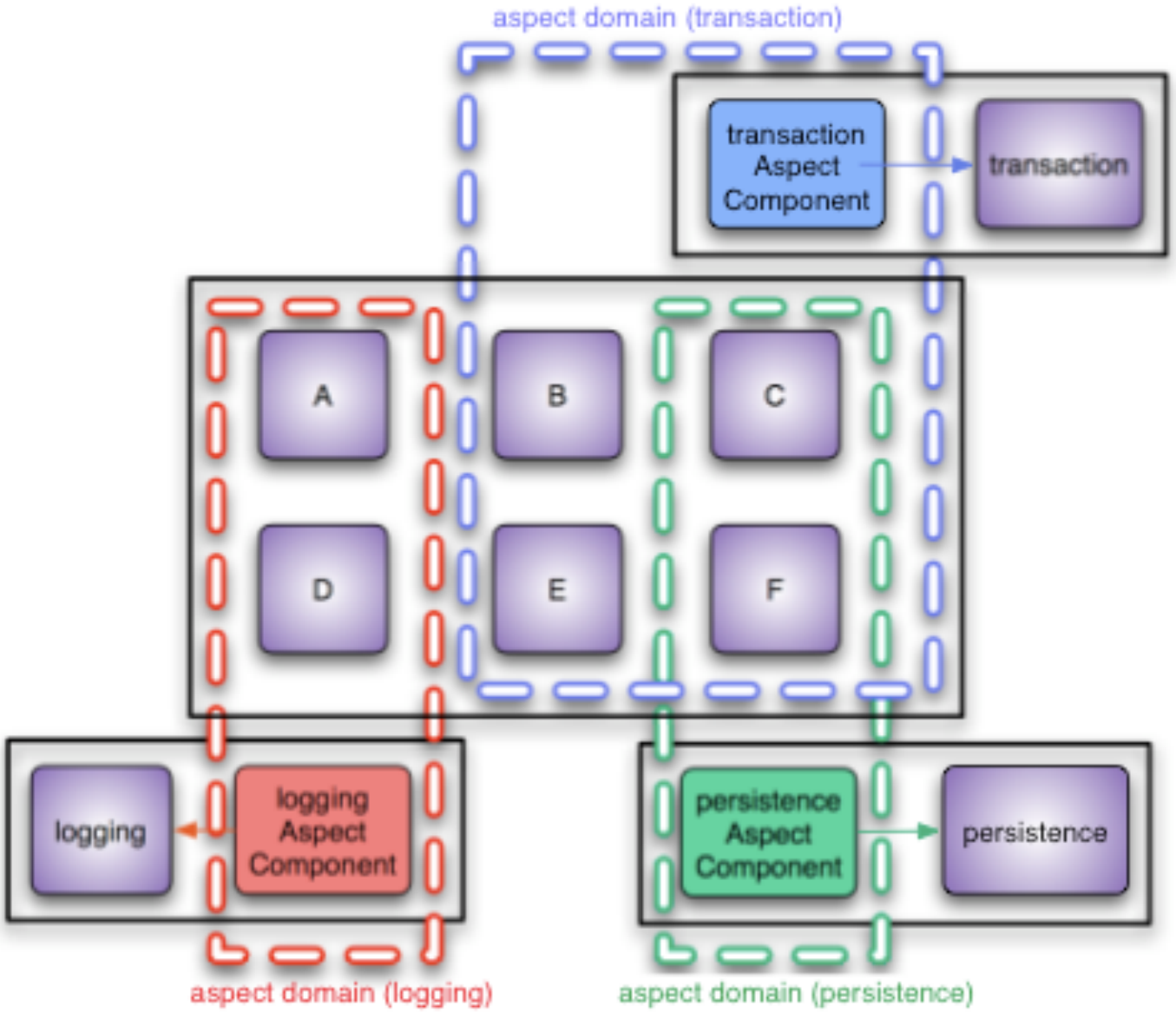
FAC: Fractal Aspect Component - Metamodel



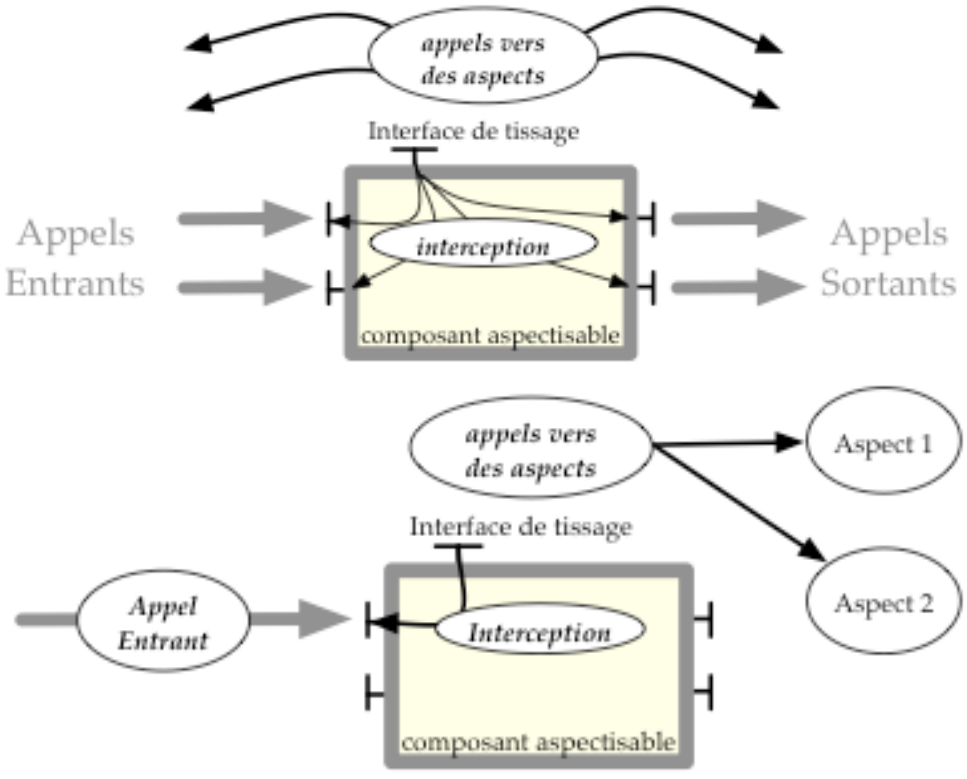
FAC: Fractal Aspect Component - Aspect Binding



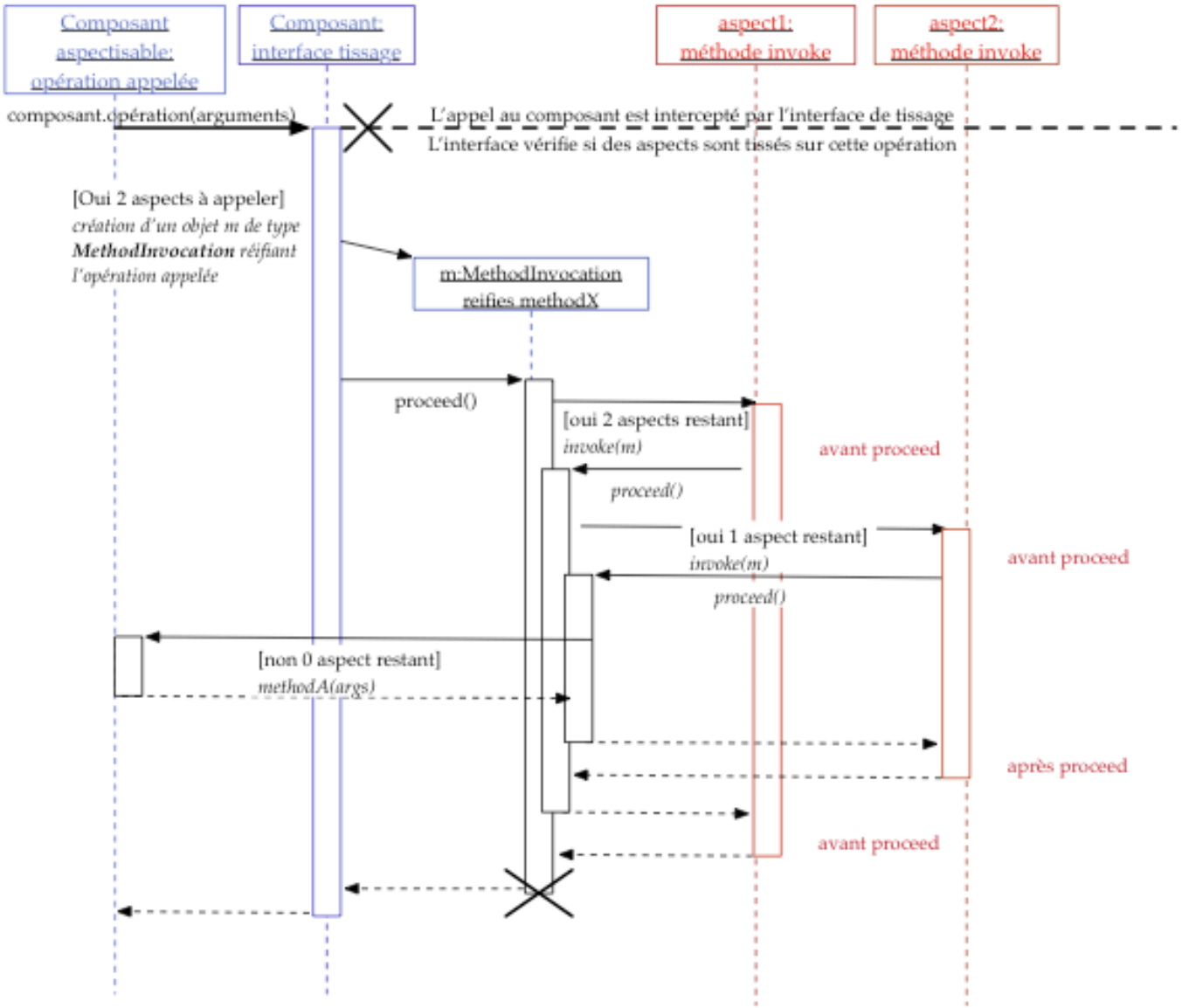
FAC: Fractal Aspect Component - Aspect Domains



FAC: Fractal Aspect Component - Join Point Model (1)



FAC: Fractal Aspect Component - Join Point Model (2)



FAC: Fractal Aspect Component - Pointcut Language

```
1  pcd::= <jp_type> <component>;<interface>;<method>
2  jp_type::= CLIENT | SERVER | BOTH
3  component::= une expression régulière sur le nom des composants
4  interface::= une expression régulière sur le nom des interfaces
5  method::= une expression régulière sur le nom des opérations
```

Pointcut Expressions	Captured Elements
`*;*;deposit*:void`	Every incoming and outgoing method returning void that start with deposit in any component and interface
`CLIENT B;*;deposit*`	Every outgoing method named deposit in any interface of a component named B
`SERVER B;ITransfert;*`	Every incoming method in ITransfert interface of a component B

FAC: Fractal Aspect Component - Weaving Interface (1)

```
interface WeavingInterface {
// Partie gestion des liaisons d'aspect
/** Etablit une liaison d'aspect
 * @param regExp une expression régulière sélectionnant les opérations à intercepter
 * @param aspect l'aspect auquel le composant se lie
 */
void setAspectbinding(ItfPointcutExp regExp, Component aspect);
/** Rompt une liaison d'aspect vers un aspect
 * @param aspect l'aspect concerné
 */
void unsetAspectbinding(Component aspect);

// Partie ordonnancement
/** Change l'ordre d'un aspect.
 * @param acName le nom de l'aspect
 * @param oldPosition l'ancienne position
 * @param newPosition la nouvelle position
 */
void changeACorder(String methodName, int oldPosition, int newPosition);
/** Donne la position d'un aspect sur une opération
 * @param ac le nom de l'aspect
 * @param methodName le nom de l'opération concernée
 */
int[] getACPosition(AspectComponent ac, String methodName);

// Partie verrouillage
/** Ouvre l'accès à une opération
 * @param op l'opération concernée
 * @param itf l'interface concernée
 */
void openAccess(Operation op, Interface itf);
/** Verrouille l'accès à une opération
 * @param op l'opération concernée
 * @param itf l'interface concernée
 */
void reduceAccess(Operation op, Interface itf);
/** Ouvre l'accès à une opération avec une politique particulière
 * @param op l'opération concernée
 * @param itf l'interface concernée
 * @param lvl le niveau d'ouverture de l'opération
 */
void openAccessTo(Operation op, Interface itf, OpennessLevel lvl);

// Partie gestion tissage
/** Tisse un aspect sur un ensemble de composants
 * @param root le composant racine de la navigation
 * @param regExp la coupe sélectionnant les opérations à intercepter
 * @param aspect l'aspect à tisser
 * @param aDomain le nom du domaine d'aspect à créer
```

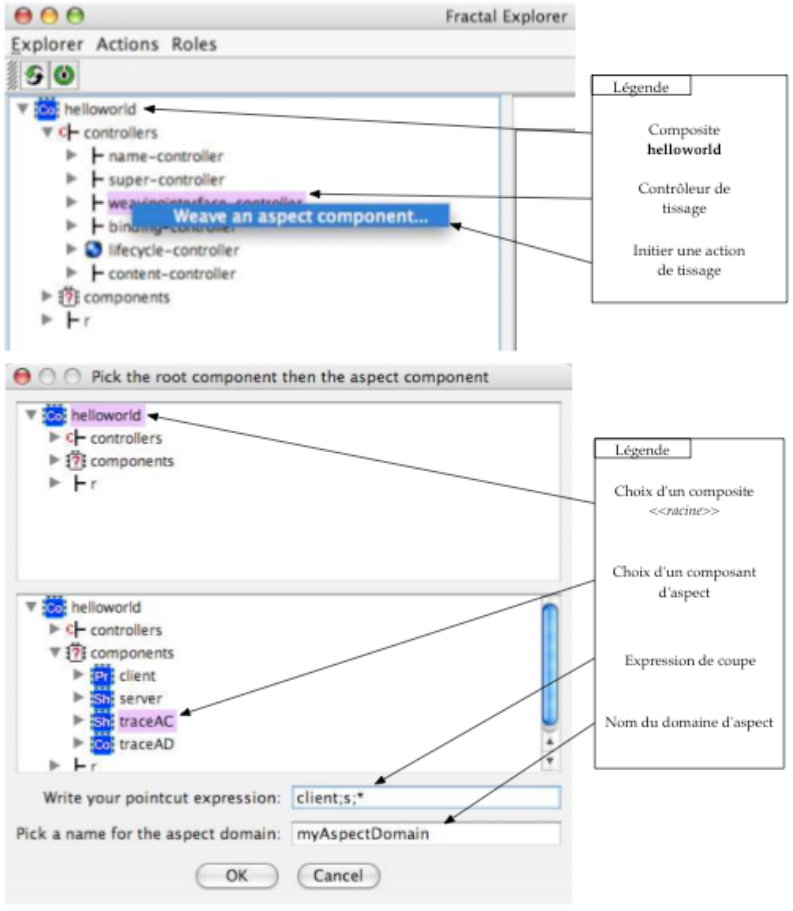
FAC: Fractal Aspect Component - Weaving Interface (2)

```
50  */
51  void weave(Component root, Pointcut pcut, Component aspect, String aDomain);
52  /** Opération inverse du tissage
53   * @param aspect l'aspect à dé-tisser
54   * @param
55   */
56  void unweave(Component aspect);
57
58  // Partie introspection de coupe
59  /** Donne la liste des aspects tissés sur le composant
60   * @return un tableau contenant les références vers ces aspects (composants)
61   */
62  Component[] listAC();
63  /** Donne la liste des composants possédant des opérations liées à l'aspect passé en paramètre.
64   * L'architecture est introspectée à partir du composant racine (root).
65   * @param root le composant racine à partir duquel initier la recherche
66   * @param ac l'aspect recherché
67   * @return un tableau de références vers les composants concernés
68   */
69  Component[] listCrosscutComps(Component root, Component ac);
70  /** Donne une représentation sous forme d'arbre des opérations, interfaces et composant
71  * potentiellement concernés par la coupe passée en paramètre. Comme l'opération précédente,
72  * la recherche récursive démarre d'un composant racine.
73  * @param root le composant racine
74  * @param pcut la coupe
75  * @return une représentation objet des opérations, interfaces et composants concernés
76  */
77  PointcutRepresentation aspectizableComps(Component rComp, ItfPointcutExp pcut);
78 }
```

FAC: Fractal Aspect Component - Weaving using Fractal-ADL

```
01 <definition name="C">
02 <component name="traceAC"/>
03    <interface name="ACI" role="server"  signature="AspectComponent"/>
04 </component>
05 <component name="A"/>
06    <interface name="itf1" role="client"  signature="Itf1"/>
07 </component>
08 <component name="B"/>
09    <interface name="itf1" role="server"  signature="Itf1"/>
10 </component>
11 <binding client="A.itf1" server="B.itf1"/>
12 <weaving ac="traceAC" pcd="*;*;s*:void"  rootComp="this" adomain="D"/>
13 </definition>
```

FAC: Fractal Aspect Component - Weaving using Fractal-Explorer

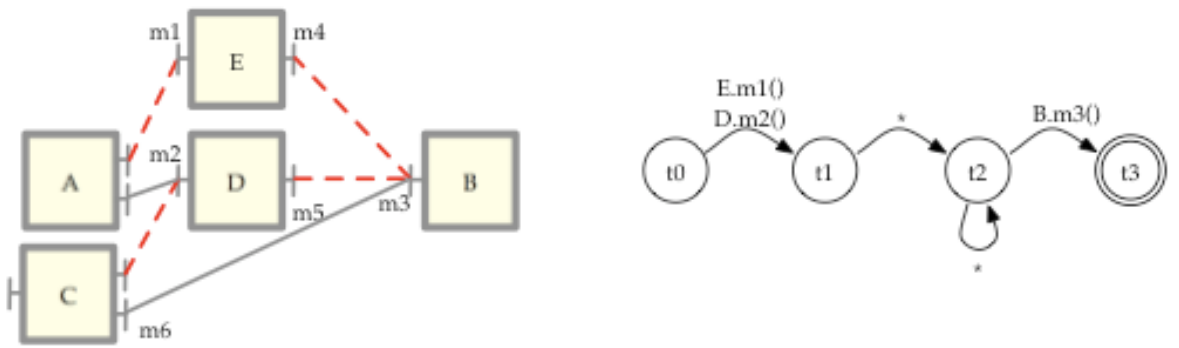


FAC: Fractal Aspect Component - cflow Pointcut

```
1 tracematchespcd::=tracematchespcd .tracematchespcd
2                  | (tracematchespcd)
3                  | tracematchespcd+tracematchespcd
4                  | tracematchespcd
5                  | #
6  #::= désigne une étoile : 0 à n transitions quelconques
7  pcd::= jp_type  component;interface;method
8  jp_type::= CLIENT | SERVER | BOTH
9  component::= une expression régulière sur le nom des composants
10 interface::= une expression régulière sur le nom des interfaces
11 method::= une expression régulière sur le nom des opérations
```

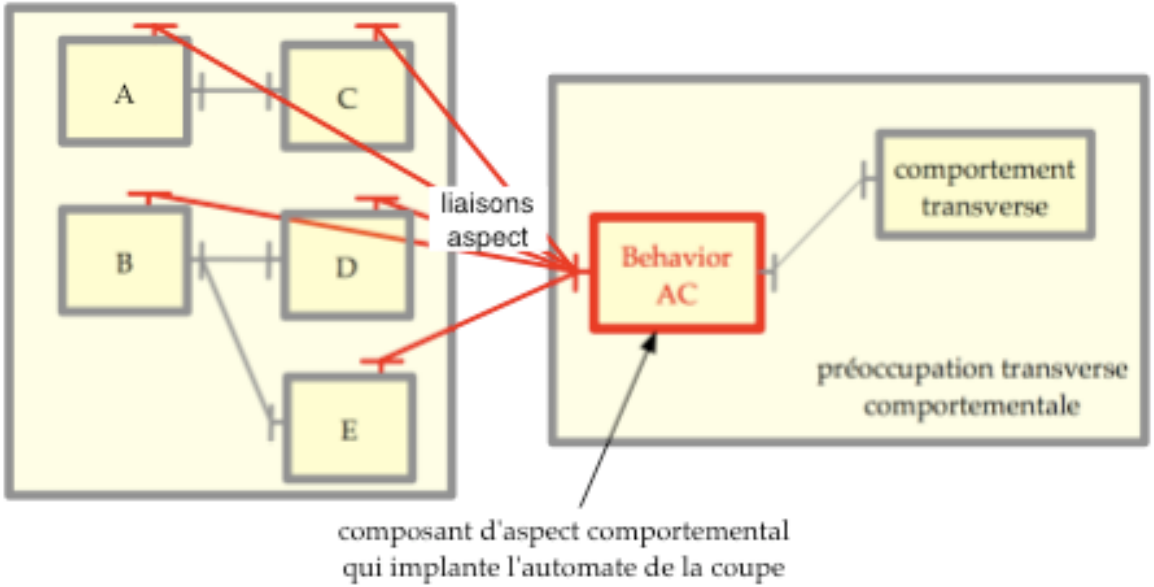
Examples:

$$A.m_1*; b.m_2*$$

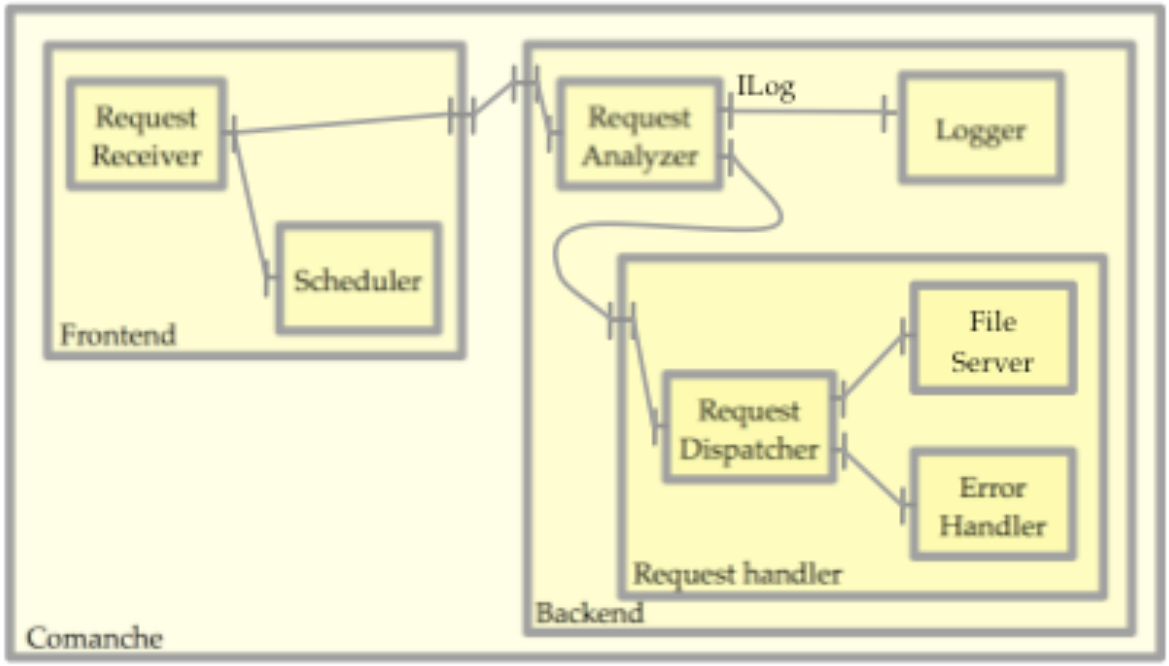


$$E.m_1() + D.m_2(); *; b.m_3()$$

FAC: Fractal Aspect Component - cflow Pointcut Implementation



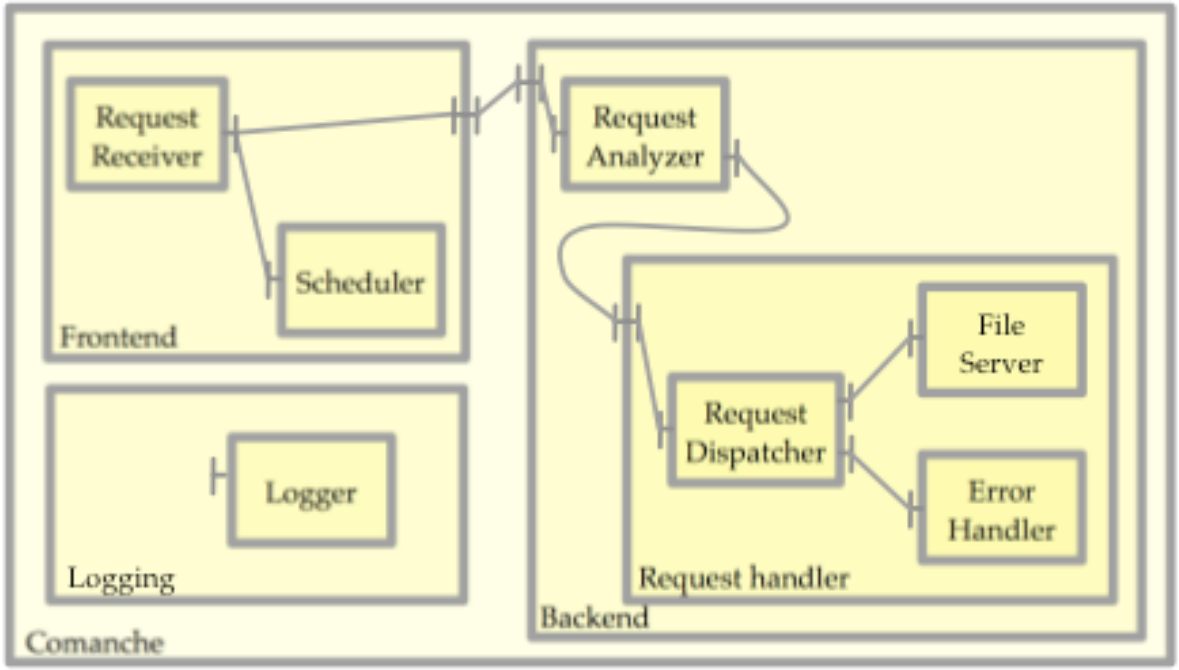
FAC: Example - Comanche



FAC: Example - Step 1 : Aspect Definition

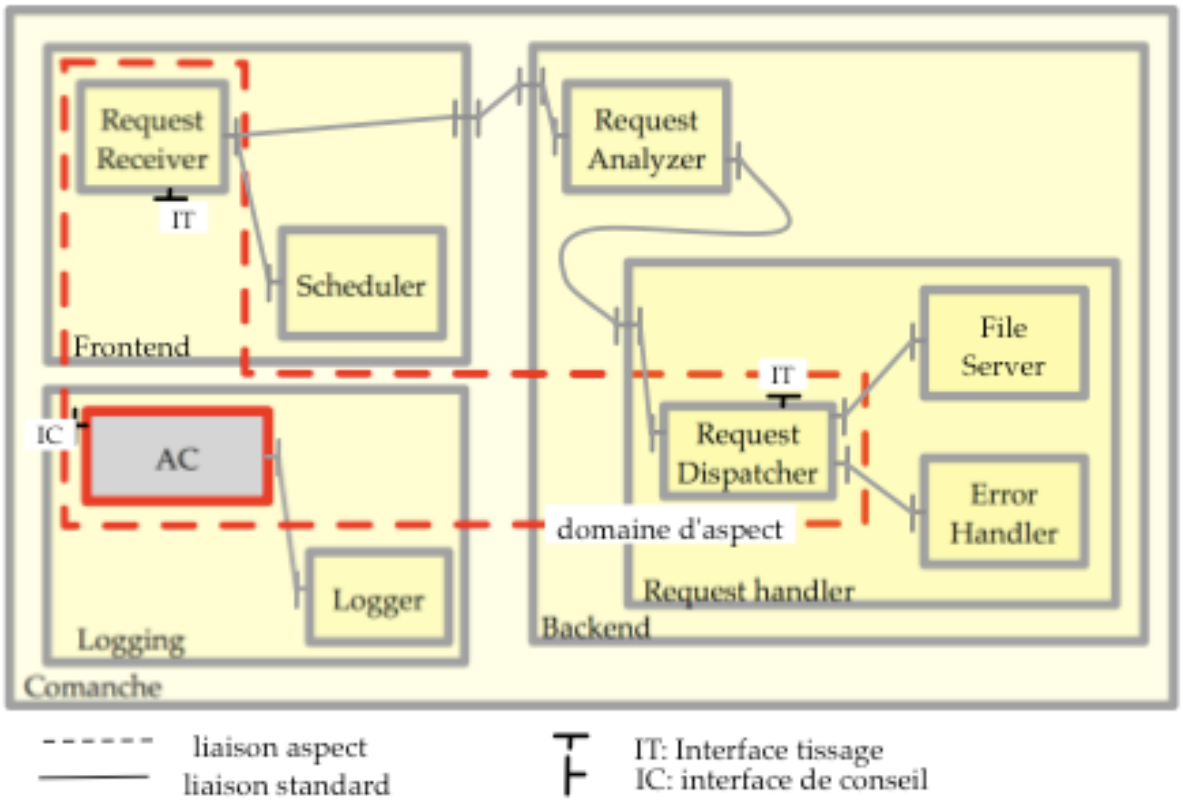
```java
/** Interface de journalisation */
@Interface(name="logger")
public interface ILogger {  void log (String msg); }

/** Composant de journalisation sur la sortie standard */
@FractalComponent
public class Logger implements Logger {
  public void log (String msg) { System.out.println(msg); }
}
```

```xml
<definition name="comanche.LoggingComposite" >
  <component name="logger">
      <interface name="logger" signature="comanche.ILogger" role="server"/>
  </component>
  <component name="loggingAC">
      <interface name="aspectComponent" signature="fac.AdviceInterface" role="server" />
      <interface name="logger" signature="comanche.ILogger" role="client"/>
  </component>
  <binding client="loggingAC.logger" server="logger.logger" />
</definition>
```

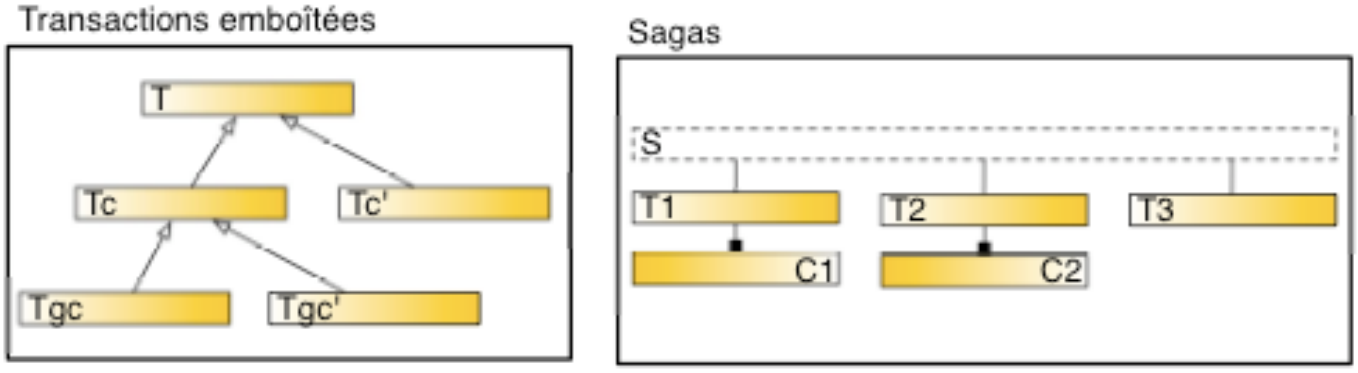


FAC: Example - Step 2 : Aspect Weaving

```
1<definition name="comanche.Comanche" >
2  <interface name="r" signature="java.lang.Runnable" role="server"
3               cardinality="singleton" contingency="mandatory"/>
4  <component name="frontend" definition="comanche.FrontendComposite" />
5  <component name="backend" definition="comanche.BackendComposite" />
6  <component name="logging" definition="comanche.LoggingComposite" />
7  <binding client="this.r" server="frontend.r" />
8  <binding client="frontend.rh" server="backend.rh" />
9  <weave root="this" ac="loggerAC" pointcutExp="request*;rh;*"/>
10</definition>
```



FAC: Case Study 1: Long Lived Transactions : Presentation



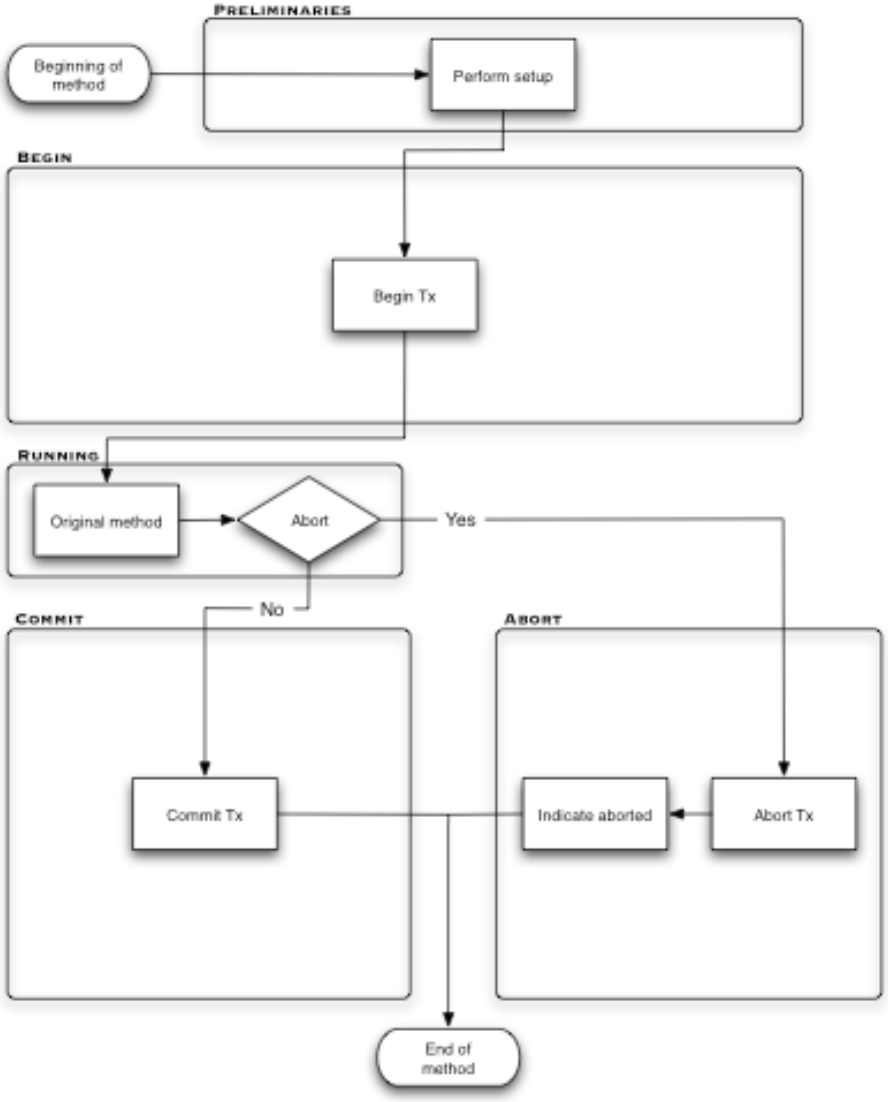
FAC: Case Study 1: Long Lived Transactions : Problem (1)

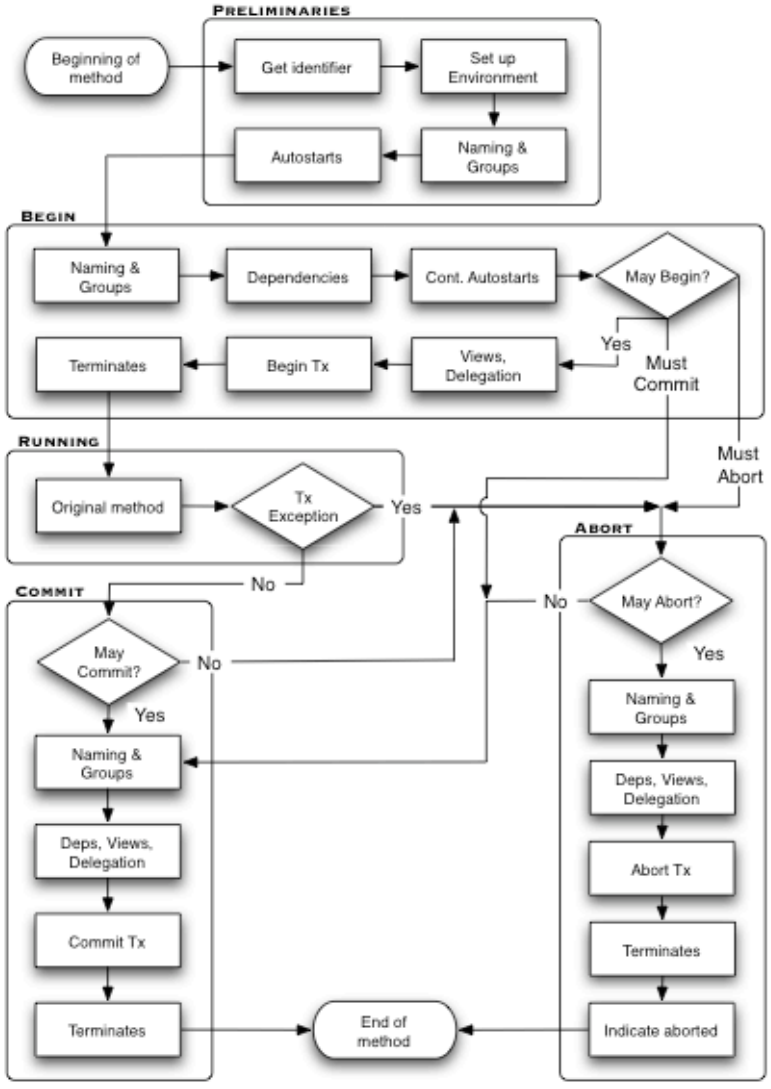
```
util.strategy.Hierarchical.childMethod() {
  alias(root <"parent"> );
  begin {
     dep(self wd root, root cd self);
     view(self, parent) }
     commit { del(self, parent) }
}
```

Problem :

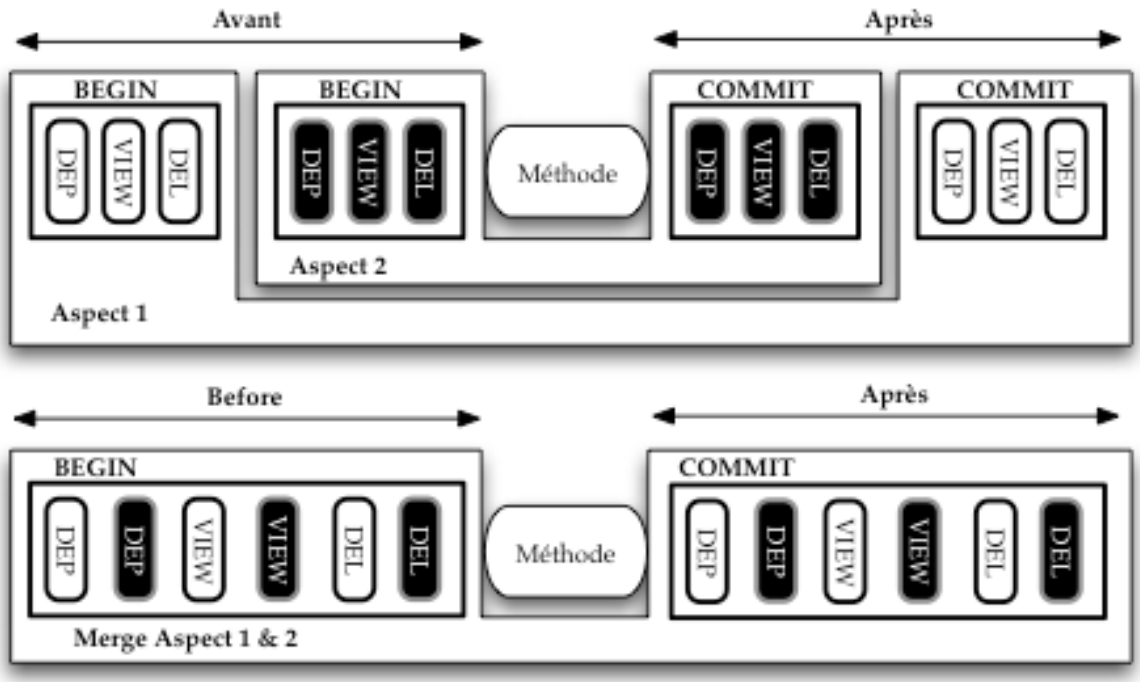
```
[...]
begin { [...]
   del(donor, self)}
commit { [...]
   del(self, donor)}
abort { [...]
   del(self, compensator)}
```

FAC: Case Study 1: Long Lived Transactions : Problem (2)





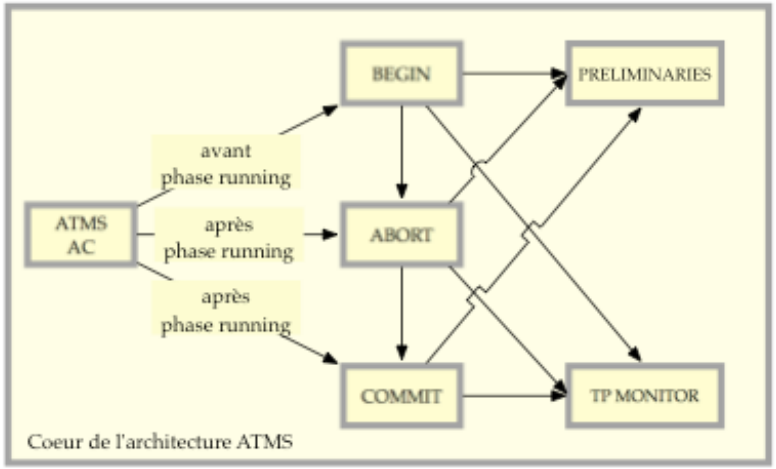
FAC: Case Study 1: Long Lived Transactions : Problem (3)



FAC: Case Study 1: Long Lived Transactions : KALA Solution

```
util.strategy.Hierarchical.childMethod() {
    alias(root <"parent"> );
    begin { dep(self wd root, root cd self) }
}
util.strategy.Hierarchical.childMethod() {
    alias(root <"parent"> );
    begin { view(self, parent) }
}
util.strategy.Hierarchical.childMethod() {
    alias(root <"parent"> );
    commit { del(self, parent) }
}
```

FAC: Case Study 1: Long Lived Transactions : FAC Solution (1)



client ⟶ server

Liaison classique

```
1 @AspectComponent
2 public class AtmsAC implements AroundAspectComponent {
3     @Requires(name = "abort") private IAbort abort;
4     @Requires(name = "commit") private ICommit commit;
5     @Requires(name = "begin") private IBegin begin;
6     @Requires(name ="preliminaries") private IPreliminaries preliminaries;
7     // code de l'advice de type autour
8     public Object invoke(FcMethodInvocation m) throws Throwable {
9       Object retval = null;
10      // fixe l'environnement de la transaction
11      preliminaries.setEnvironnement();
12      // phase de begin
13      if (begin.begin()) // may begin and begin
14        try {
15            // phase de running
16            retval = m.proceed();
17        } catch (TxException ex) {  abort.abort(tx_id); // phase d'abort }
18        commit.commit(tx_id); // phase de commit
19        return retval;
20    }
21 }
```

FAC: Case Study 1: Long Lived Transactions : FAC Solution (2)

```
1 @FractalComponent(controllerDesc = "aspectizableComponent")
2 public class Begin implements IBegin {
3    @Requires(name = "tpmonitor") private static ITPMonitor txmgr;
4    @Requires(name = "abort") private IAbort abort;
5    public Boolean begin(Integer tx_id) throws TxException {
6       // may begin?
7       Forcing beginForcing = txmgr.mayBegin(tx_id);
8       Boolean runTx = true;
9       if (beginForcing != null) {
10         if (beginForcing.direction.equals("Abort"))
11            try {
12               abort.abort(tx_id);
13            } catch (TxException e) {
14               throw e;
15            }
16            else
17               runTx = false;
18      }
19      // begin
20      if (runTx) {
21         txmgr.begin(tx_id);
22      }
23      return runTx;
24   }
25 }
```

FAC: Case Study 1: Long Lived Transactions : FAC Solution (3)

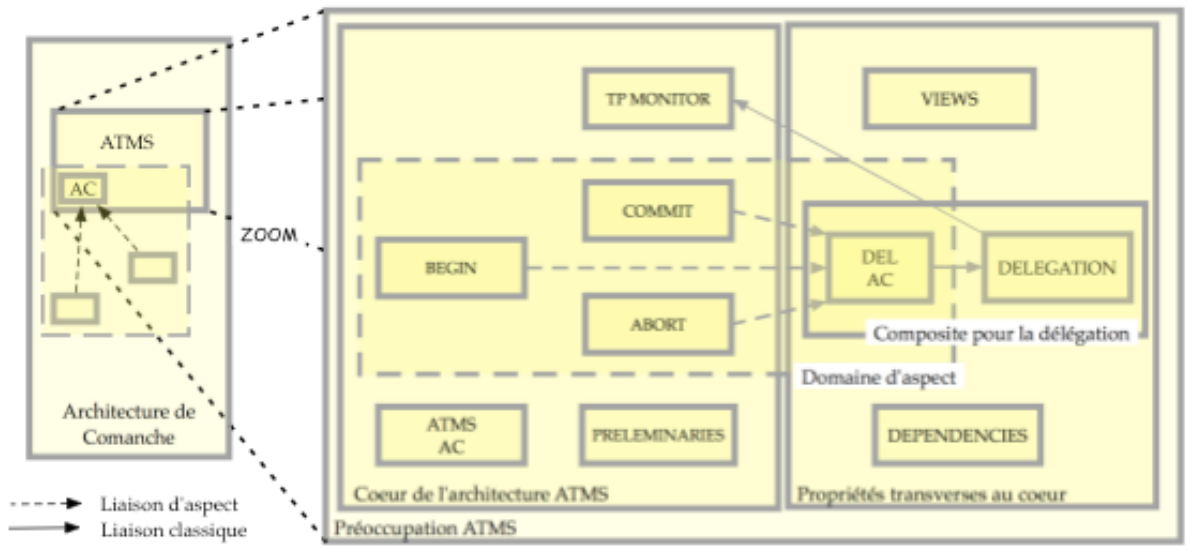
```
@FractalComponent(controllerDesc = "aspectizableComponent")
public class Commit implements ICommit {
   @Requires(name = "tpmonitor") private static ITPMonitor txmgr;
   @Requires(name = "abort") private IAbort abort;
   public void commit(Integer tx_id) throws TxException {
      //  May Commit
      Forcing commitForcing = txmgr.mayCommit(tx_id);
      if (commitForcing != null) {
         abort.abort(tx_id);
      }
      try {
         txmgr.commit(tx_id);
      } catch (TxException e) {
         abort.abort(tx_id);
      }
   }
}
```

```
<definition name="core.ATMSComposite" >
  <!-- composants de l architecture coeur -->
  <component name="atmsaspect" definition="core.AtmsAC" />
  <component name="environment" definition="core.Preliminaries" />
  <component name="begin" definition="core.Begin" />
  <component name="commit" definition="core.Commit" />
  <component name="abort" definition="core.Abort" />
  <component name="tpmonitor" definition="core.TPMonitor" />
  <!-- déclarations de liaisons -->
  <binding client="atmsaspect.begin" server="begin.begin" />
  (...)
</definition>
```

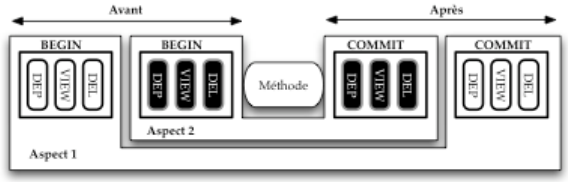
```
@FractalComponent
public class Delegation implements IDelegation {
   // Required interfaces
   @Requires(name = "tpmonitor") private static ITPMonitor txmgr;
   @Requires(name = "preleminaries") private IPreleminaries preleminaries;
   // 2 attributs de composant
   @Attribute private String delsource;
   @Attribute private String deltarget;
   // Implementation of the IDelegation interface
    public void performDelegation() throws TxException {
         ...
    }
}
```

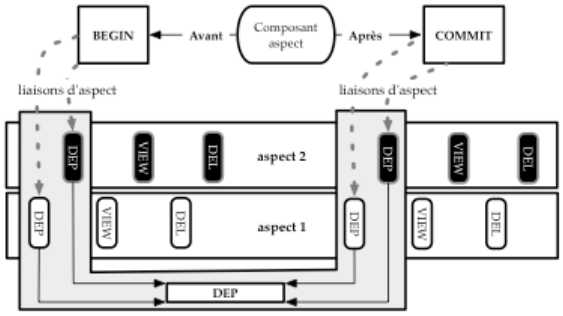
FAC: Case Study 1: Long Lived Transactions : FAC Solution (4)

```
1<definition name="subconcern.StructureManagement" >
2   <!-- composants spécifigues définissant des propriétés -->
3   <component name="dep" definition="additional.Dep(self;wd;parent,parent;cd;self)" />
4   <!-- weaving declarations -->
5   <weave root="this" ac="dep" pointcutExp="begin;begin;begin*" aDomain="depAD" />
6</definition>
```



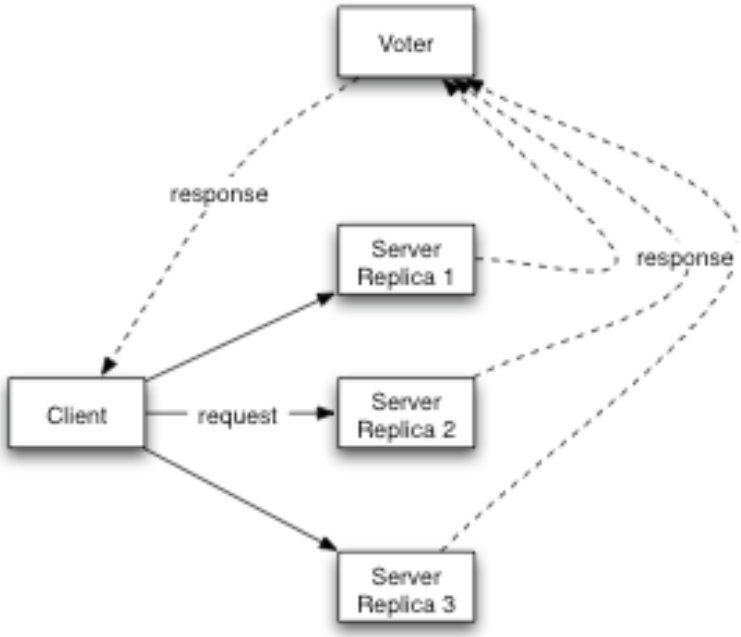
FAC: Case Study 1: Long Lived Transactions : FAC Solution (5)

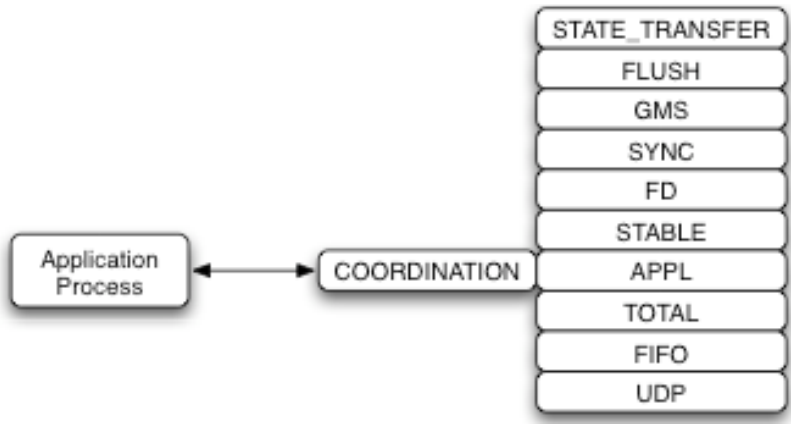




Critère	KALA	FAC
Cycle d'une transaction		
Représentation	Bloc de code KALA	Composant
Ordonnancement	Tisseur KALA	Composant d'aspect `ATMS AC`
Propriété		
Représentation	Ligne de code KALA	Composant d'aspect déléguant éventuellement à un composant
Configuration	paramètre d'une fonction del, dep, ou view	Attribut de composant configurable au niveau du fichier ADL
Sous-préoccupations)		
Représentation	Fichier KALA	Fichier FRACTAL-ADL avec les composants et leur tissage
Séparation des préoccupations		
Propriétés	+	+
Cycle d'une transaction	+	+
Base/aspect GCS	+	+

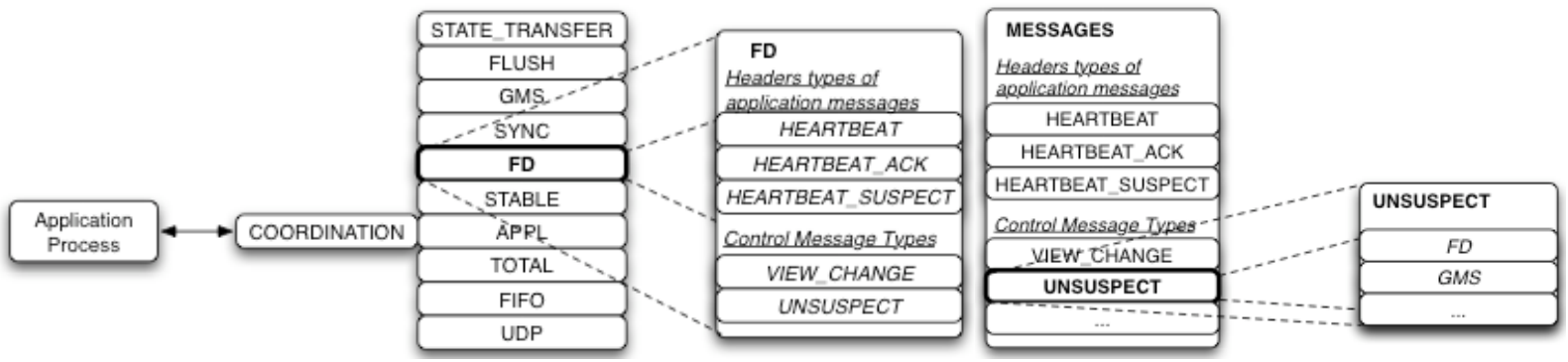
FAC: Case Study 2: Group Communication Management : Presentation



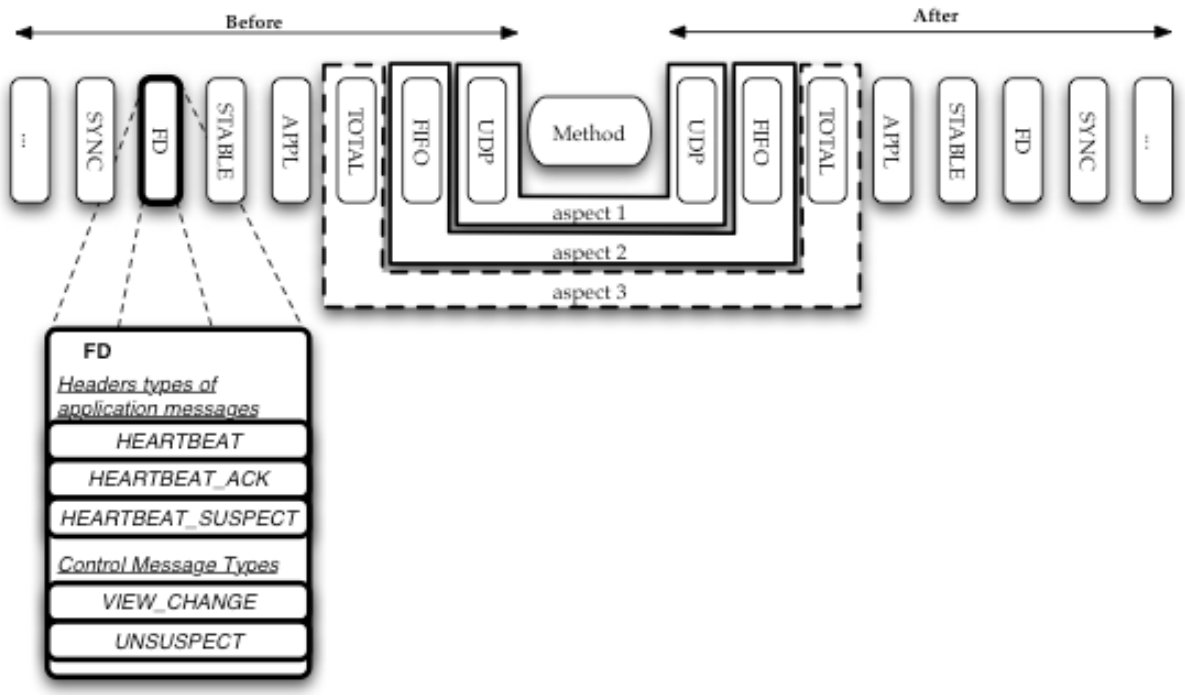


FAC: Case Study 2: Group Communication Management : Problem

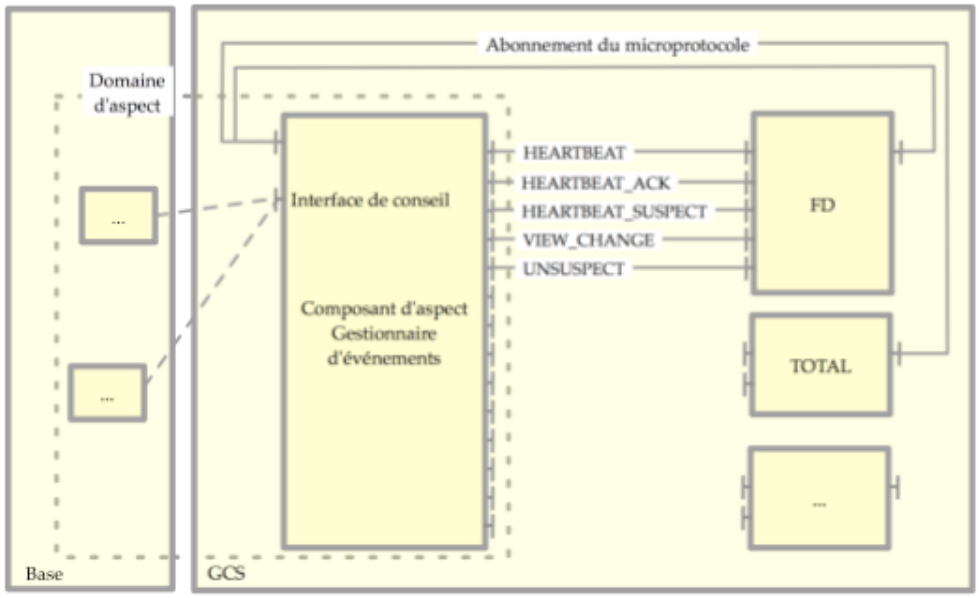
```
1  String props="UDP:PING:FD:STABLE:NAKACK:UNICAST:" +
2               "FRAG:FLUSH:GMS:VIEW_ENFORCER:STATE_TRANSFER:QUEUE";
3  Message send_msg;
4  Object  recv_msg;
5  Channel channel=new JChannel(props);
6  channel.connect("MyGroup");
7  send_msg=new Message(null, null, "Hello world");
8  channel.send(send_msg);
9  recv_msg=channel.receive(0);
10 System.out.println("Received " + recv_msg);
11 channel.disconnect();
12 channel.close();
```



FAC: Case Study 2: Group Communication Management : Problem



FAC: Case Study 2: Group Communication Management : FAC Solution



Liaison d'aspect - - - -
Liaison standard ———

Critère	JGroups	FAC
Microprotocole		
Représentation	Classe	Composant
Gestion	Pile	Assemblage de composants
Communication entre microprotocoles	Pull/push sur les messages passant par la pile	Lien entre deux interfaces
Message		
Représentation	Bloc *switch* dans le code des microprotocoles	Interface de composant
Gestion	Eparpillée dans chaque microprotocole	centralisée dans le composant d'aspect
Séparation des préoccupations		
Microprotocoles	+	+
Gestion messages	-	+
Base/aspect GCS	-	+