Génération de code à partir de modèles à composants (une instrumentation)

Pascal ANDRE, Gilles ARDOUREL

AELOS / LINA – UMR CNRS 6241
{Firstname.Lastname}@univ-nantes.fr

Exposés AeLoS

Plan de l'exposé

Plan

Introduction / contexte

Contexte : Implantation, animation, test avec Kmelia

- animation, support de tests
- intégration support d'exécution
- aspects dynamiques (cohérence des échanges, pas de blocage...)
- aspects fonctionnels (calculs et effets sur la dynamique - au moins les gardes)

$\Longrightarrow$ revoir l'exposé exposé Gencode (`gencode.pdf`) : contexte, bibliographie...

Introduction / cible

Kmelia (boîte à outils COSTO) :

- structure : Composants/Interfaces/Service/Architectures/Composites
- calculs : actions
- dynamique : LTS + communication
  - \+ infrastructures : EJB, CORBA, SOAP...
  - \- Java support : RMI, JMS
  - \- Java basic : Class, threads
- Assemblages et Composites
  - \+ système / programme
  - \- connecteurs / communications

Kmelia (Sémantique opérationnelle) + traçabilité

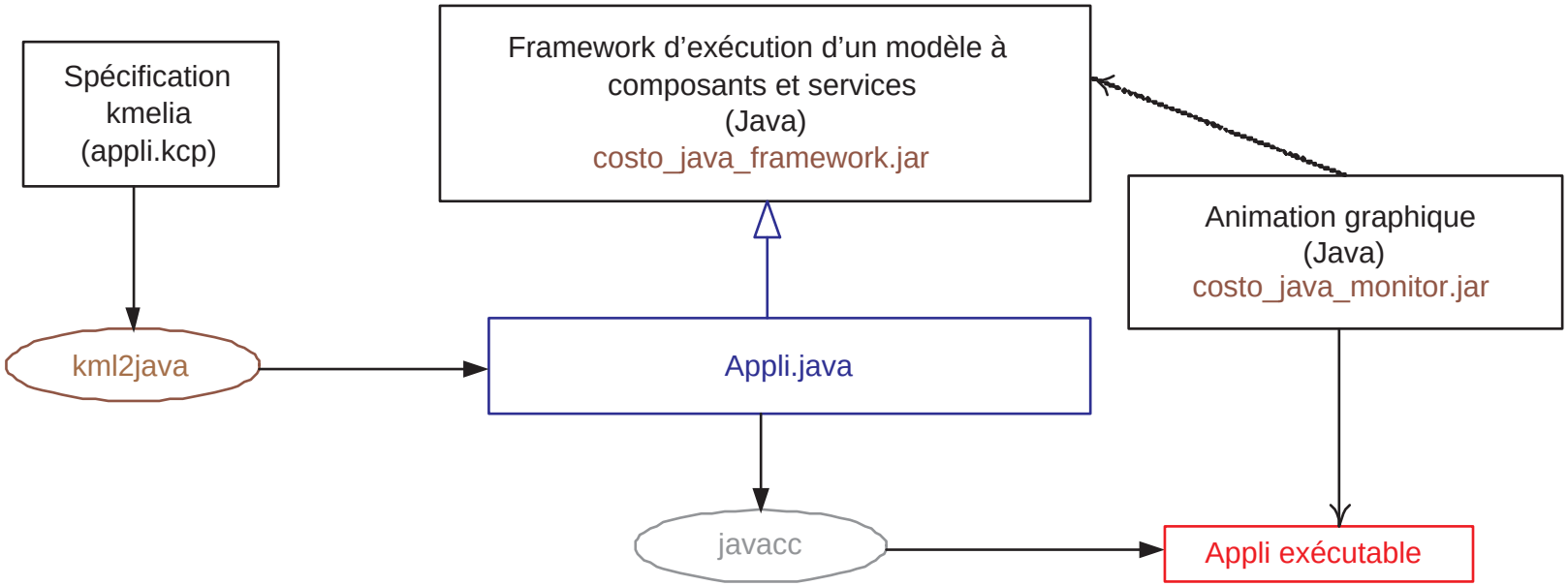
Introduction / focus

Paradigmes à l'exécution (traçabilité) :

- Données & actions
  - types de base, fonctions et mappings
  - types utilisateur, fonctions et mappings
  - expressions de base
  - communications et appels de service
- composants et services
  - composant = structure + process (autorun, terminate)
  - service offert = LTS + process
  - service requis = passerelle
- assemblage
  - canaux & synchronisation
  - mappings de données et messages
  -
- Composites
  - Promotion channel
  - Application

Plan

Kmelia/Java Architecture



Framework + génération de code = application exécutable

Implantation d'un modèle Kmelia avec Java 1/2

Plugin `Kml2Java`

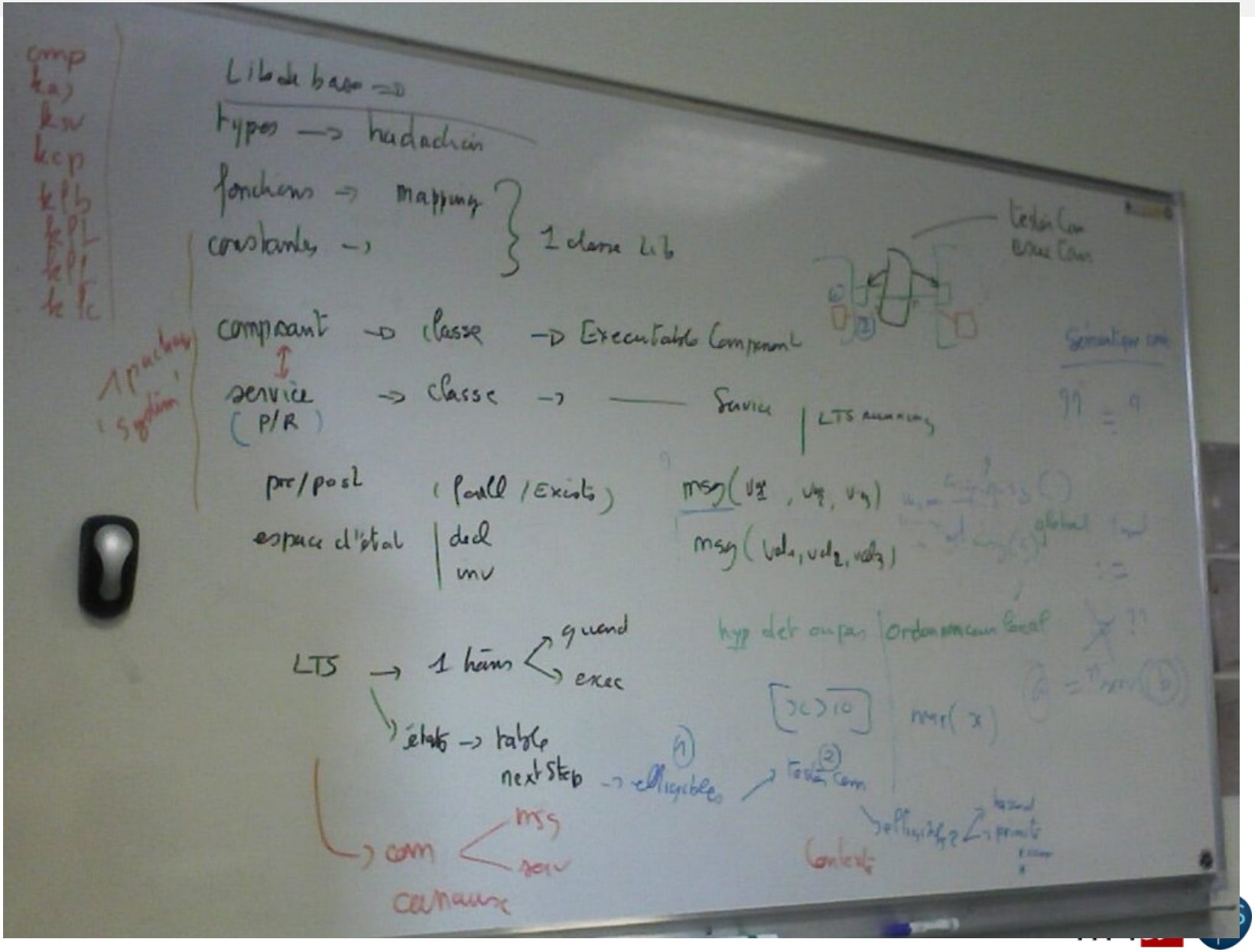
- Types et bibliothèques
  - Types de base : traduction *ad-hoc*
  - Types utilisateurs : problème de conflits et surcharge de types
  - Constantes, fonctions $\Longrightarrow$ 1 classe `ProjectLib`
- Assembly $\Longrightarrow$ 1 package `System` 1 classe `EntryPoint`
- Composant $\Longrightarrow$ 1 classe `ExecutableComponent`
  - espace d'état $\Longrightarrow$ variables + invariant
  - services (après)
  - ordonnanceur ?
- Service P/R $\Longrightarrow$ 1 classe `ExecutableService`
  - pre/post $\Longrightarrow$ prédicats
  - LTS $\Longrightarrow$ 1 classe `ProjectLib`
    - transition $\Longrightarrow$ (1) garde - (2) exec
    - état $\Longrightarrow$ table + `nextStep` (elligibles $\rightarrow$ testercom $\rightarrow$ elligibles2 $\rightarrow$ priorités (random, com first...)
    - ordonnanceur = *thread* / déterminisme

Implantation d'un modèle Kmelia avec Java 2/2

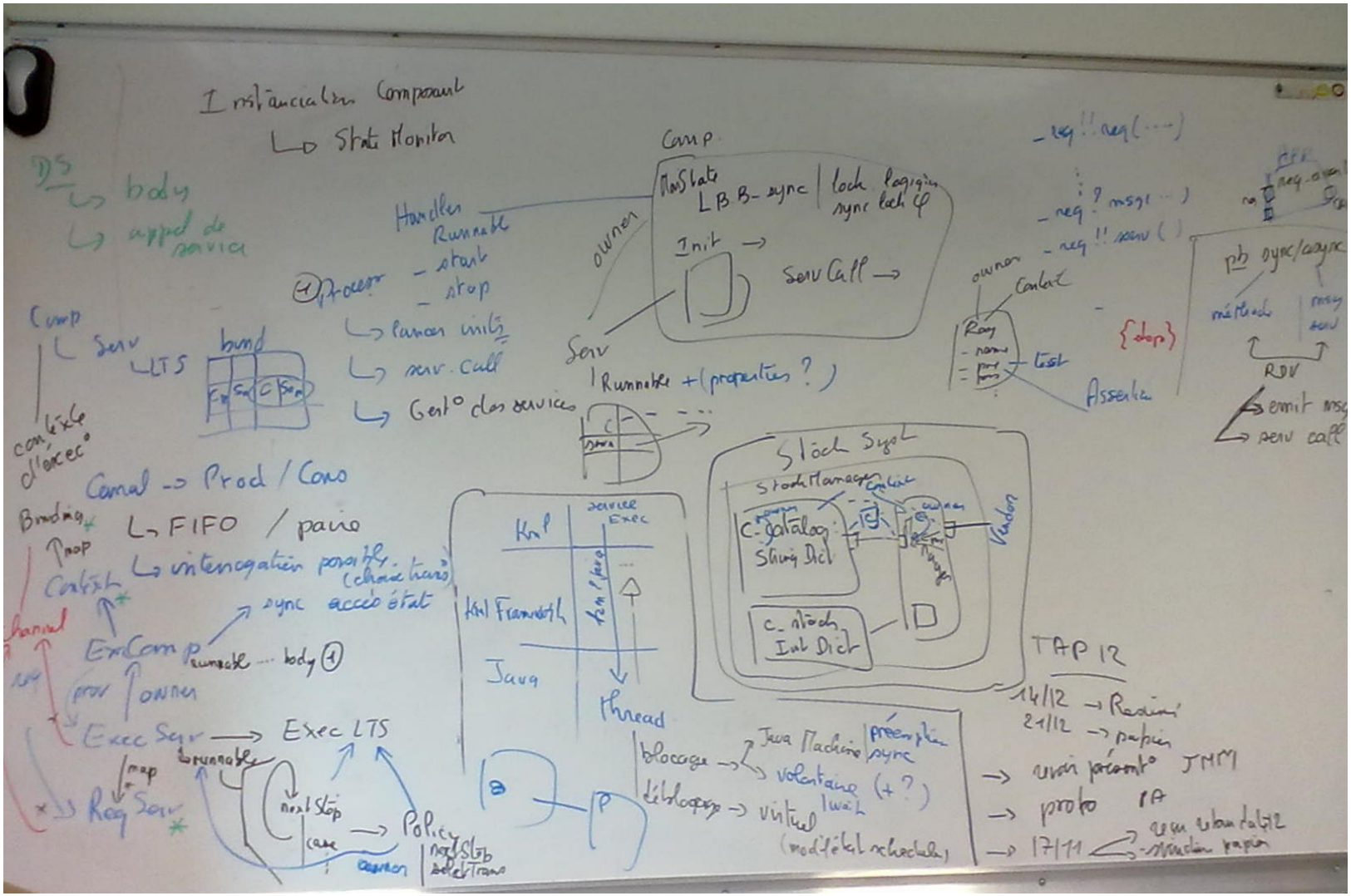
Plugin `Kml2Java`

- Communications -> Framework
  - canal $\Longrightarrow$ classe
    - transformation des com $\Longrightarrow$ garde + exec
    - tester com
    - exécuter synchro + com
    - ordonnanceur = *thread*
  - échange paramètres
- création / arrêt de services
- etc.

Kmelia/Java (version initiale)



Kmelia/Java (version initiale suite)



Plan

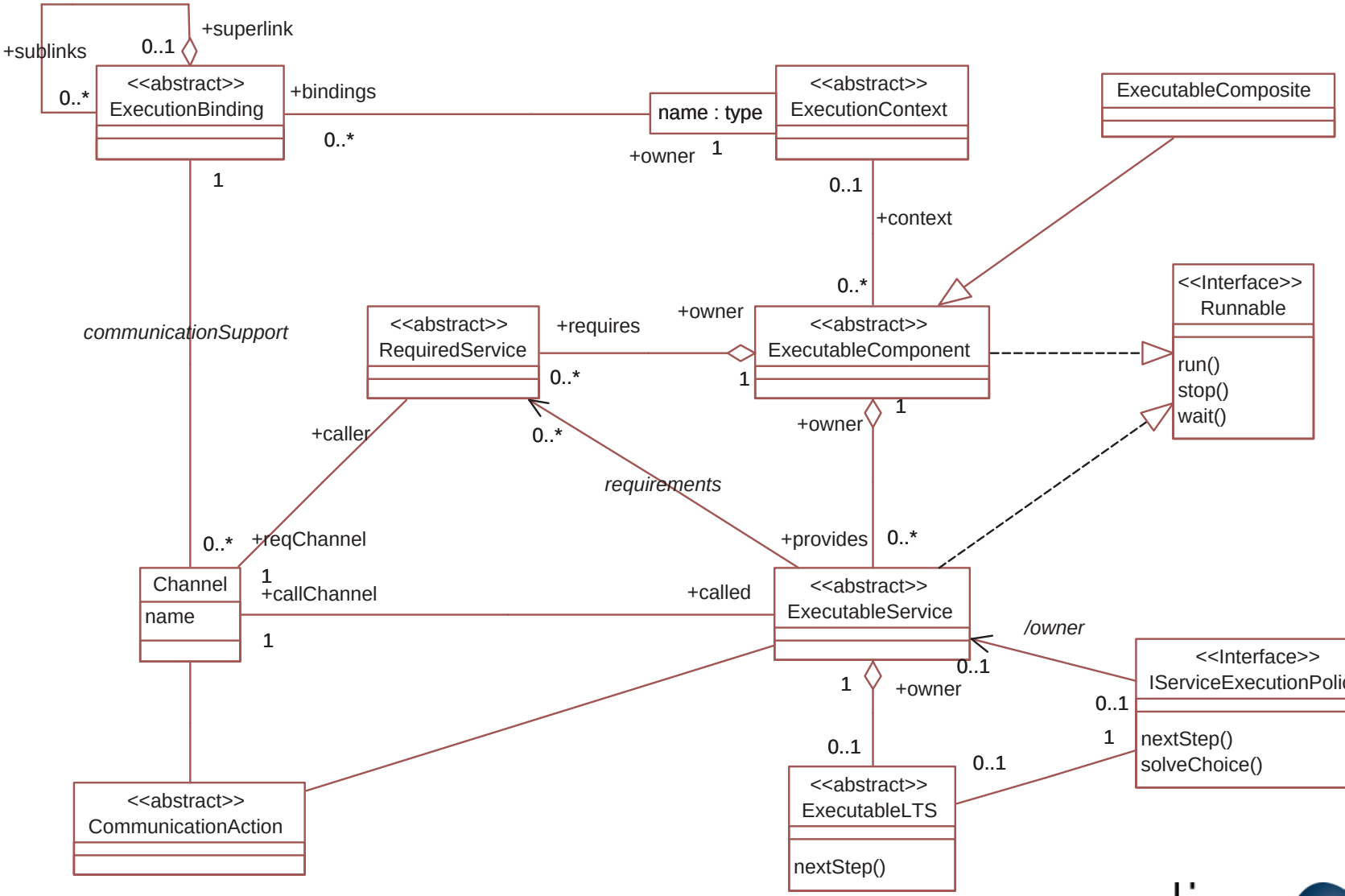
Kmelia/Java

sous CVS

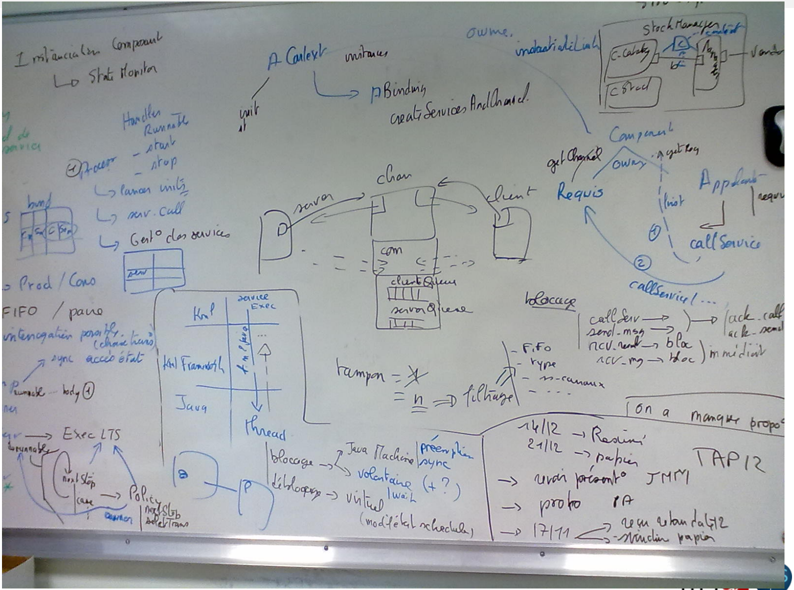
`DRAFTS/MetaModel/Executable/costo3.mdl`

le code a un peu varié depuis...

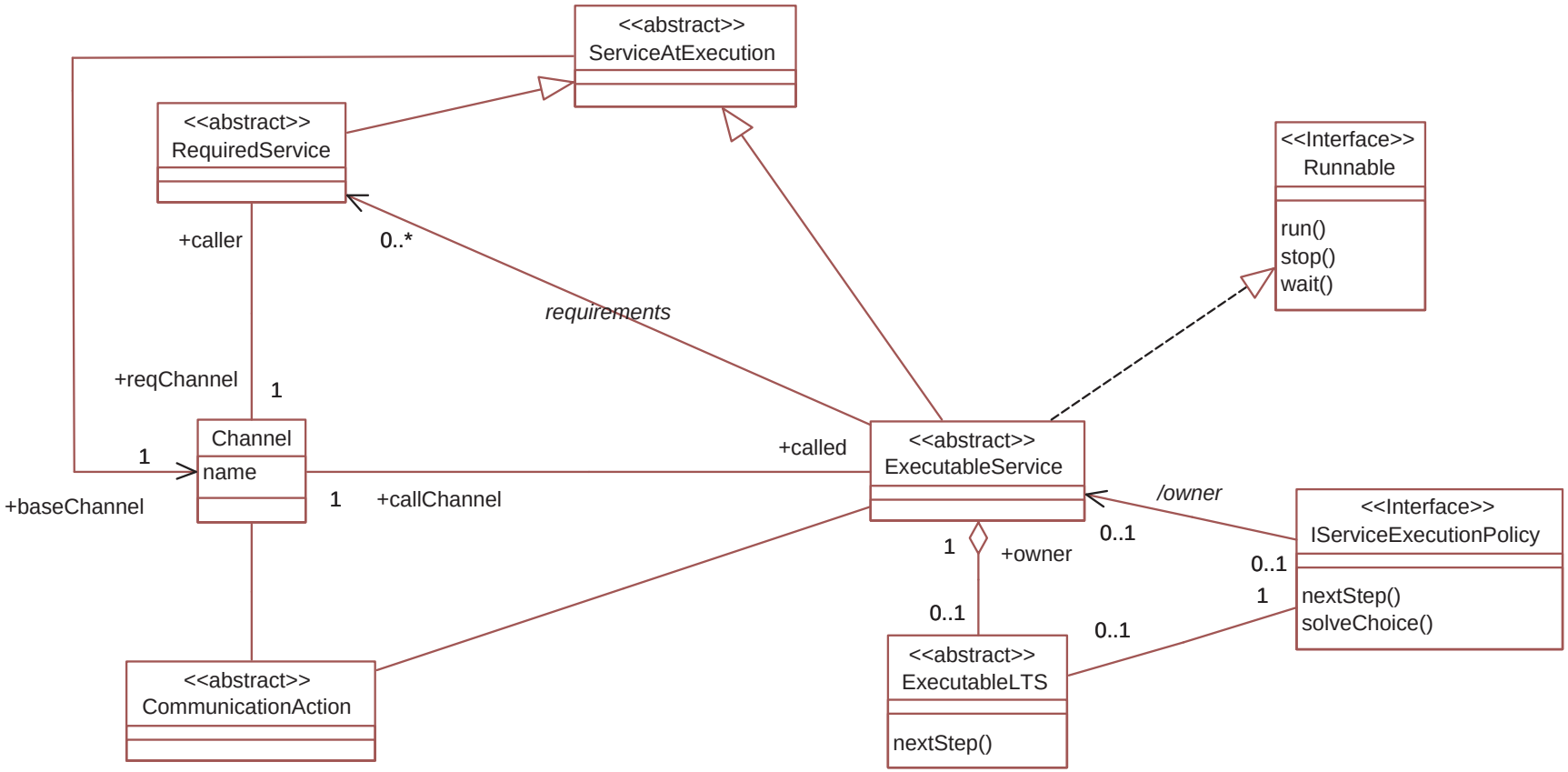
Kmelia/Java


+sublinks
0..1
+superlink
0..*
<<abstract>>
ExecutionBinding
+bindings
0..*
name : type
<<abstract>>
ExecutionContext
ExecutableComposite
+owner
1
0..1
+context
1
communicationSupport
<<abstract>>
RequiredService
+requires
0..*
+owner
<<abstract>>
ExecutableComponent
<<Interface>>
Runnable
run()
stop()
wait()
1
+owner
1
+caller
0..*
requirements
0..*
+reqChannel
+provides
0..*
Channel
name
1
+callChannel
1
+called
<<abstract>>
ExecutableService
/owner
0..1
1
+owner
<<Interface>>
IServiceExecutionPolicy
0..1
1
nextStep()
solveChoice()
0..1
0..1
<<abstract>>
CommunicationAction
<<abstract>>
ExecutableLTS
nextStep()

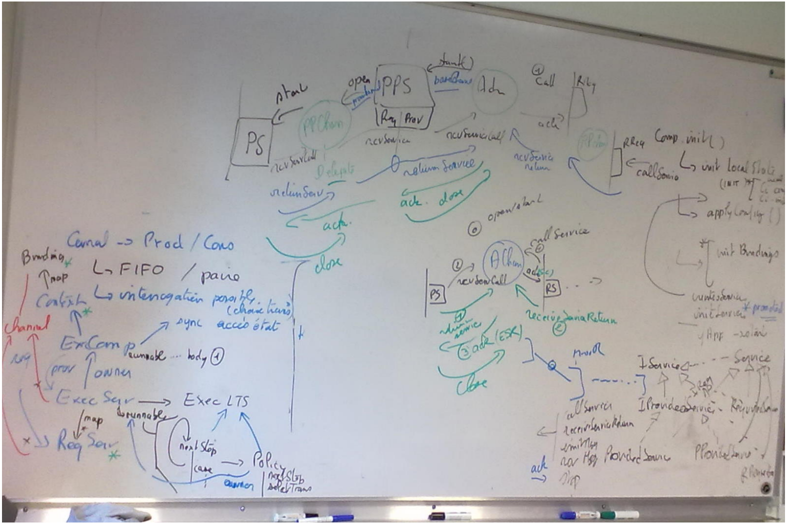
Tableau


Instanciation Composant
Stat Monitor
Binding
createServicesAndChannel
Component
Requis
callService
client
server
chan
com
clientQueue
serverQueue
blocage
FIFO
Prod / Cons
Java
Thread
Exec LTS
Policy
Java Machine
préemption
sync
volontaire
JMM
TAP12

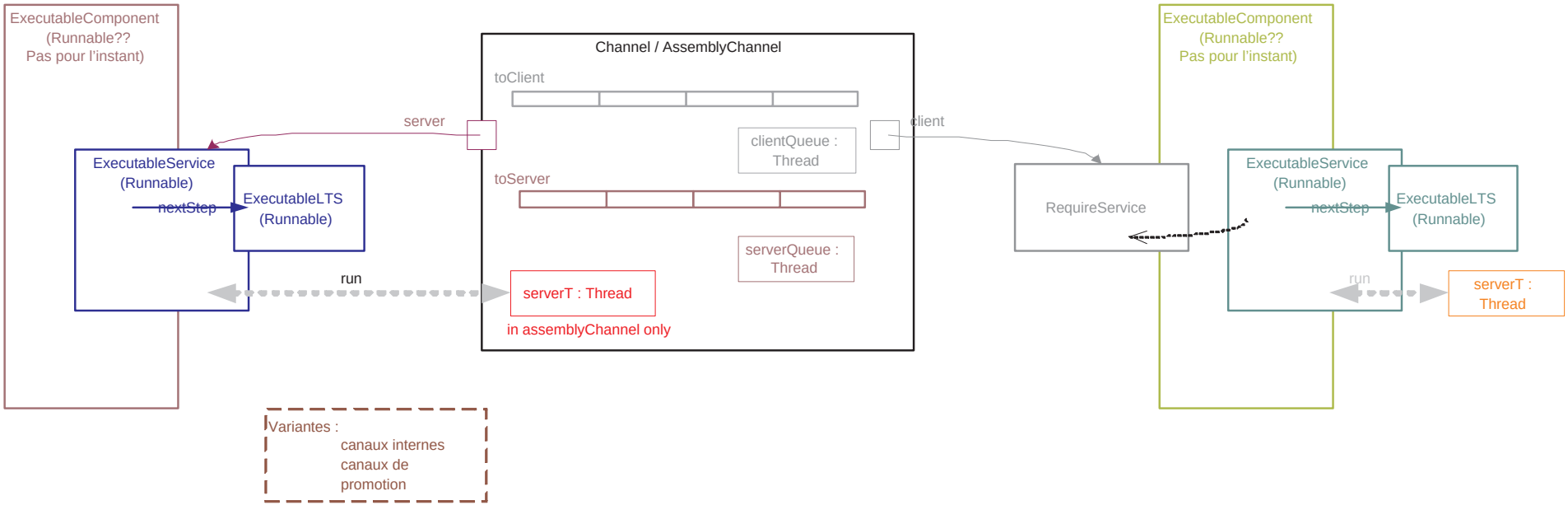
Kmelia/Java


<<abstract>>
ServiceAtExecution
<<abstract>>
RequiredService
<<Interface>>
Runnable
run()
stop()
wait()
+caller
0..*
requirements
+reqChannel
1
Channel
name
1
+baseChannel
1
+callChannel
+called
<<abstract>>
ExecutableService
/owner
<<Interface>>
IServiceExecutionPolicy
nextStep()
solveChoice()
0..1
1
+owner
0..1
0..1
1
0..1
<<abstract>>
CommunicationAction
<<abstract>>
ExecutableLTS
nextStep()

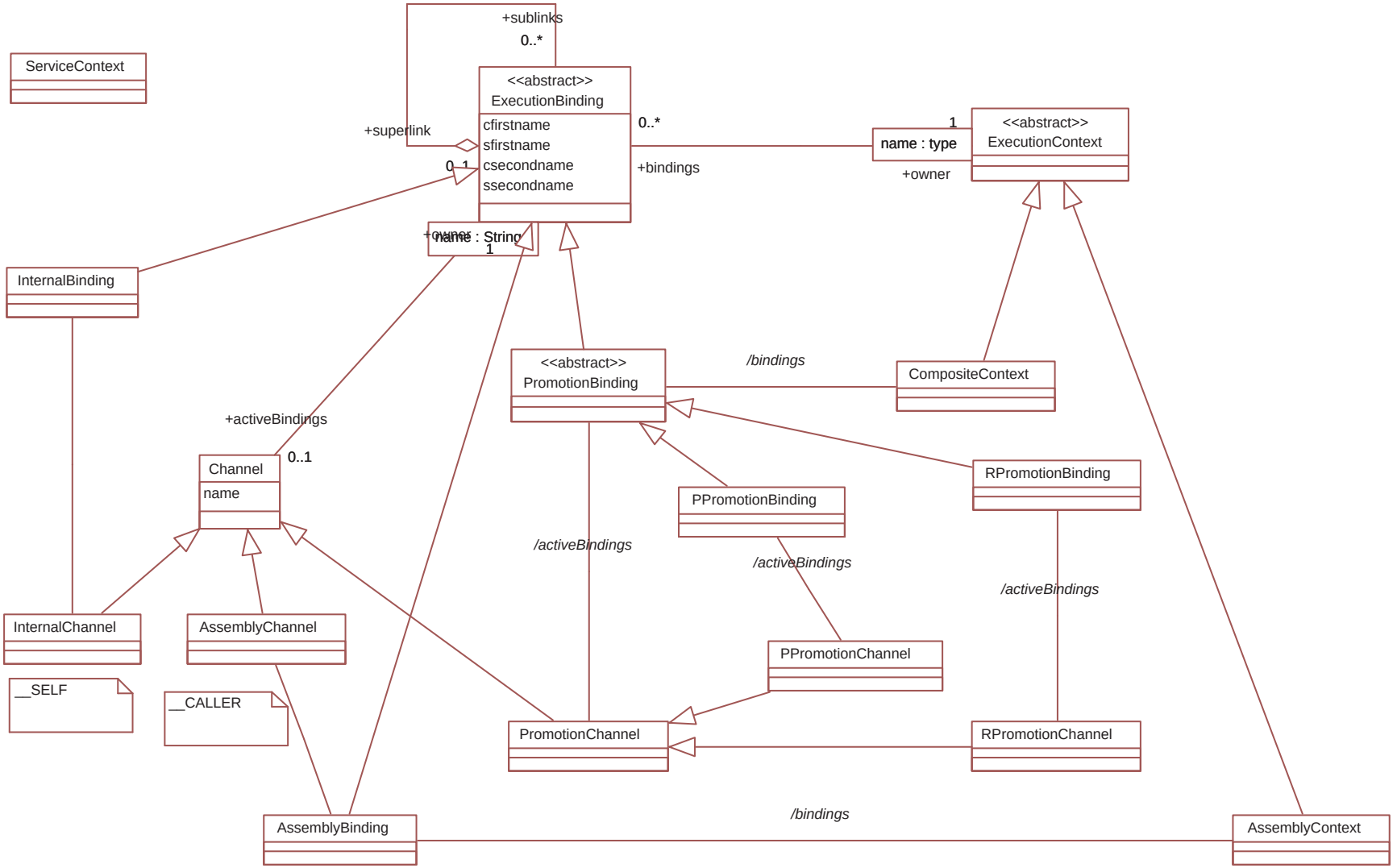
Tableau


PS
PPS
Req | Prov
Canal -> Prod / Cons
FIFO
ExecComp
Exec Serv -> Exec LTS
Policy
ack
close
Channel

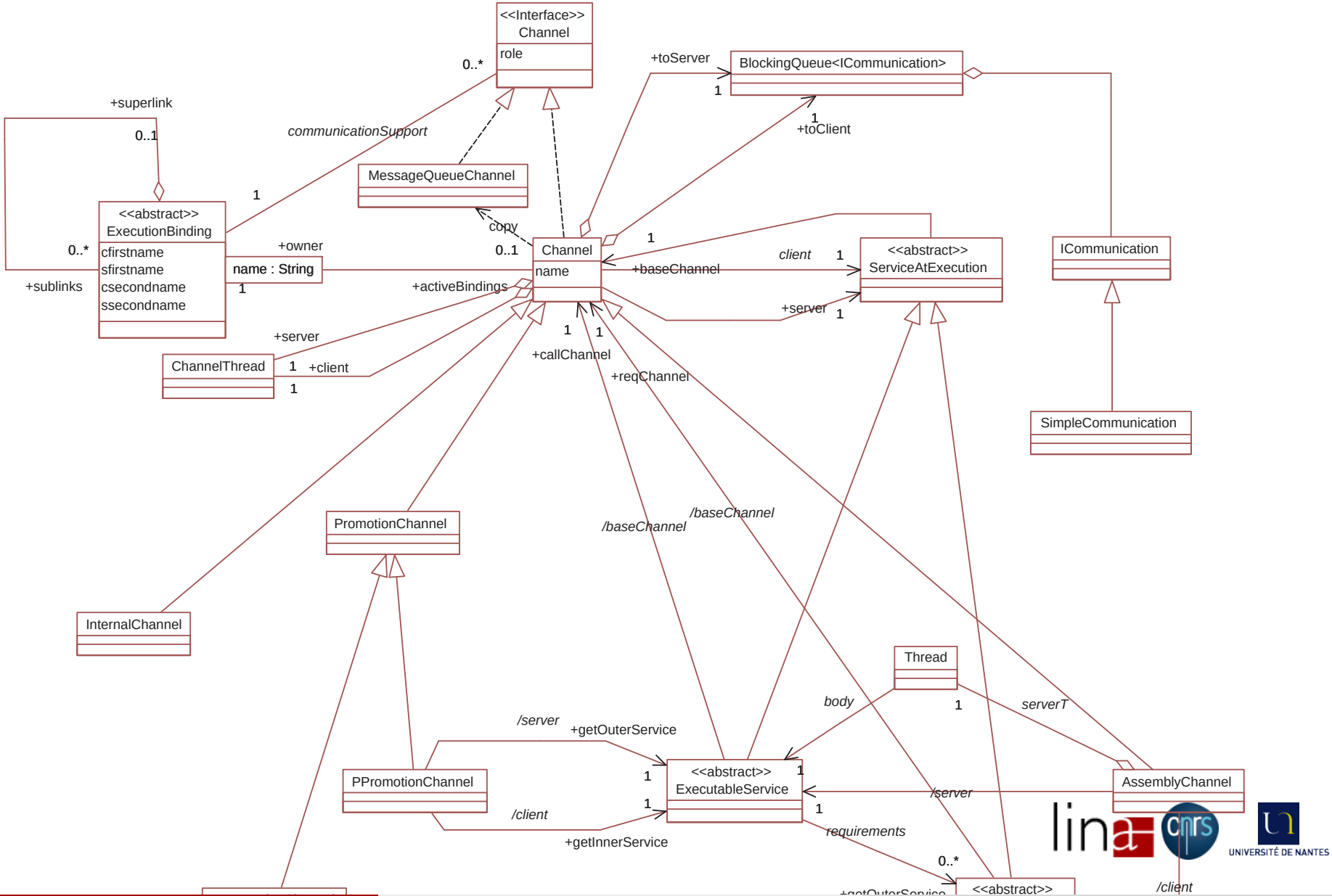
Framework


ExecutableComponent
(Runnable??
Pas pour l'instant)
ExecutableService
(Runnable)
nextStep
ExecutableLTS
(Runnable)
run
server
Channel / AssemblyChannel
toClient
clientQueue :
Thread
toServer
serverQueue :
Thread
serverT : Thread
in assemblyChannel only
client
RequireService
ExecutableComponent
(Runnable??
Pas pour l'instant)
ExecutableService
(Runnable)
nextStep
ExecutableLTS
(Runnable)
run
serverT :
Thread
Variantes :
canaux internes
canaux de
promotion

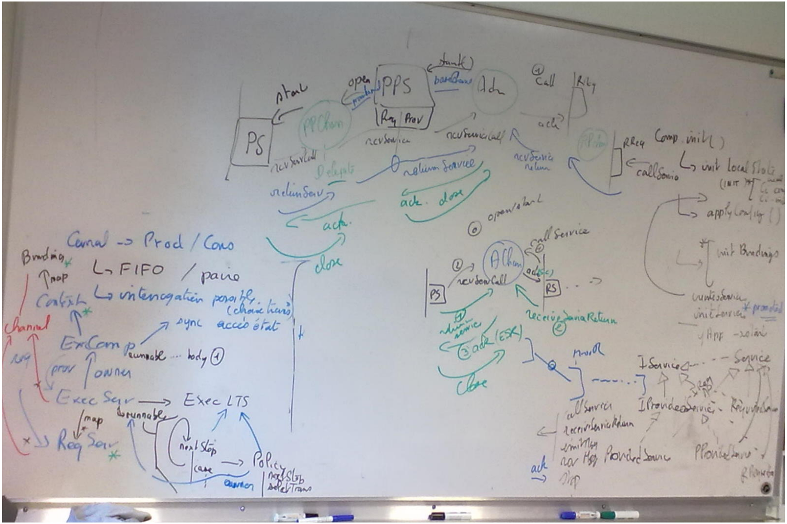
Contextes


ServiceContext
+sublinks
0..*
<<abstract>>
ExecutionBinding
cfirstname
sfirstname
csecondname
ssecondname
+superlink
0..1
0..*
+bindings
1
name : type
+owner
<<abstract>>
ExecutionContext
name : String
1
InternalBinding
<<abstract>>
PromotionBinding
/bindings
CompositeContext
+activeBindings
0..1
Channel
name
RPromotionBinding
PPromotionBinding
/activeBindings
/activeBindings
/activeBindings
InternalChannel
AssemblyChannel
PPromotionChannel
__SELF
__CALLER
PromotionChannel
RPromotionChannel
/bindings
AssemblyBinding
AssemblyContext

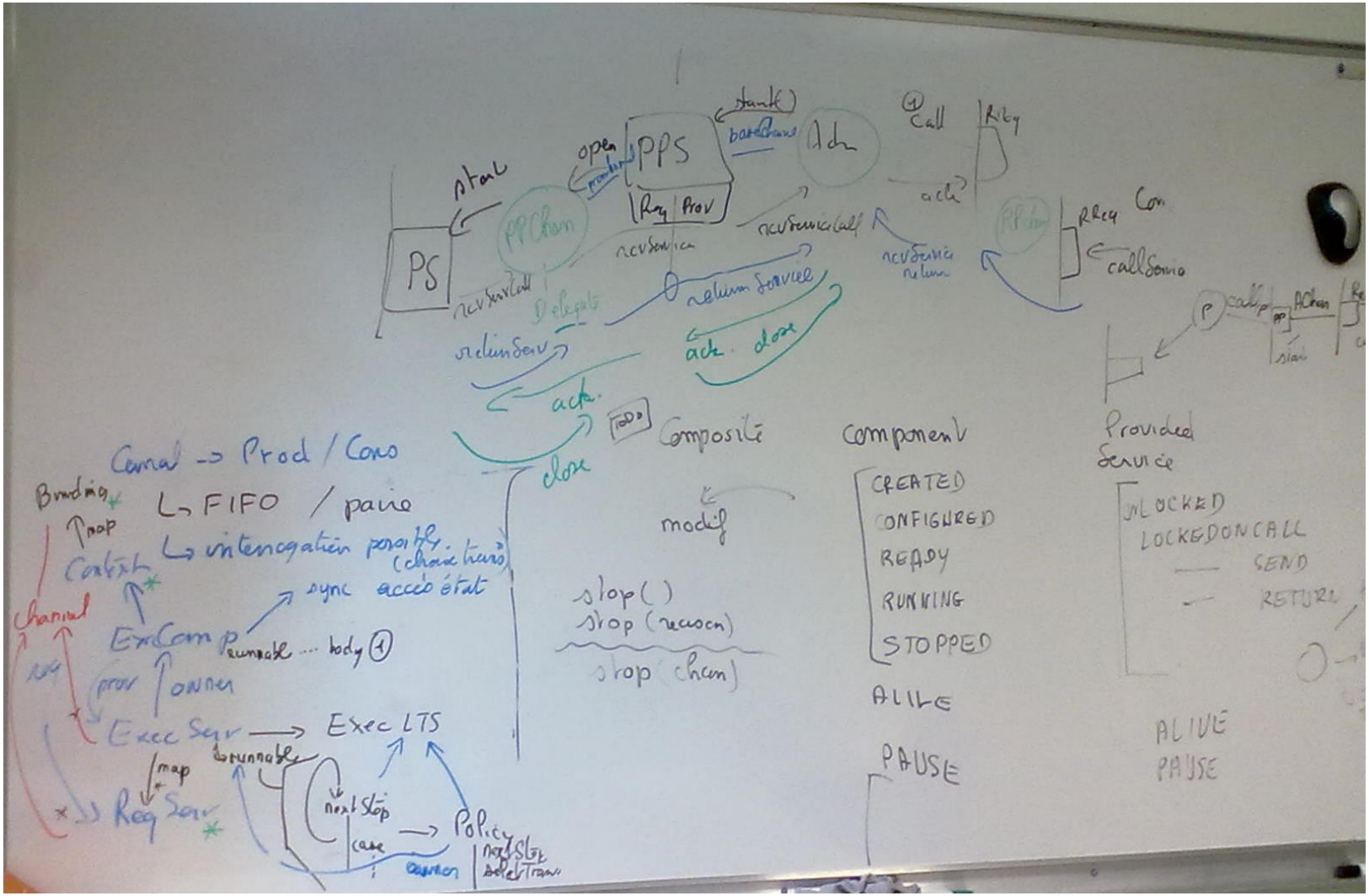
Canaux


<<Interface>>
Channel
role
0..*
communicationSupport
+superlink
0..1
<<abstract>>
ExecutionBinding
cfirstname
sfirstname
csecondname
ssecondname
0..*
+sublinks
1
+owner
name : String
1
MessageQueueChannel
copy
0..1
Channel
name
+activeBindings
+server
ChannelThread
1
+client
1
1
1
+callChannel
1
+reqChannel
+toServer
1
BlockingQueue<ICommunication>
1
+toClient
1
+baseChannel
client
1
<<abstract>>
ServiceAtExecution
+server
1
ICommunication
SimpleCommunication
PromotionChannel
/baseChannel
/baseChannel
InternalChannel
Thread
body
1
serverT
/server
+getOuterService
PPromotionChannel
1
1
<<abstract>>
ExecutableService
1
1
/server
AssemblyChannel
/client
+getInnerService
requirements
0..*
<<abstract>>
/client

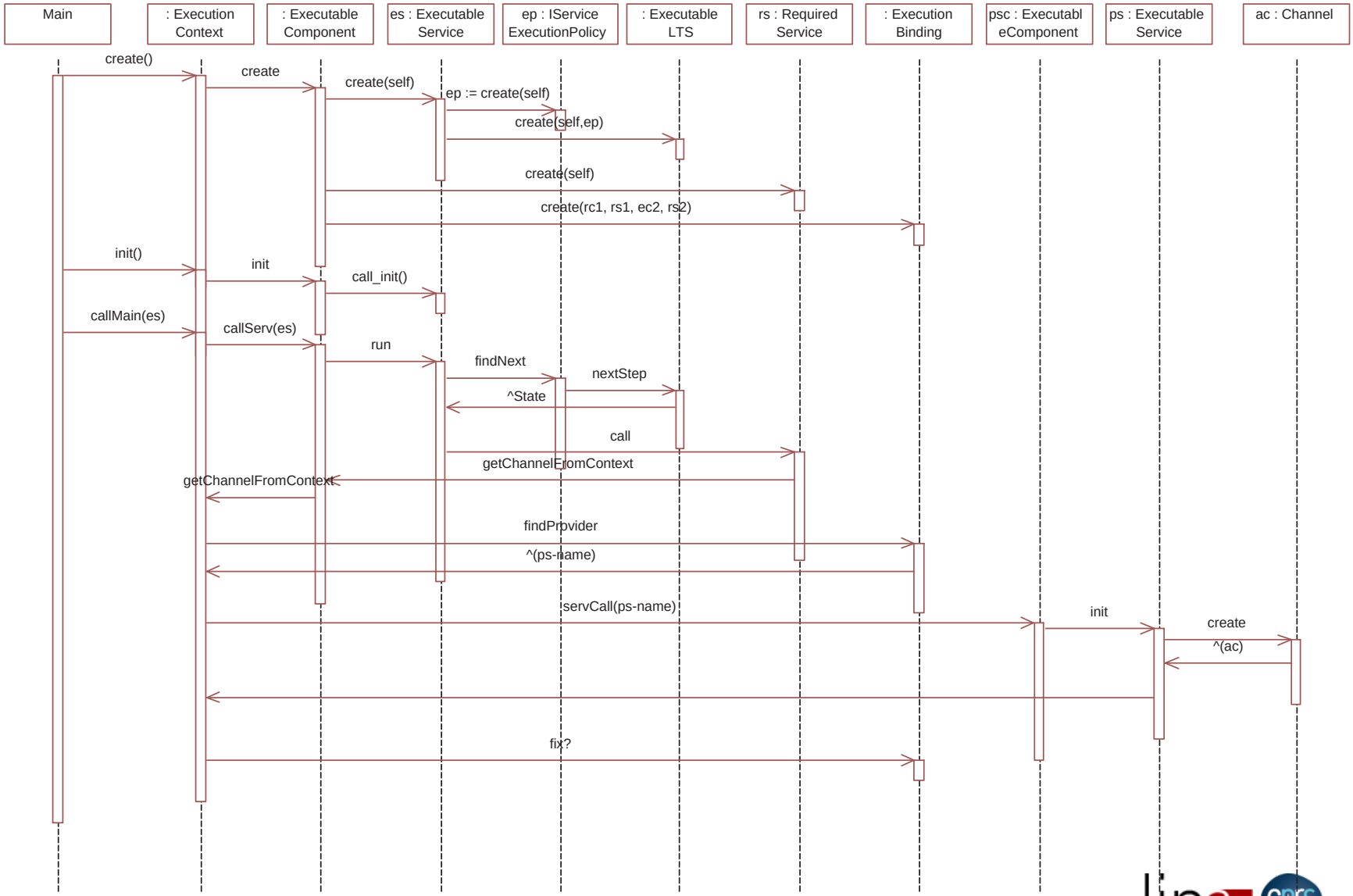
Tableau


PS
PPS
Canal -> Prod / Cons
FIFO
Exec Serv
Exec LTS
Req Serv
Policy
close
ack
Channel

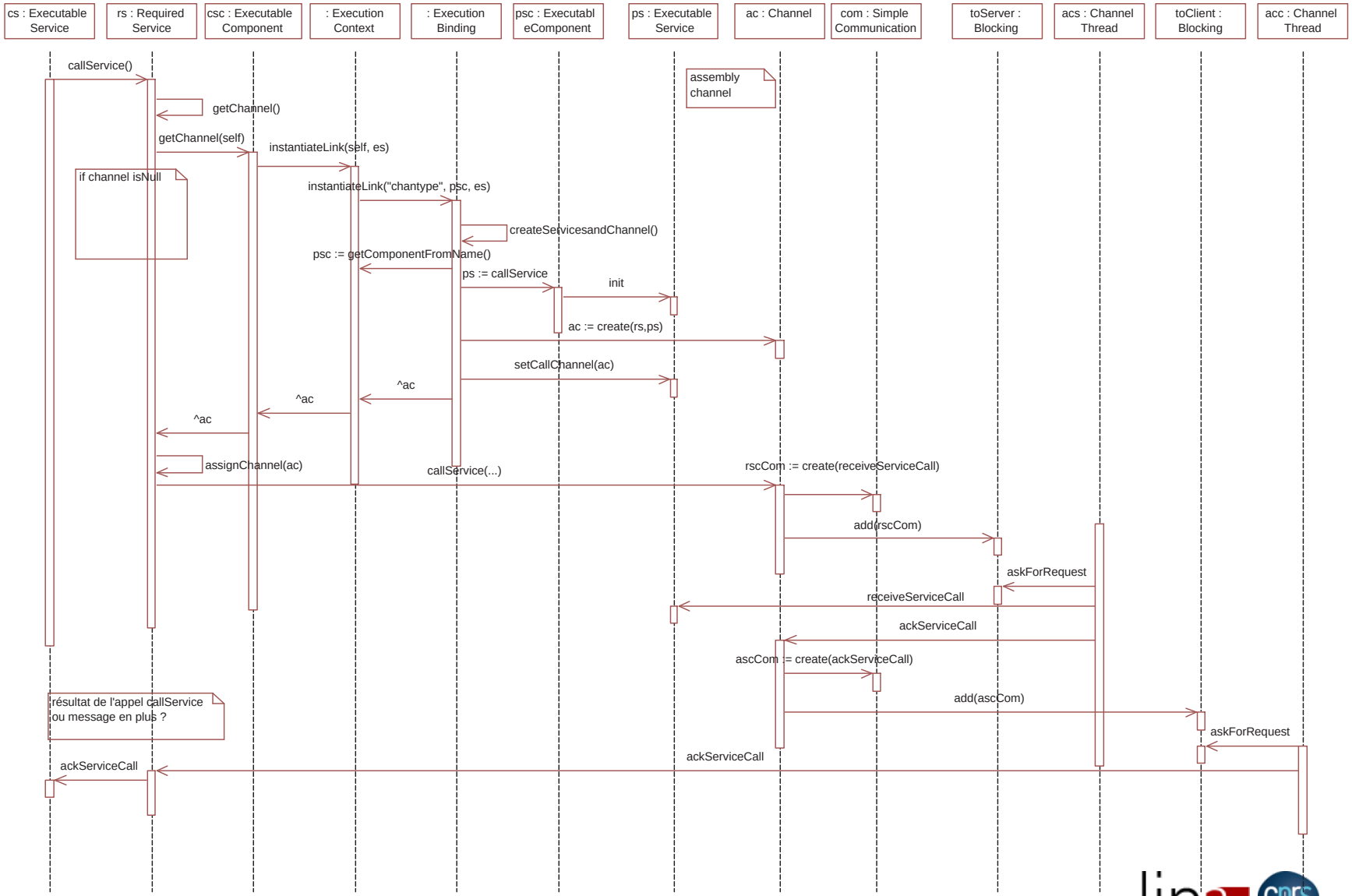
Tableau


PPS
PS
Call
Composite
Component
CREATED
CONFIGURED
READY
RUNNING
STOPPED
ALIVE
PAUSE
Provided Service
BLOCKED
LOCKEDONCALL
SEND
RETURN
Canal -> Prod / Cons
FIFO / pair
ExecComp
ExecSer
ExecLTS
ReqSer
Policy
modif
stop()
stop (recursion)
stop (chan)

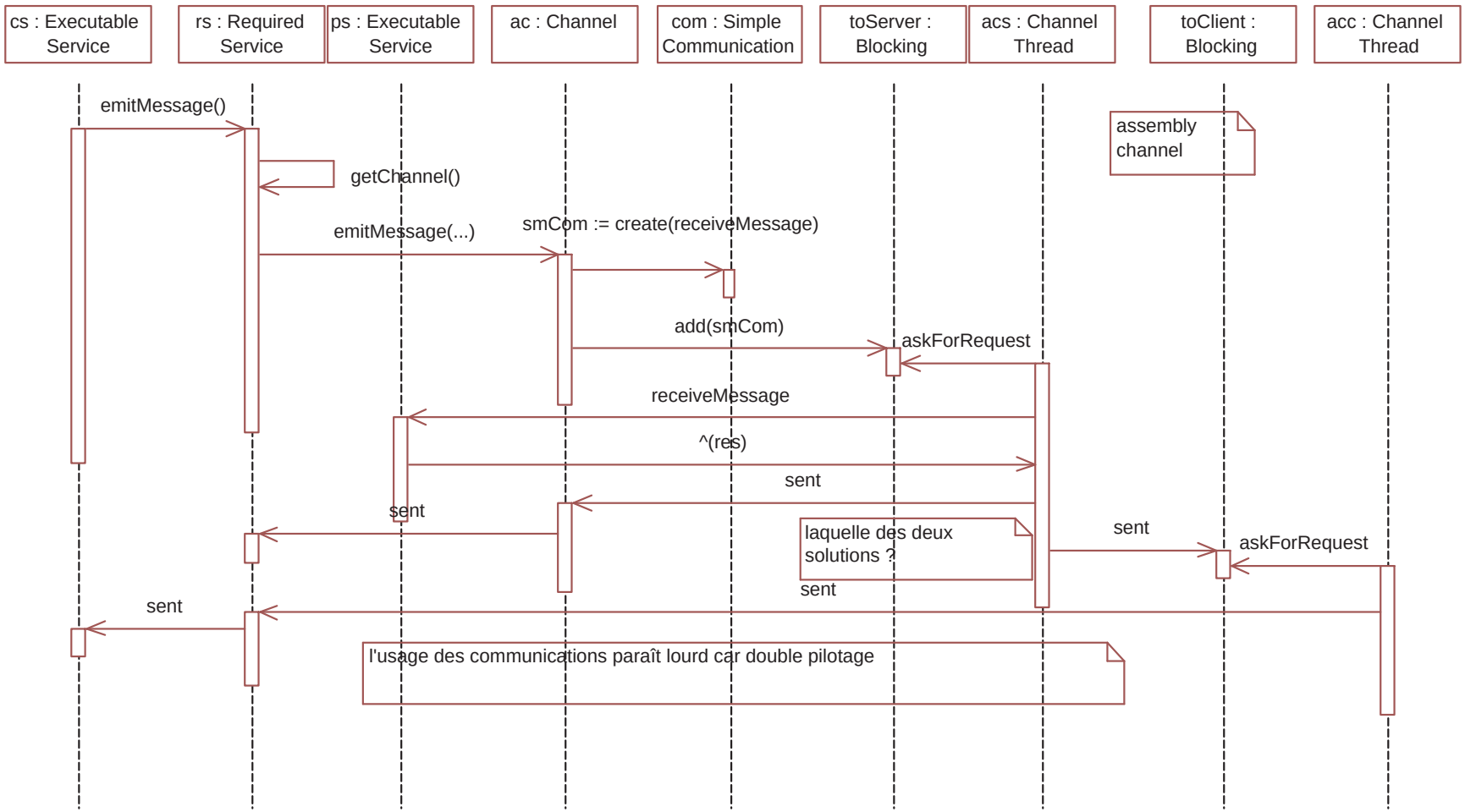
Instanciation



Service call



Message send



Plan

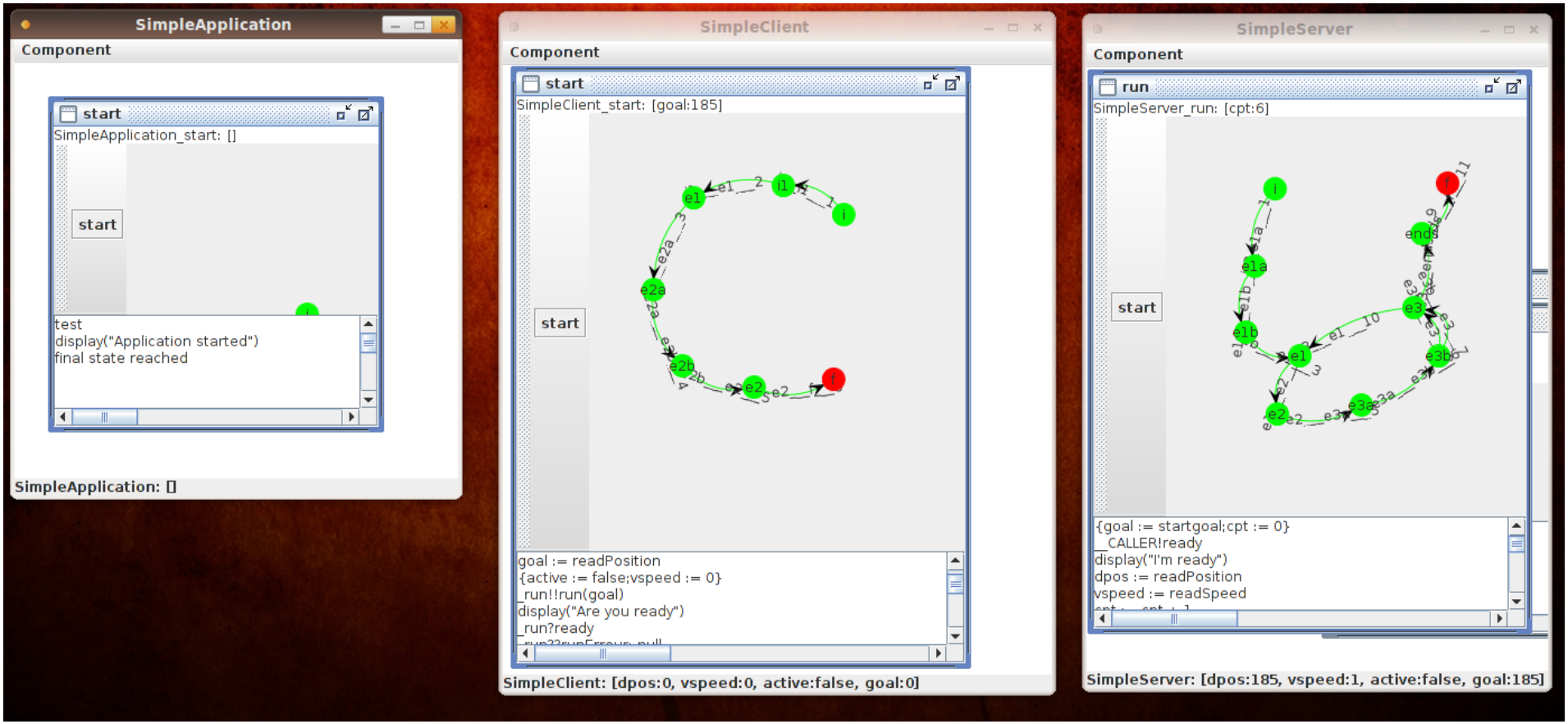
Plugin `kml2Java` - explications

GA

- templates
- visiteurs
- mappings
- application

Plan

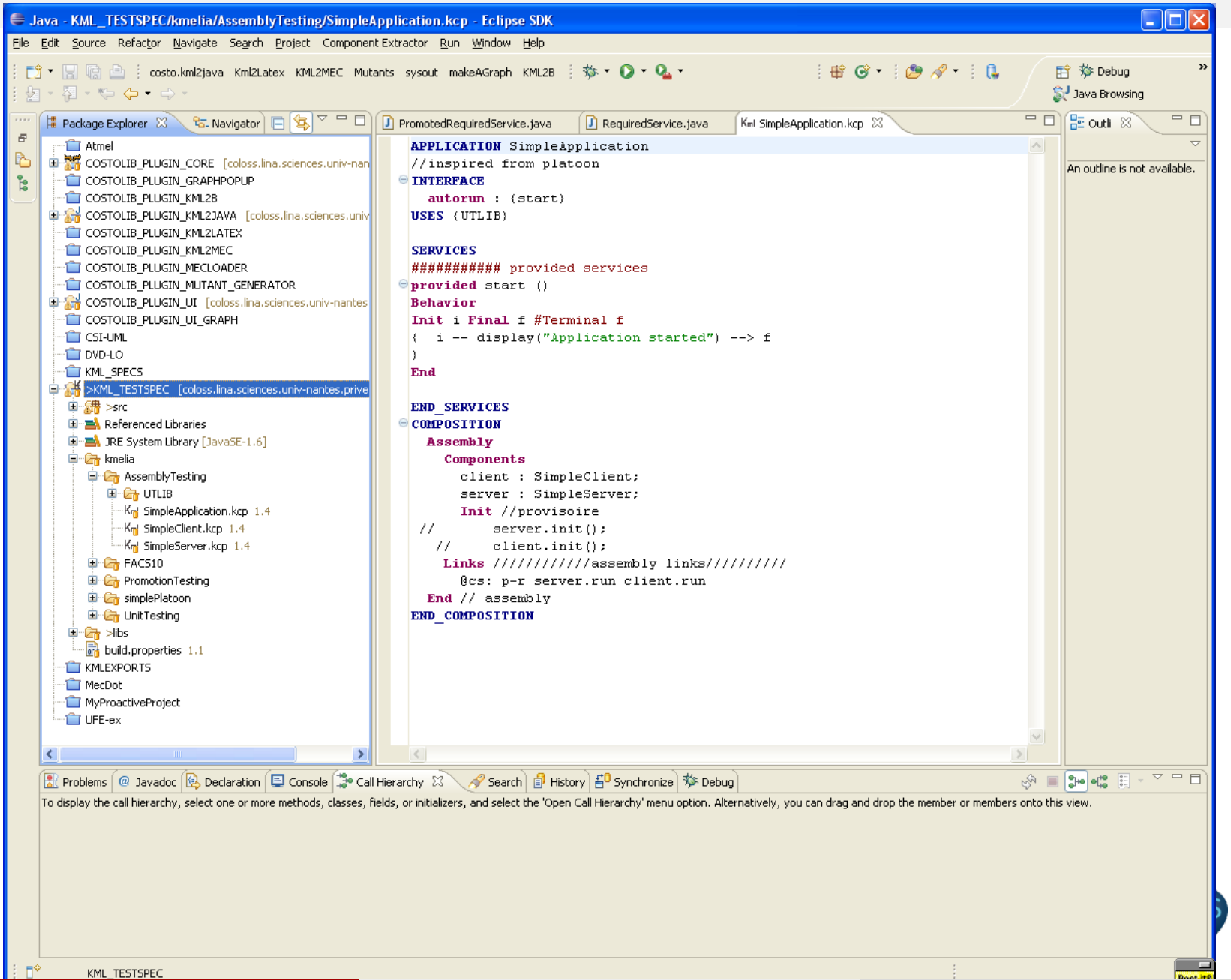
Plugin `costo_java_monitor` - Animation



démo de l'outil d'animation

Plan

Plugin kml2Java - démo



Plan

Vérification d'un modèle Kmelia

Nouvelle gamme de propriétés ...

- cohérence contrat/comportement dynamique/calculs des actions
- cohérence contrat/comportement dynamique
- typage code
- tests fonctionnels
- validation / prototype

... et de techniques

- à la JML (vérification des contrats)
- Tests
- Animation

traitements connexes

- répartition
- déploiement
- reconfiguration

Bilan partiel et Perspectives

- Prototype avec infrastructure limitée
  - composants et services (LTS)
  - canaux et support de synchro
  - ordonnanceurs / *threads*
  - assemblage / *entry point*
- **Support pour le test**
- Composition de services
  - canaux hiérarchiques, partagés
  - composition de *threads*

  $\Longrightarrow$ à creuser
- Problèmes des communications ()
  - JMS et files
  - *broadcast*

  $\Longrightarrow$ à creuser
- Traduire dans un autre modèle ?
- Vers un autre LTSA *CB Education perspective*