ETIA'09 – Lille – Juillet 2009

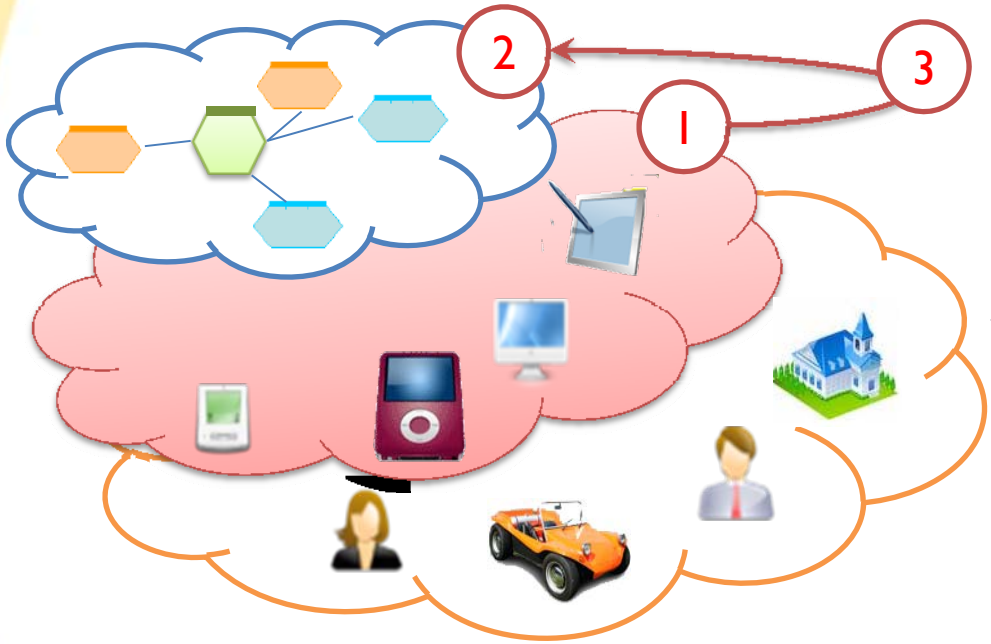
ATELIER WComp : Composition dynamique de Web Services pour Dispositifs dans WComp

Stéphane Lavirotte, Jean-Yves Tigli, Gaëtan Rey

Equipe Rainbow, Laboratoire I3S, UMR CNRS 6070,
Université de Nice Sophia Antipolis,

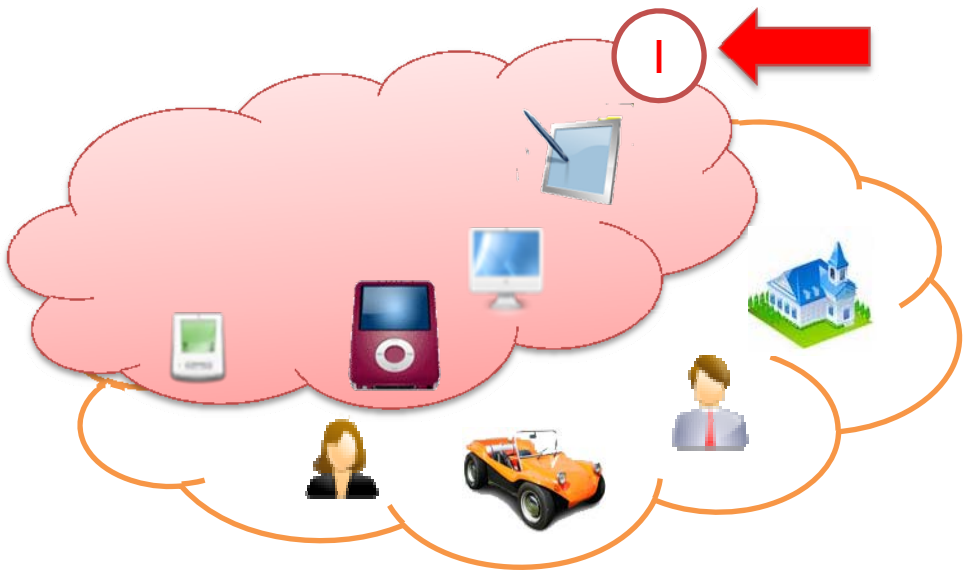
Email : <prenom.nom>@unice.fr
Web : http://rainbow.i3s.unice.fr/wcomp

Rappel sur WComp



1. Une infrastructure logicielle de services pour dispositifs dynamique (basée sur UPnP dans WComp.Net)
2. Deux niveaux de composition dynamique (LCA et SLCA)
3. Un mécanisme d'adaptation dynamique basé sur les AA

1. Infrastructure de Services pour Dispositifs



1. Une infrastructure logicielle de services pour dispositifs dynamique (basée sur UPnP dans WComp.Net)

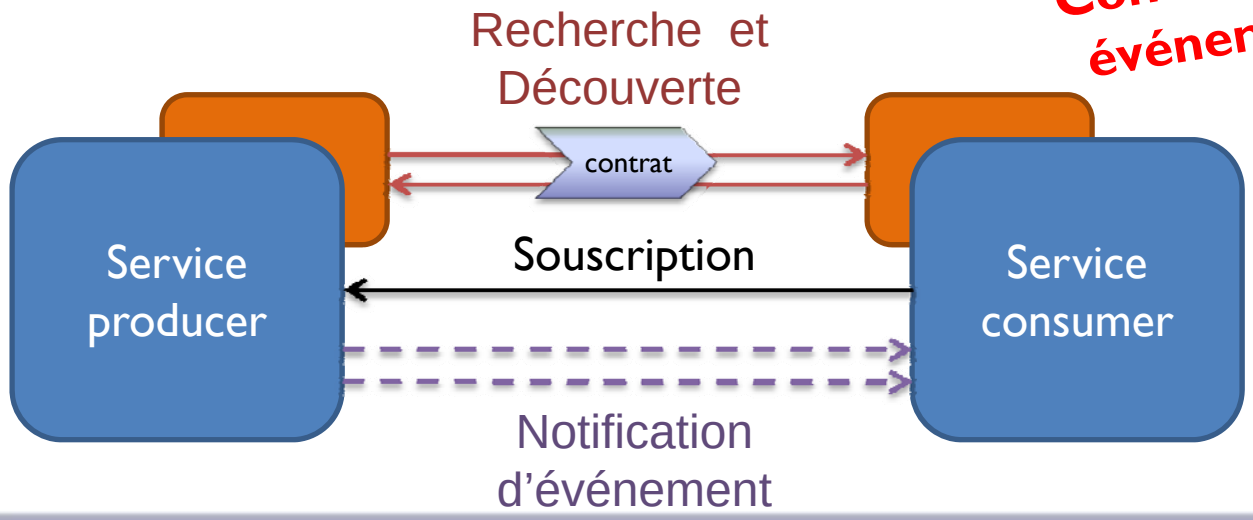
.

1. Services pour Dispositifs

- Trois évolutions majeures :
  - Publication distribuée et Découverte contextualisée
  - Gestion dynamique de l'apparition / disparition de services
  - Autre modèle d'interaction basé sur des événements

- Correspond en partie aux dernières évolutions des Web Services
  - Historiquement UPnP puis maintenant DPWS

Faible couplage
Interopérabilité
Découverte
Dynamique et contextuelle
Gestion Apparition / Disparition
Communications événementielles



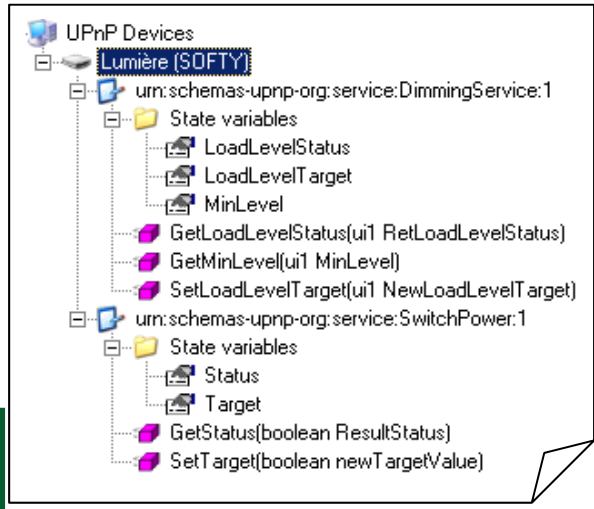
I. En fait … UPnP, est une première tentative de standard Web Service pour Dispositifs

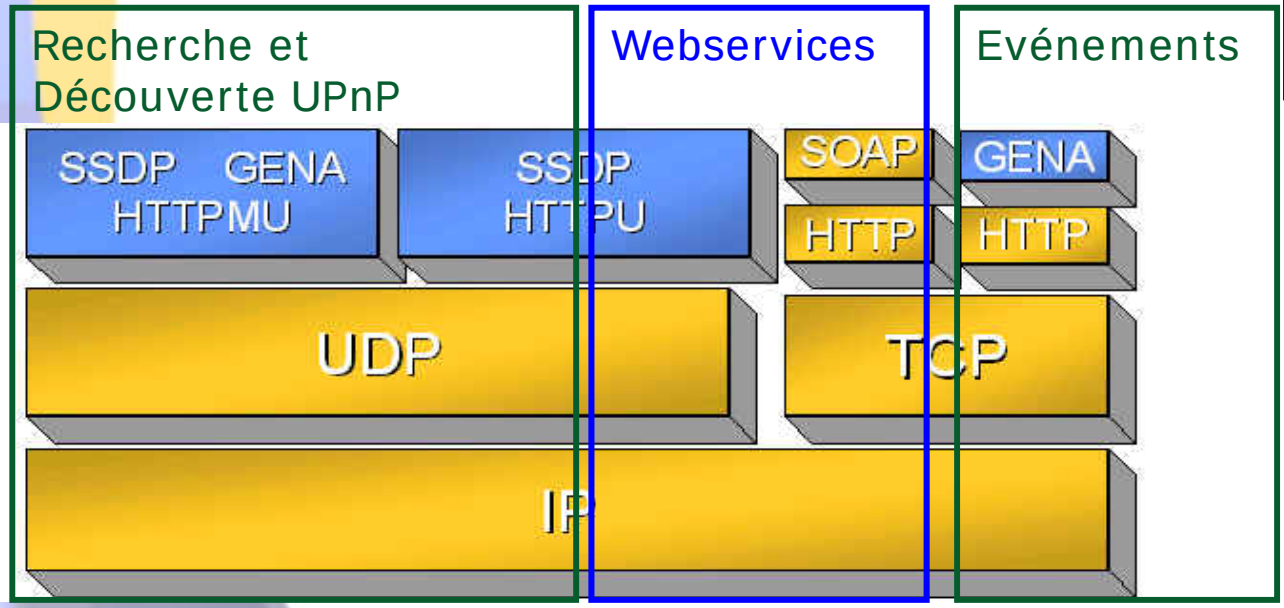
http://www.upnp.org/

Recherche et Découverte sur réseau local

Web Services classiques

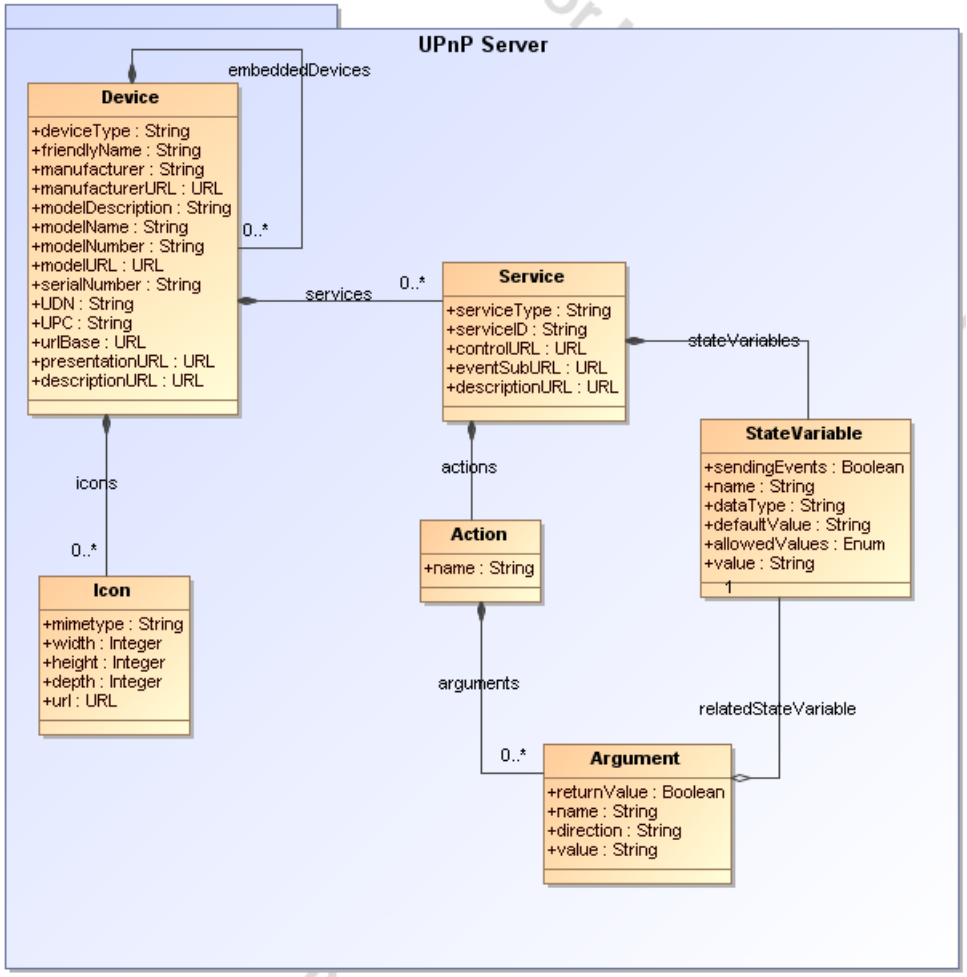
Interaction Evénementielle





1. Modèle d'UPnP

- Un *dispositif* offre un ensemble de services
- Un *service* offre des actions (méthodes) et des variables d'état (get/set)
- Les *variables d'état* peuvent être « evented », soit déclencher un *événement* sur changement de valeur



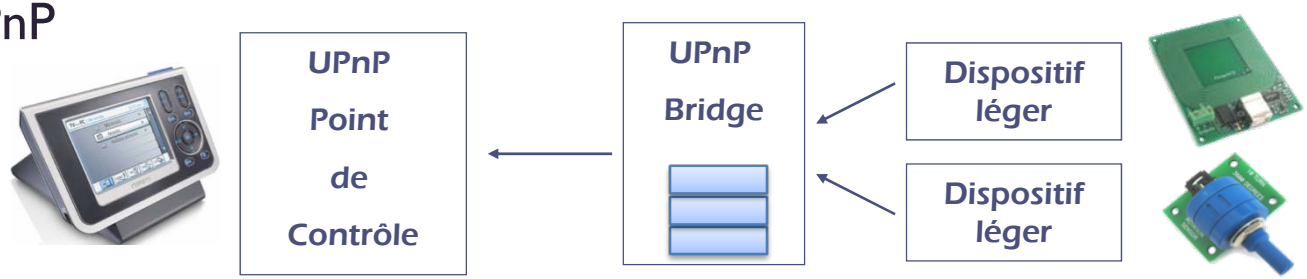
I. Des produits intégrant UPnP

- Logiciel

- Matériel

Sinon …

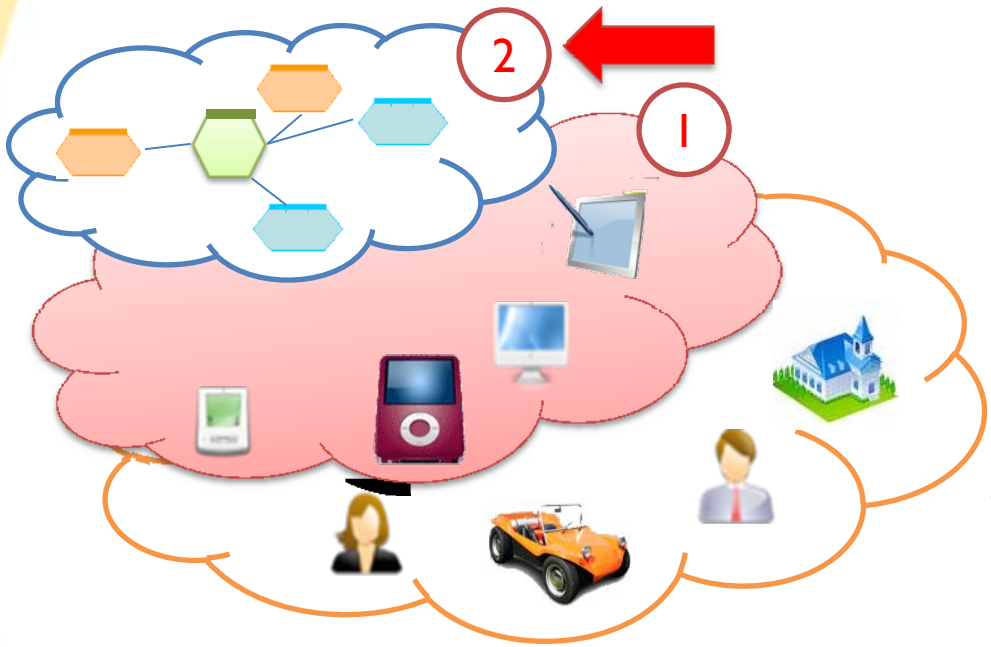
- L'approche retenue réside alors dans la conception de ponts (bridge) UPnP



I. Mise en œuvre d'UPnP pour WComp

- Lancer un Dispositif Virtuel (Light)
- Utilisation du Device Spy d'Intel
  - Dispositifs et leurs Services
  - Actions sur un service (méthodes) : invocation d'une méthode (ex. On())
  - Variables d'état d'une service : ex. status (lecture avec GetStatus())
  - Souscription à l'événement correspondant
  - Visualisation du contrat du protocole SSDP

2. Deux niveaux de composition dynamique (LCA et SLCA)



1. Une infrastructure logicielle de services pour dispositifs dynamique (basée sur UPnP dans WComp .Net)
2. Deux niveaux de composition dynamique (LCA et SLCA)

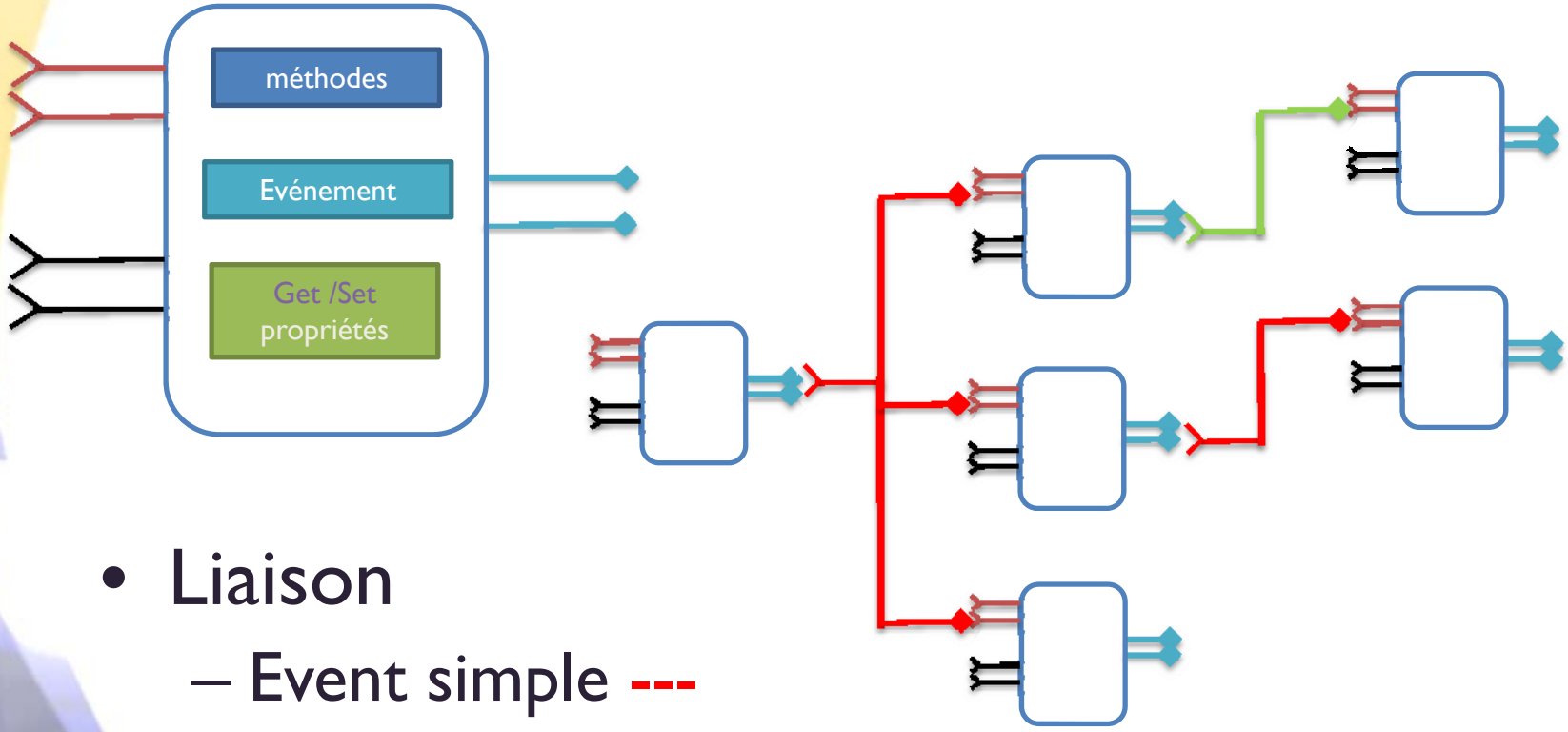
2. Modèle LCA

- LCA : Lightweight Components Architecture

1. L'application est un assemblage de composants légers
2. Composition par flots d'événements
3. Pas de distribution a priori (distribution obligatoirement explicite au travers des composants « proxy » vers des services pour dispositifs)

2. Des composants légers BeanWComp

- Composant BeanWComp



- Liaison
  - Event simple ---
  - Event Complexe ---

2. Exemple de BeanWComp .Net

- Template de Bean

```
10  using System;
11  using WComp.Beans;
12  using WComp.EventedBeans;
13
14  namespace WComp.Beans
15  {
16      [Bean(Category="Basic")]
17      public class Bean1 : EventedDrawable
18      {
```

Directive de compilation

2. Exemple de BeanWComp .Net

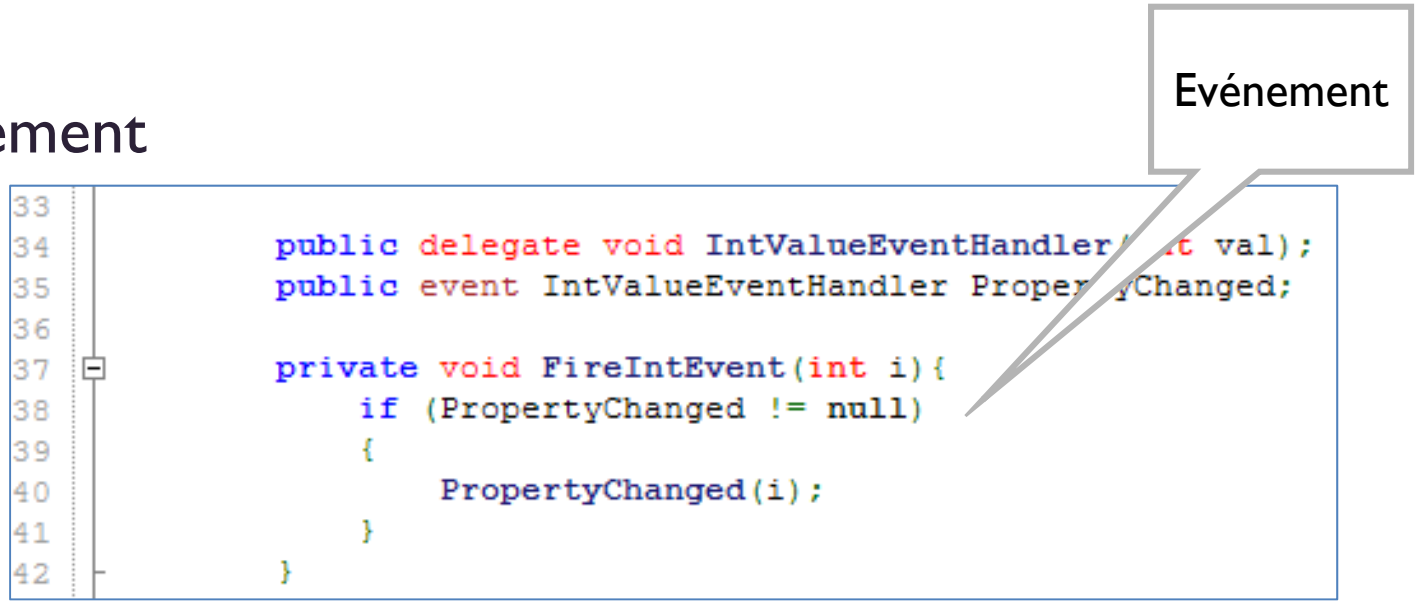
- Propriété

```
19          private int propVal;
20
21          public int MyProperty
22          {
23              get
24              {
25                  return propVal;
26              }
27              set
28              {
29                  propVal = value;
30
31              }
32          }
```

Propriété

2. Exemple de BeanWComp .Net

- Evénement



- Ex. Emission sur changement de valeur

```
27          set
28          {
29              propVal = value;
30              FireIntEvent(propVal);
31          }
```

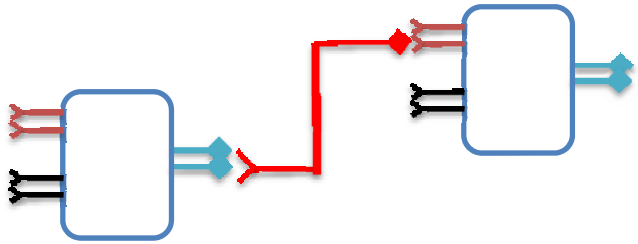
2. Exemple de BeanWComp .Net

- Méthodes publique et privée :

```
44
45          public void Method(bool val){
46              // TODO: Implement your public method
47          }
48
49          private void PrivateMethod(int val){
50              // TODO: Implement your private method
51          }
52
```

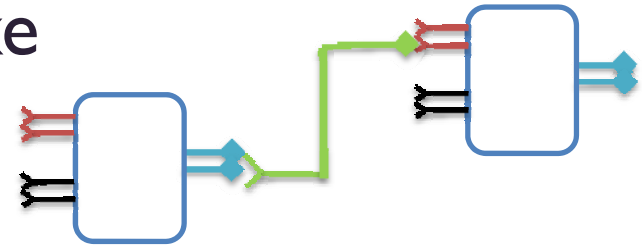
2. Modèles de connecteurs : Event et Event Complexe

- Connecteur Event Simple



  - C1.Event (param) => C2.Methode (param)

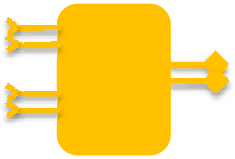
- Connecteur Event Complexe

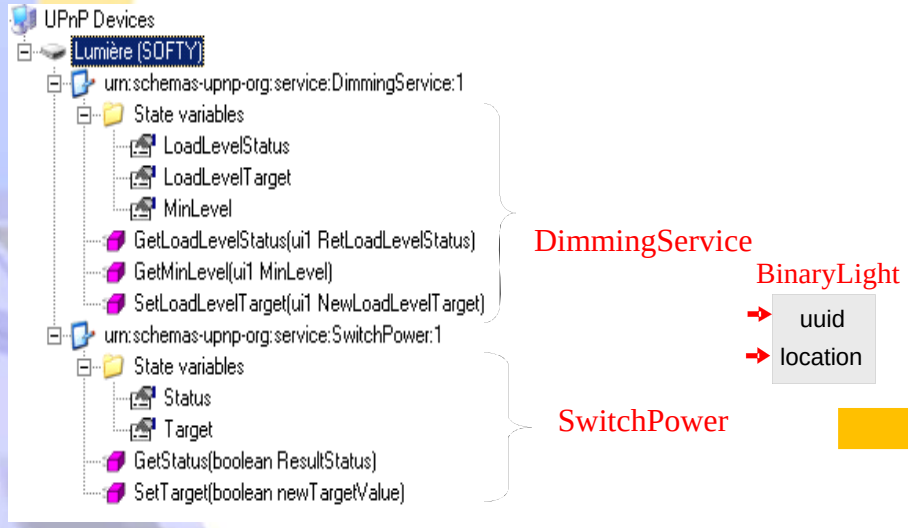


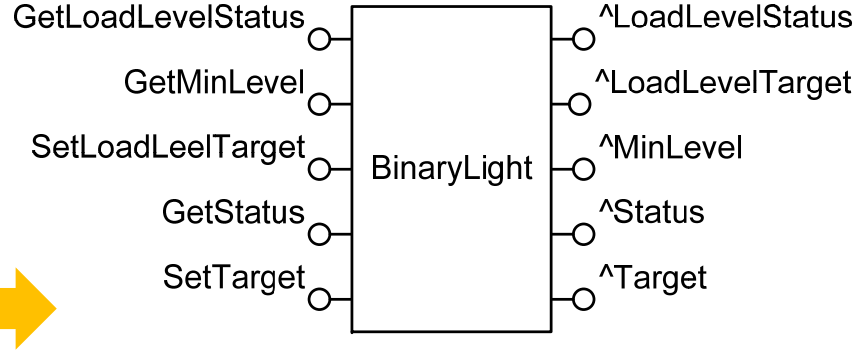
  - C1.Event (param) => C2.Methode (C1.GetProp())

2. Des Composants Proxy (pour UPnP et WebServices)

- Exemple Light UPnP





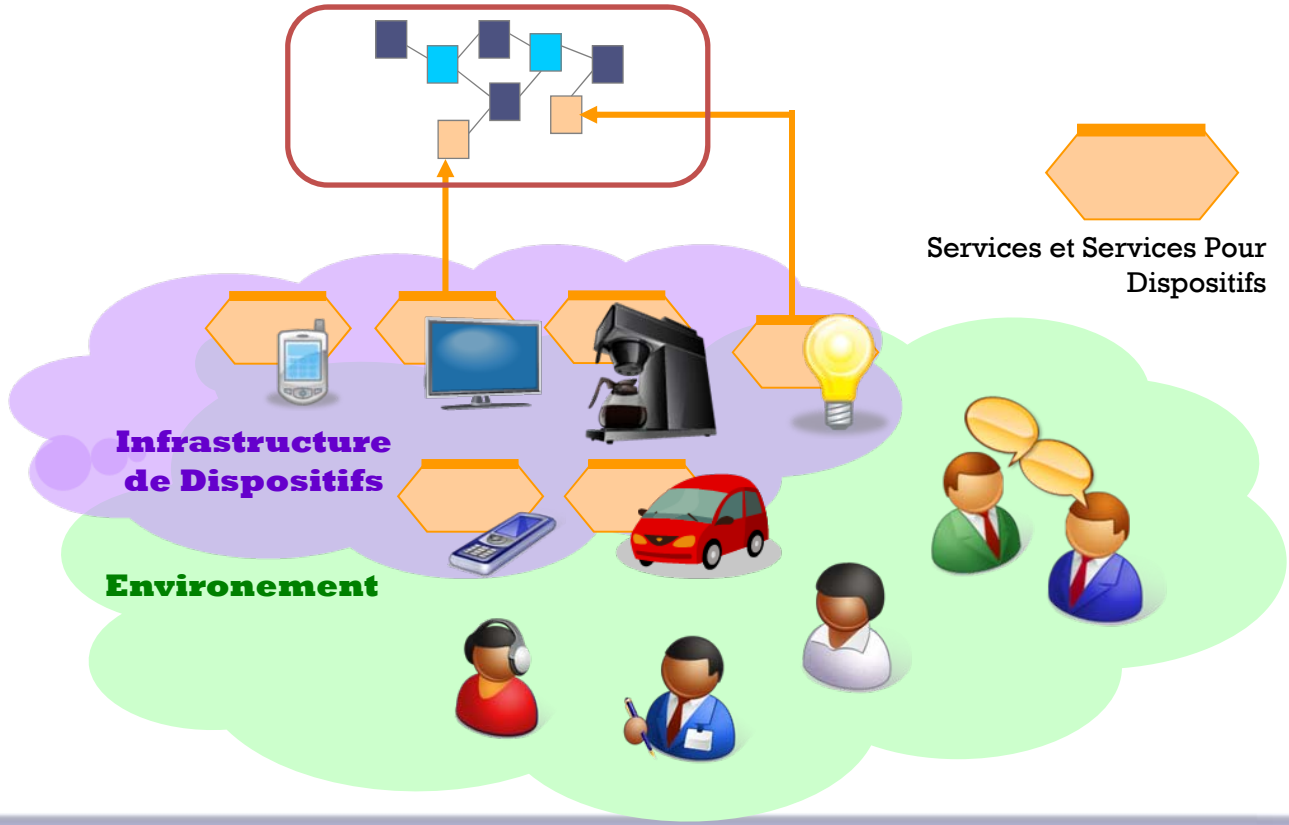


Service et Device UPnP

Composant Proxy

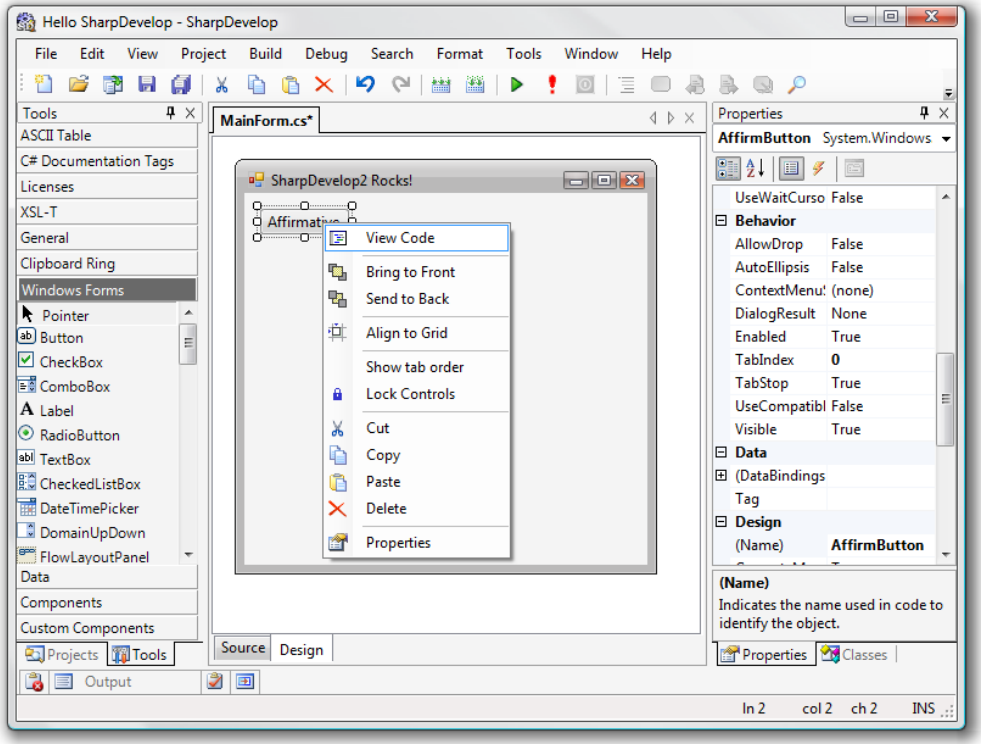
2. Mise en œuvre

- LCA pour la Composition Dynamique Locale de Services pour Dispositifs



Outils : Dans SharpDevelop (Addon)

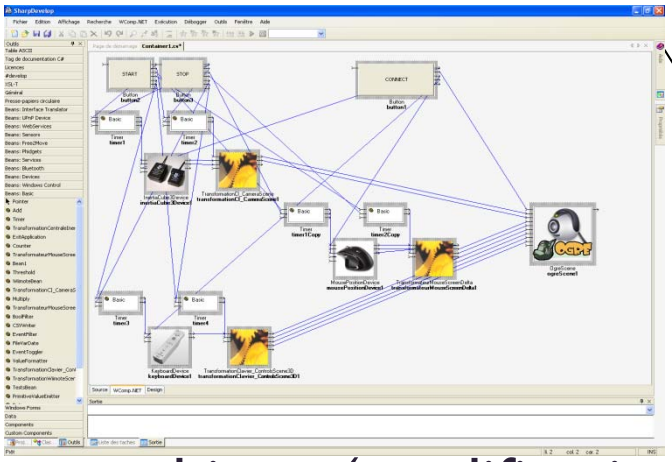
- SharpDevelop (SD) est un IDE basé sur la plate-forme .NET (2) qui offre un environnement de développement de qualité et libre comparable à VisualStudio
- Licence GPL



http://sharpdevelop.net/OpenSource/SD/

2. Les premiers Designers

- Designer visuel (modification graphique de l'assemblage de composant)
- Designer de code statique (projection)



Représentation graphique de l'assemblage

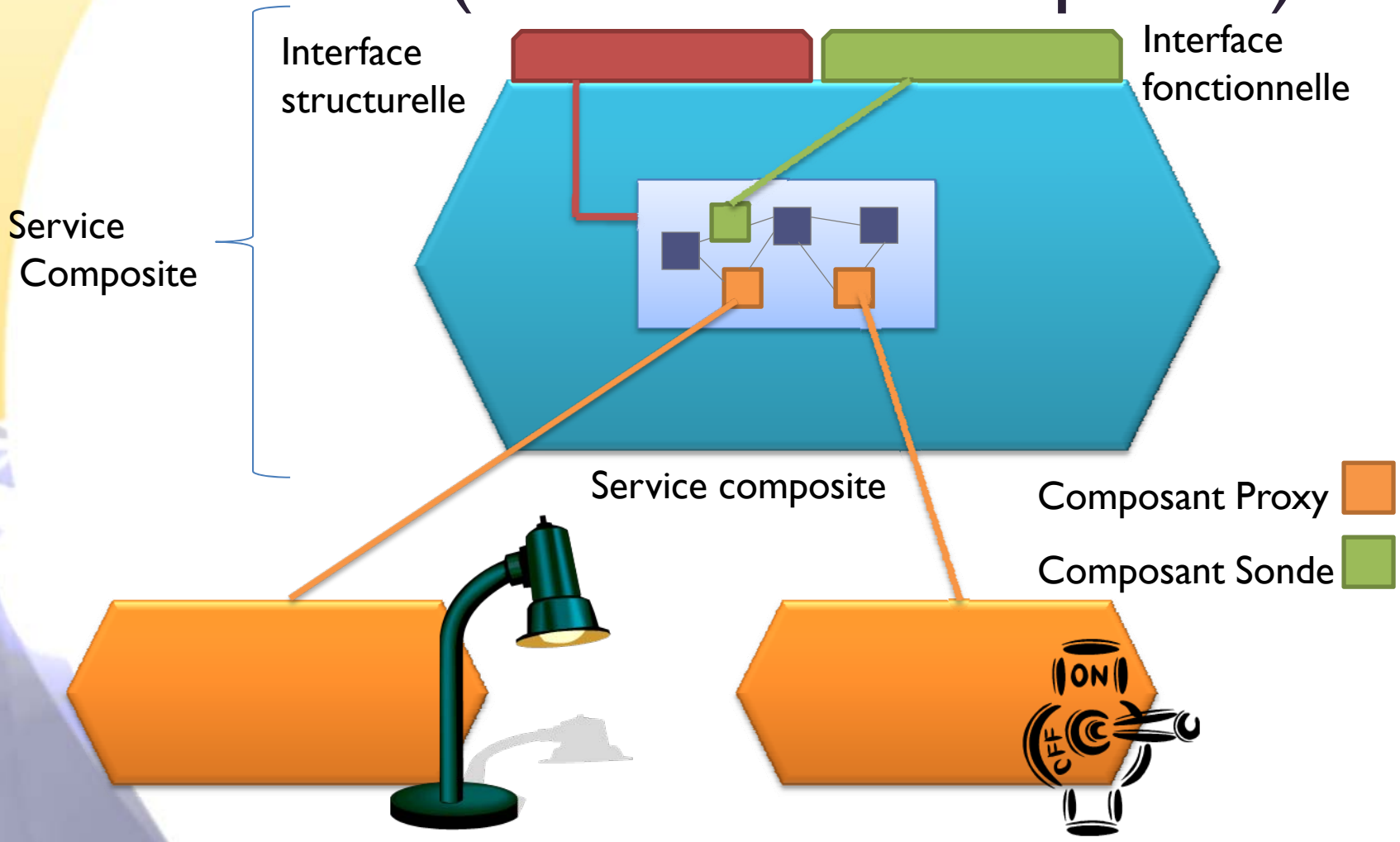
- Designer graphique (modification du rendu graphique)
- …

2. Mise en œuvre de LCA sous WComp

- Génération de composant proxy (ex. de la « light » UPnP)
- Création d'un assemblage avec événement simple et complexe pour piloter la « light »
- Création d'un composant BeanWComp
- Création d'un assemblage utilisant le composant créé

- Projection de l'application en code C# .Net généré

2. Modèle SLCA : Encapsulation d'un Container dans un Service UPnP (dit Service Composite)



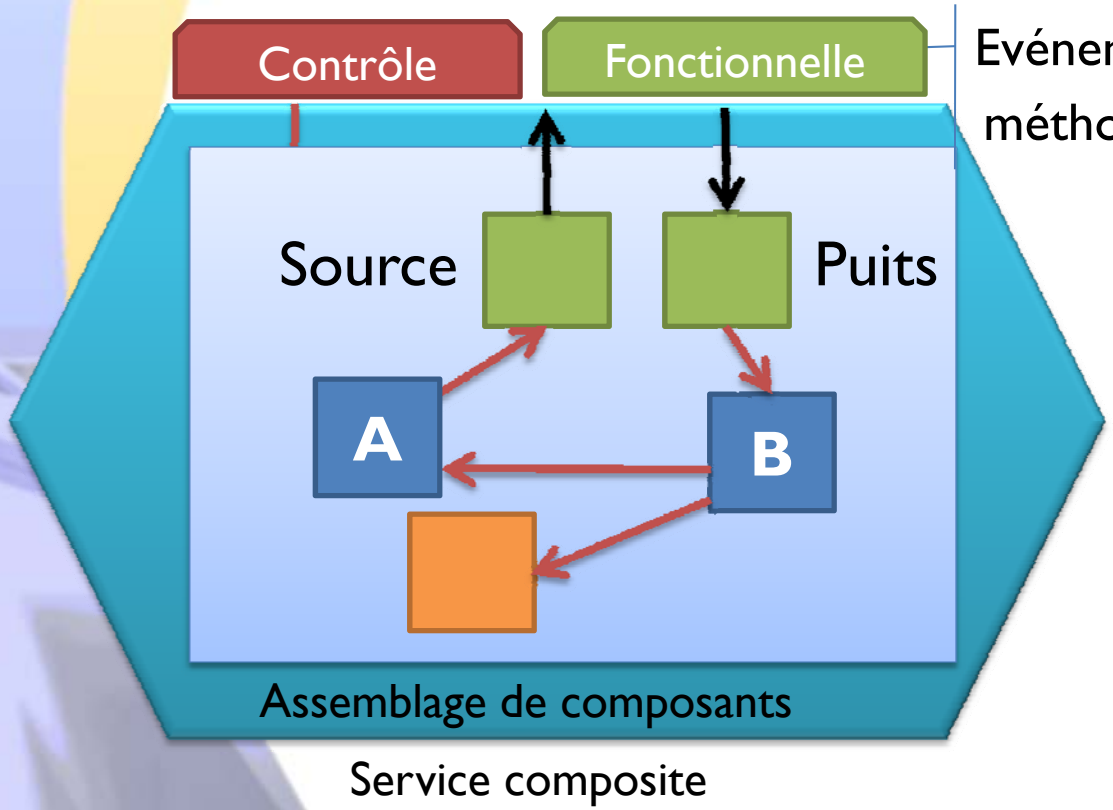
2. Interface de Contrôle du Services Composite

Contrôle

CheckBeanProperties	Donne la liste des noms et des types de propriétés pour un type de composants (prend actuellement un nom d'instance à la place)
CheckBeanPropertyValue	Donne la valeur et le type d'une propriété d'une instance, sérialisé au format XML
CheckBeans	Donne la liste des instances de l'assemblage
CheckBeanType	Donne le type de composant d'une instance
CheckBeanTypes	Donne la liste des types de composants chargés dans le container
CheckEvents	Donne la liste des événements de l'interface d'un type de composants (prend actuellement un nom d'instance à la place)
CheckLinkedBeansFrom	Donne la liste des noms des instances qui sont à la source d'une liaison en commun avec l'argument
CheckLinkedBeansTo	Donne la liste des noms des instances qui sont à la destination d'une liaison en commun avec l'argument
CheckLinks	Donne la liste des liaisons de l'assemblage
CheckLinksFrom	Donne les noms des liaisons qui partent du composant dont le nom est donné en argument
CheckLinksTo	Donne les noms des liaisons qui arrivent au composant dont le nom est donné en argument
CheckMethods	Donne la liste des méthodes de l'interface d'un type de composants (prend actuellement un nom d'instance à la place)
CreateBean	Crée une instance de composant
CreateLink	Crée une liaison entre deux composants
CreateNamedBean	Crée une instance de composant, et on peut en spécifier le nom
LoadType	Charge un type de composants dans le container
RemoveBean	Supprime une instance de composants de l'assemblage
RemoveLink	Supprime une liaison de l'assemblage
SetBeanPropertyValue	Modifie la valeur d'une propriété d'une instance, sérialisé au format XML
UnloadType	Décharge un type de composants dans le container (if ever...)

2. Interface fonctionnelle du Service Composite

- Définition de nouveaux composants Sonde



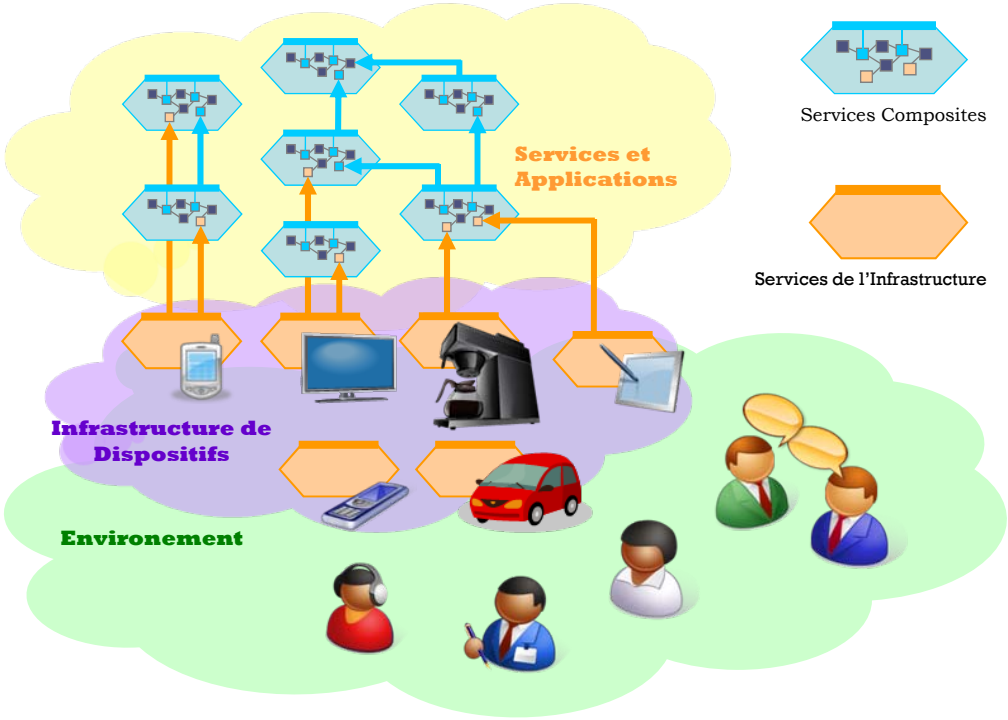
Source:
Appel local → Diffusion d'événement à partir de l'interface fonctionnelle

Puits:
Appel service → Diffusion d'événement local

2. Mise en œuvre

Compositions Dynamique Distribuées de Services pour Dispositifs

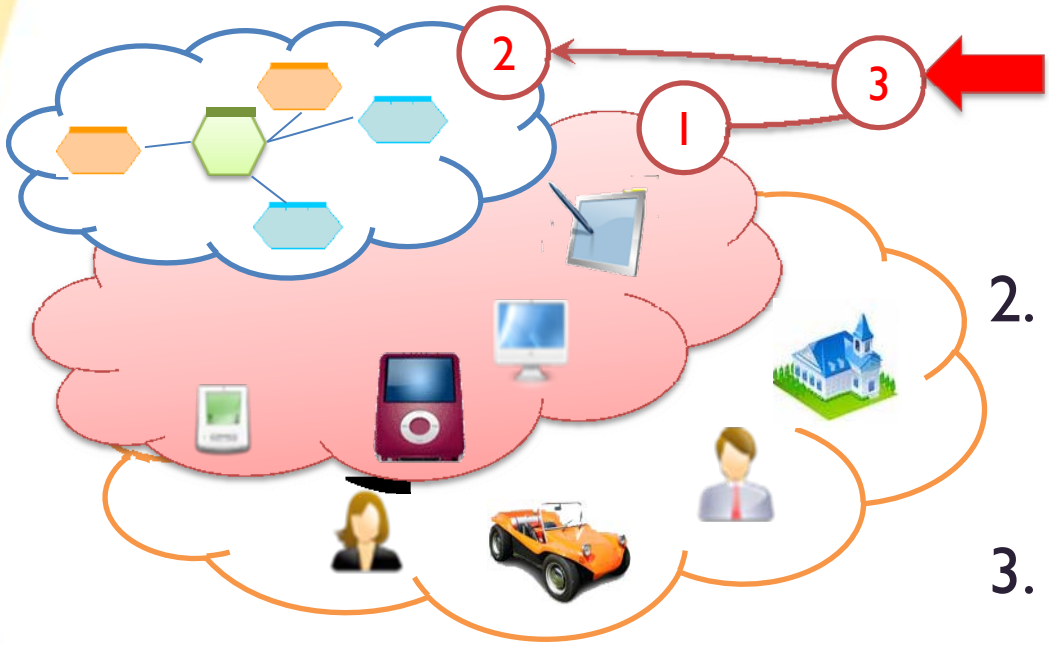
- Graphe de Services de l'infrastructure et de Services Composites
- Les Designers peuvent donc être des Service connecté à l'interface de Contrôle



2. Mise en œuvre de SLCA sous WComp

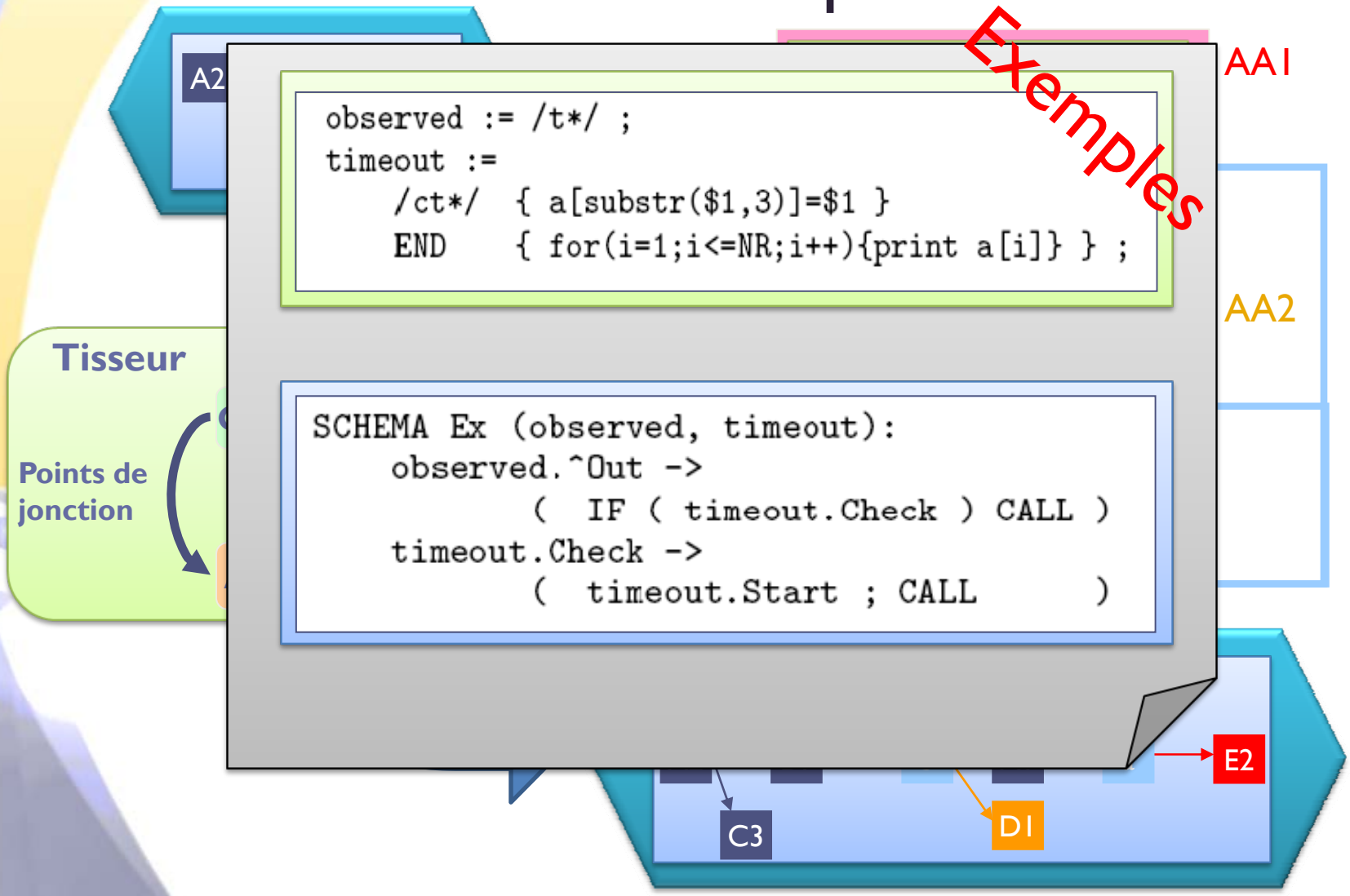
- Activation du Service sur le Conteneur
- Manipulation de l'interface de contrôle depuis le « Device Spy » d'Intel
  - Ajout/retrait de composants et de connecteurs
  - Abonnement aux événements du conteneur
- Création d'une interface fonctionnelle pour le Service composite
  - Sonde source (événement UPnP)
  - Sonde puit (action UPnP)
- Utilisation des Designers :
  - Designer UPnP (sensible à l'évolution de l'infrastructure)
  - Designer Console (Textuel) (adaptation par script)

3. Un mécanisme d'adaptation dynamique basé sur les AA

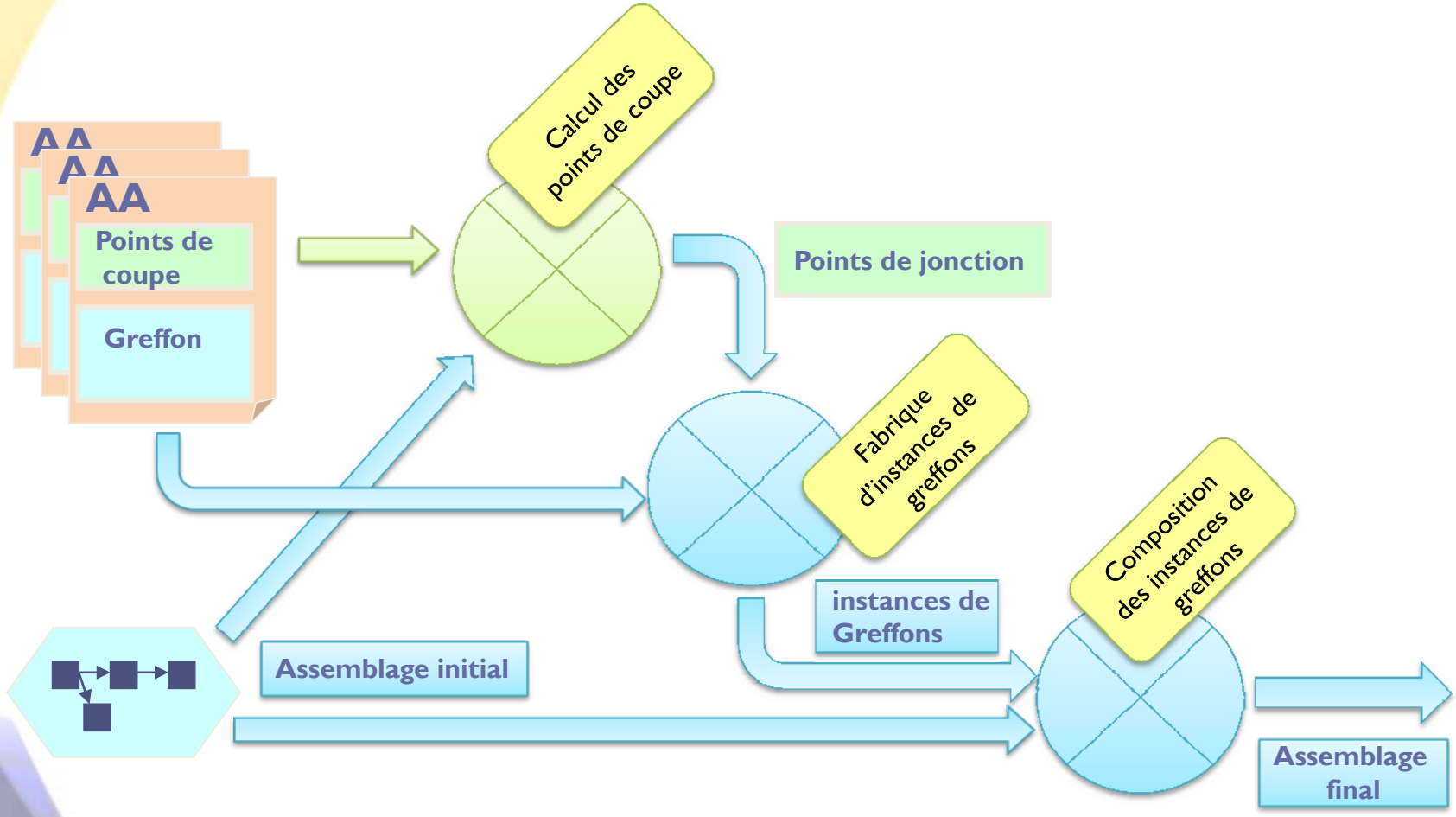


1. Une infrastructure logicielle de services pour dispositifs dynamique (basée sur UPnP dans WComp .Net)
2. Deux niveaux de composition dynamique (LCA et SLCA)
3. Un mécanisme d'adaptation dynamique basé sur les AA

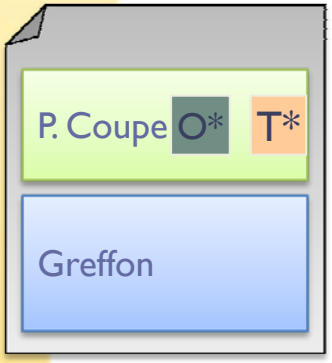
3. Aspect d'Assemblage (AA) Principes



3. Architecture fonctionnelle du Tisseur d'AA

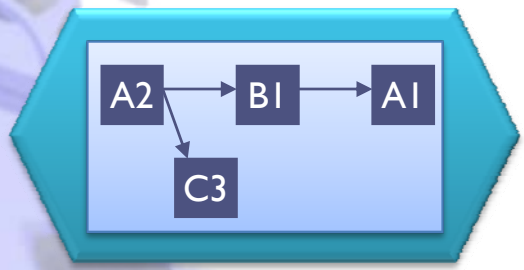


3. AA: Langage et Calcul du point de coupe




```
observed := /t*/ ;
timeout :=
    /ct*/  { a[substr($1,3)]=$1 }
    END    { for(i=1;i<=NR;i++){print a[i]} } ;
```

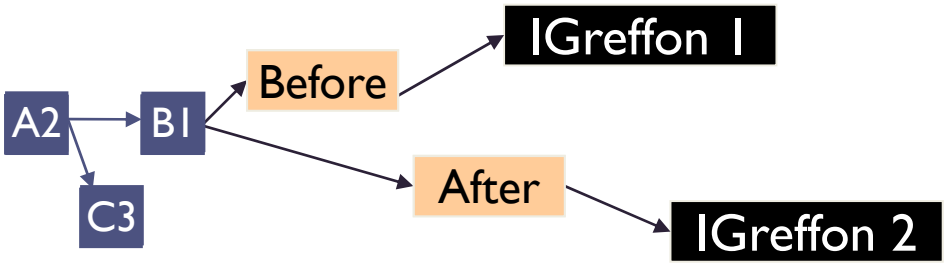
ct3, ct2, ct1, …



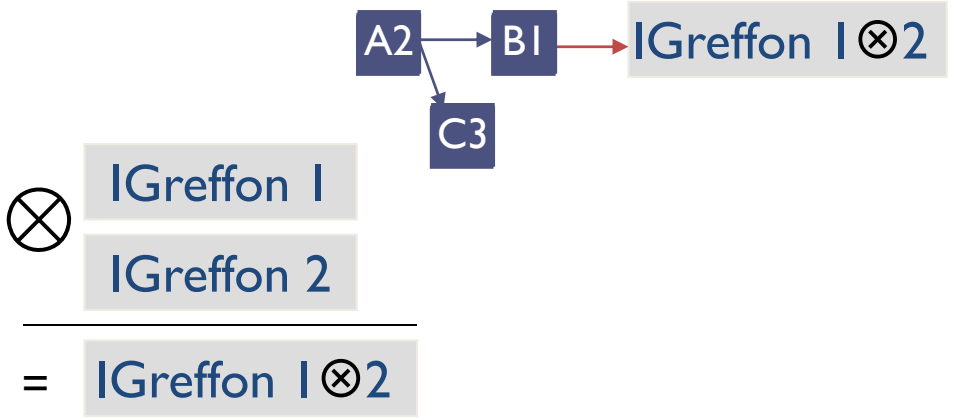
Après avoir parcouru la liste des composants de l'assemblage

3. AA: Fabrique d'i-greffons, Composition et Gestion des conflits

- A l'instar des approches classiques AOP :
  - Composition externe entre les greffons



- Notre approche intègre une logique de fusion :
  - On connait partiellement la sémantique du greffon
  - Instances de greffons fusionnés
  - Logique de Fusion dotée de propriétés



3. AA : Une logique de Fusion et ses propriétés

- Fusion à partir de règles logiques et modifié selon les langages de greffons
- Exemple de propriétés de la composition / fusion inspirées d'ISL

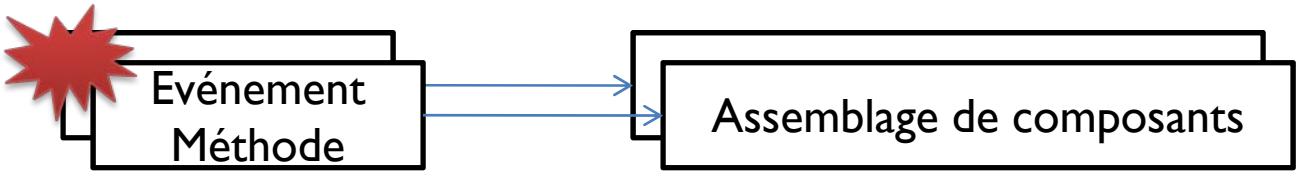
Commutativité : $AA0 \otimes AA1 = AA0 \otimes AA1$
Associativité : $(AA0 \otimes AA1) \otimes AA2 = AA0 \otimes (AA1 \otimes AA2)$

- Permet au concepteur de ne pas se soucier de l'ordre d'application des Aspects d'Assemblage
- Auquel on rajoute

Idempotence : $AA0 \otimes AA0 = AA0$

3. AA: le premier langage pour greffon ISL4WComp

- ISL4WComp pour gérer la concurrence au niveau de la diffusion d'événements et réécriture de méthode



3. AA: le premier langage pour greffon ISL4WComp

Composants spécifiques :

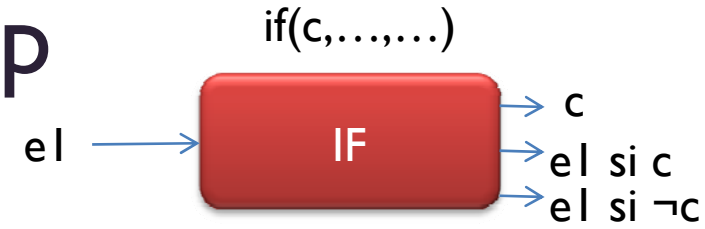
Condition, séquence, parallélisme

Opérateurs internes :

call, delegate, nop

Exemple :

C1.^op → seq(C2.a,call)



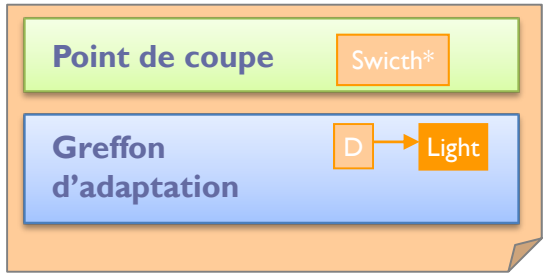
Redirection d'un événement selon que c renvoie vrai ou faux

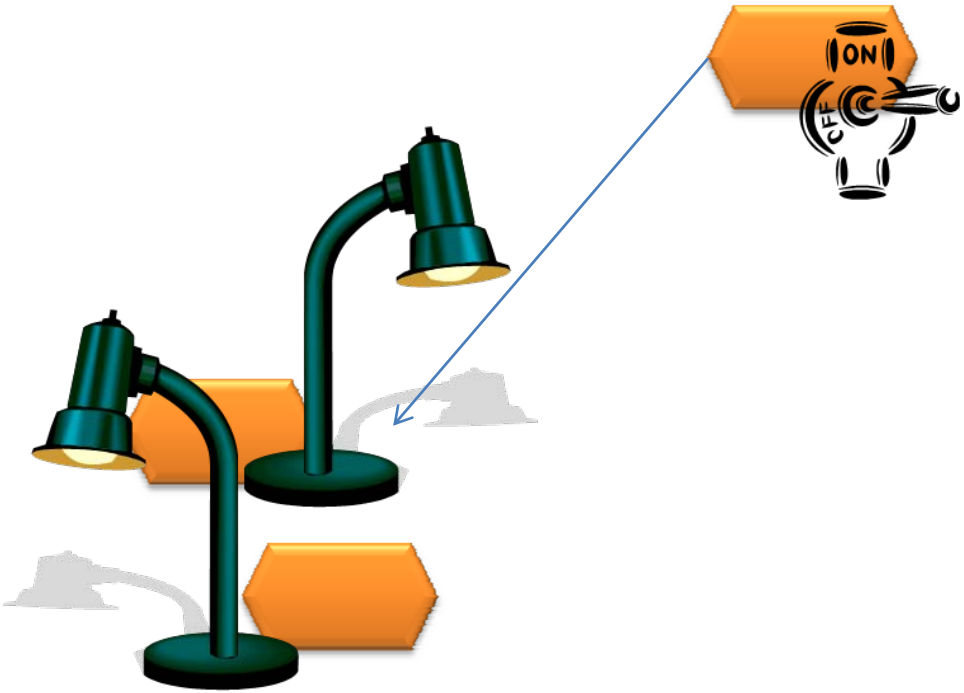
Déclenchement de el sur la 1ère sortie, ensuite sur la seconde

Déclenchement de el sur les deux sorties

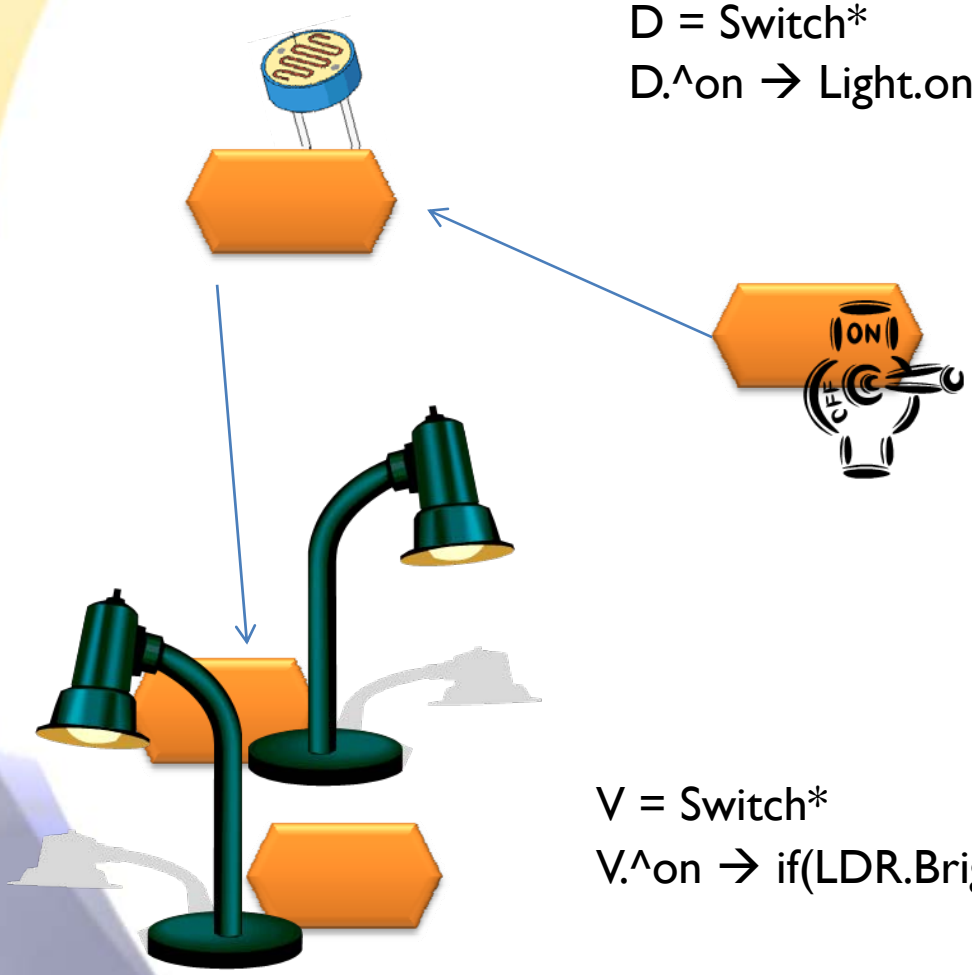
3. Exemple de fusion ISL4WComp

D = Switch*
D.^on → Light.on

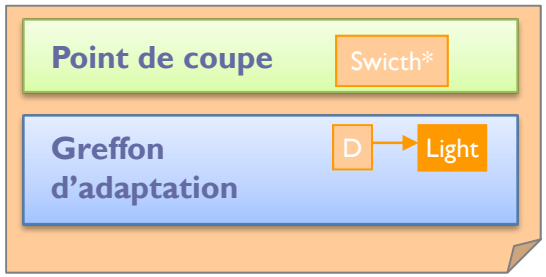




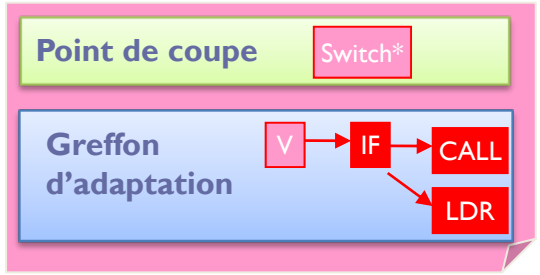
3. Exemple de fusion ISL4WComp



D = Switch*
D.^on → Light.on



AA1



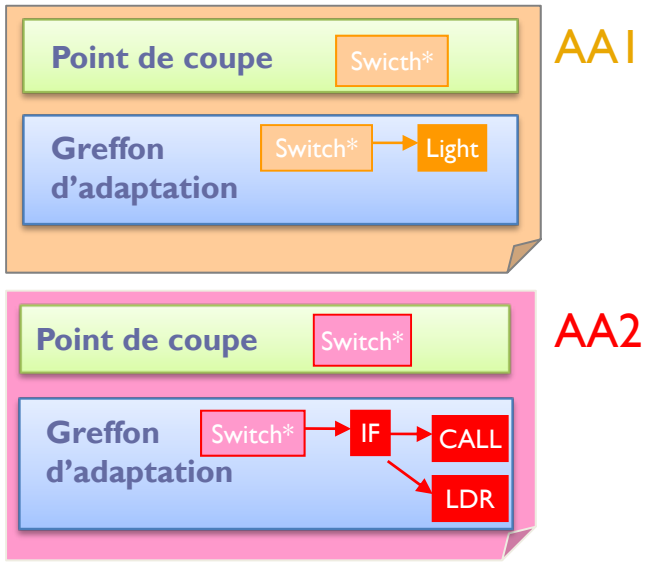
AA2

V = Switch*
V.^on → if(LDR.Bright) nop else call

3. Résultat de la fusion ISL4WComp

D = Switch*
D.^on → Light.on

V = Switch*
V.^on → if(LDR.Bright) nop else call



Switch.^on → Light.on + if(LDR.Bright) nop else call
→ if(LDR.Bright) Light.On + nop else Light.On + call [Classement]
→ if(LDR.Bright) nop else Light.On [Règles finales]

3. Démonstrations

- Ecriture des premiers AA (pour « switch » et « light »)
  - Greffon avec connecteur simple
  - Création de composant dans le greffon
  - Greffon avec connecteur complexe
- Exemple de fusion de greffons
  - Exemple simple (apparition du par)
- Autre exemple de fusion sur greffon complexe
  - avec les opérateurs if, call, seq (introduction du lecteur RFID)

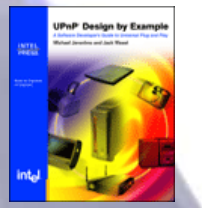
4. Références Bibliographiques

- Référence principale :

  - J.-Y. Tigli, S. Lavirotte, G. Rey, V. Hourdin, D. Cheung, E. Callegari, M. Riveill "*WComp middleware for ubiquitous computing : Aspects and composite event-based Web services*" dans la revue Annals of Telecommunications, éditeur Springer Paris, ISSN 0003-4347 (Print) 1958-9395 (Online), Vol. 64, No 3-4, March-April 2009

- Références UPnP :

  - Vincent Hourdin, Stéphane Lavirotte, Jean-Yves Tigli. "*Service UPnP pour dispositifs autonomes*" vol. H5002, Techniques de l'Ingénieur, feb 2007



  - UPnP* Design by Example *A Software Developer's Guide to Universal Plug and Play* by Michael Jeronimo and Jack Weast, Intel Press

4. Références Bibliographiques

- Référence SLCA :
  - [2008] Vincent Hourdin, Jean-Yves Tigli, Stéphane Lavirotte, Gaëtan Rey, Michel Riveill, "SLCA, Composite Services for Ubiquitous Computing", in International Conference on Mobile Technology, Applications and Systems, septembre 2008.

- Référence AA :
  - [2007] Daniel Cheung-Foo-Wo, Jean-Yves Tigli, Stéphane Lavirotte et Michel Riveill. « Self-adaptation of event-driven component-oriented Middleware using Aspects of Assembly ». Dans 5th International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC), California, USA, novembre 2007.

- Thèse de Doctorat :
  - Daniel Cheung-Foo-Wo, Adaptation Dynamique par Tissage d'Aspects D'assemblage, mars 2009.