Usage des langages de script pour des composants adaptables

Didier Donsez
Université Joseph Fourier (Grenoble 1)
IMAG/LSR/ADELE
didier.donsez@imag.fr
didier.donsez@ieee.org

Sommaire

- Rappels
- Motivations
- Proposition : Modèle de composants scriptables
- Réalisations
- Conclusion et perspectives

Rappel sur les langages de script et les langages dynamiques

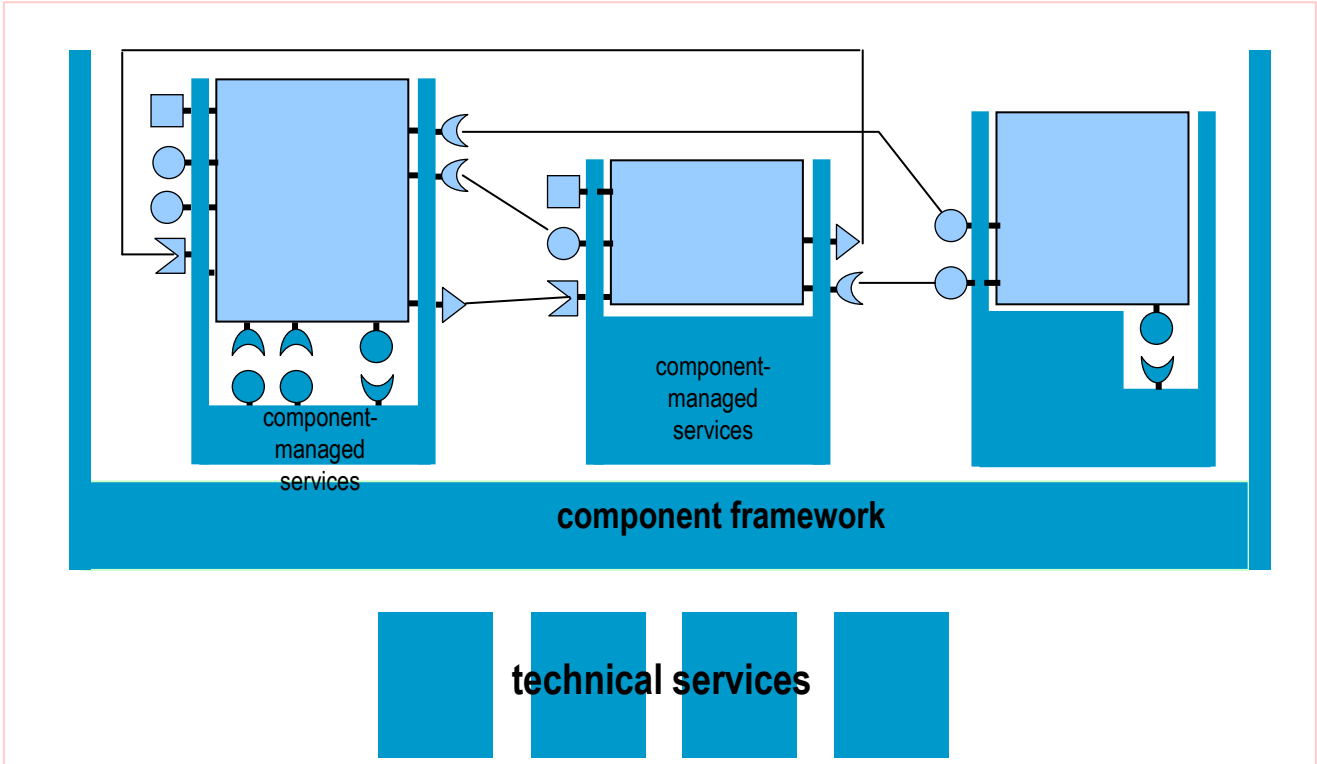
- « Ma Définition »
  - Famille de langages supportant la simultanéité du développement et de l'exécution
    - Boucle interactive, « interprétation », …

Vous pouvez ne pas être d'accord !

- Les plus
  - Comportement évoluant « sans impact » sur l'état
  - Largement adopté/utilisé par les développeurs (novices)
  - Langages métiers (*sysadmin, webmaster, comptable, …, geek*)
- Les moins
  - Mise au point et maintenance difficiles dans les gros codes
- Le Zoo
  - **JavaScript, PHP, ActionScript , VB,** ANT, Groovy, Ruby, Python, Perl, BeanShell, *Self* …
  - ~110 langages au dessus de la Java Virtual Machine et autant sur la CLR de .NET

Rappel sur les composants

- Définition de Szyperski
  - *“Un composant logiciel est une unité de composition ayant des interfaces spécifiées de façon contractuelle et possédant uniquement des dépendances de contexte explicites. Un composant peut être déployé de manière indépendante et il est sujet à composition par des tierces parties.”*



Rappel sur les composants

- Nombreux implémentations
  - JavaBeans, MBeans, EJB, Julia / AoKell, SCA, CCA …
- Les plus
  - Réutilisation, Architecture, Séparation des préoccupations …
- Les moins
  - cycle « développement, déploiement, exécution »

Motivations

- Programmation incrémentale/empirique des composants
  - Contexte non connu à l'avance (avant déploiement)
  - Prototypage rapide de maquettes
  - Programmes à usage unique (workflow Web Services. . .)
  - Programmes incomplets (unfinished computing)
  - Adaptation dynamique/*Autonomic computing* (règles ECA, …)

→ Evitez le cycle « développement, déploiement, exécution »

- Utilisation de scripts dans de gros projets logiciels maintenables grace aux composants.

Propositions

- #1 Pourquoi pas utiliser des langages de script ?

- #2 Quel impact peut avoir la possibilité de faire évoluer le comportement ?
  - Ajout/retrait/évolution dynamique de fonctionnalités
  - Ajout/retrait/évolution dynamique de facettes/réceptacles
  - Ajout/retrait/évolution dynamique de attributs
  - Ajout/retrait/évolution dynamique de contrôles
  - …

Propositions

- Modèle de composants scriptables
  - Administration (Contrôleurs)
  - Types
  - Prototypes

Contrôleurs

- FacetController (FC)

  String[] getFacetNames();

  void setFacet (String name, String[] interfaces, Map properties, Handler handler);

  String[] getFacetInterfaces(String name);

  Map getFacetProperties(String name);

  void removeFacet(String name);

- ReceptacleController (RC)

  String[] getReceptacleNames ();

  void setReceptacle (String name, String[] interfaces,
  String filter, boolean multiplicity);

  String[] getReceptacleInterface (String name);

  String getReceptacleFilter (String name);

  void removeReceptacle (String name);

Contrôleurs

- PropertyController (PC)

```
String getPropertyNames();
Object getProperty(String name);
void addProperty(String name, Object value)
                throws ImpedanceMismatchException, ClassCastException;
void removeProperty(String name);
```

- ScriptController (SC)

```
void setLanguage(String language) throws UnsupportedLanguageException;
void concatScript(String text) throws SyntaxErrorException
void concatScript(Reader reader) throws SyntaxErrorException;
String getScriptText();
void reset();
void reset(String language) throws UnsupportedLanguageException;
```

Types et Prototypes

Types

- Définitions
  - CBSE: « { facettes, réceptacles, propriétés *(, contrôleurs)* } »
  - SOA: « { facettes *(,propriétés)* } » (cf Contrat de service)
- Le type d'une instance d'un composant évolue indépendamment des autres.

Prototypes

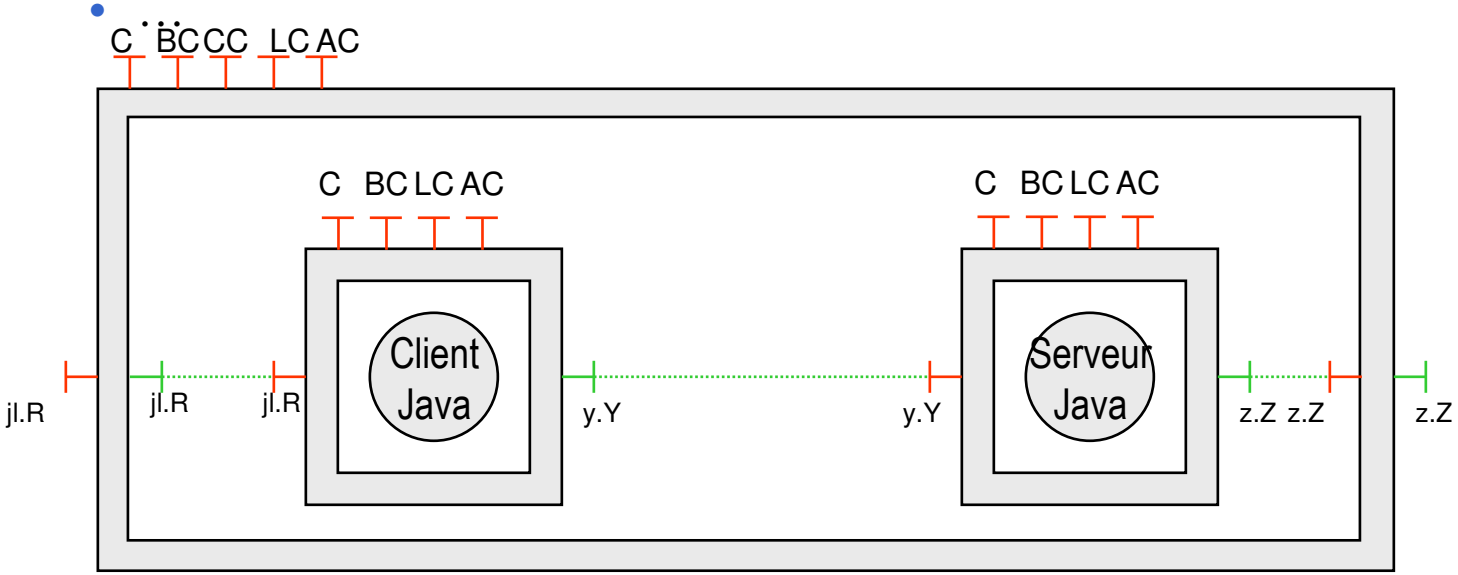
- Fabrique d'instances de composants à partir d'un type initial
- *template* = type + comportement
- Clonage
- Cf langages à prototype (*Self, JavaScript, …*)

Mises en pratique

- CBSE : Fractal
- SOA : OSGi (Declarative Service)

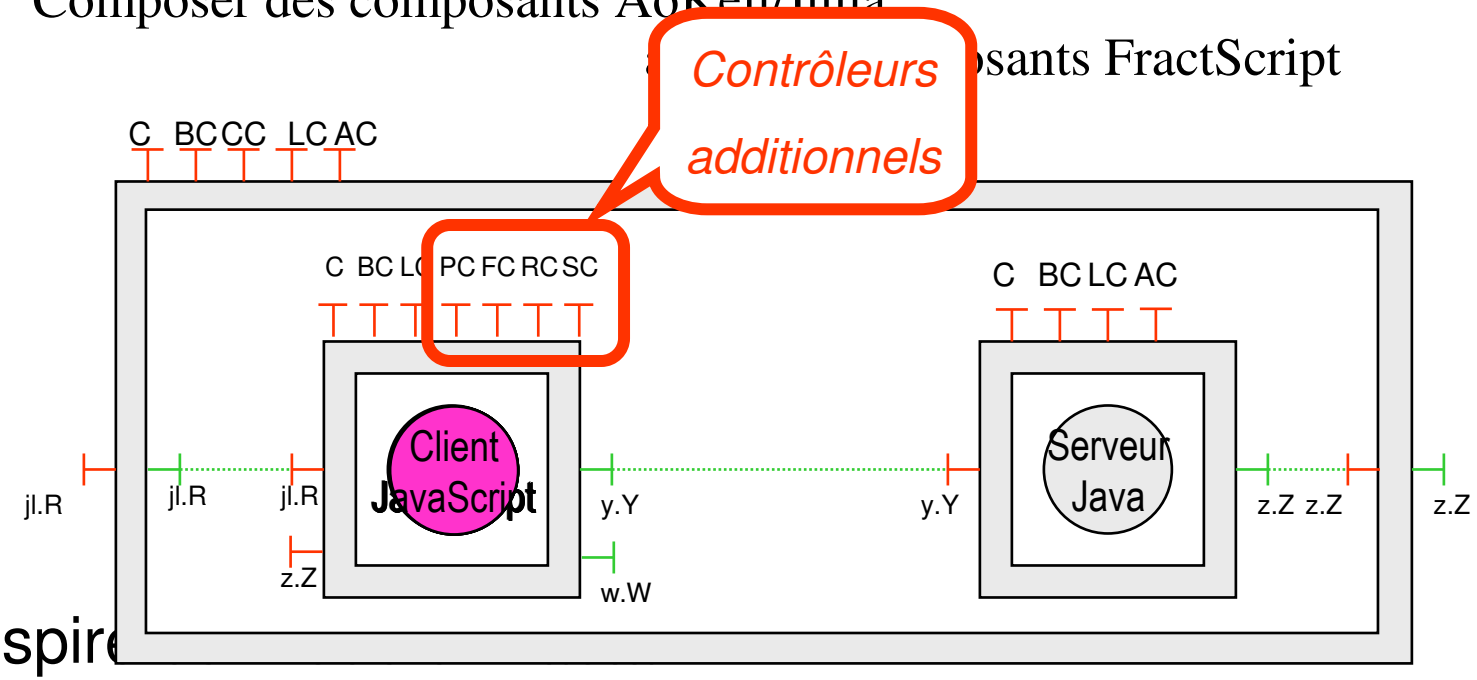
FractScript (i)

- Fractal
  - Modèle des composants hiérarchiques
    - Primitifs, Composites, Types, Factories, …
  - Nombreuses implémentations pour différents langages
    - Julia, AoKell, ProActive, Fractalk, FractNet, …
  - Nombreux travaux
    - Reconfiguration et adaptation dynamiques (FScript, AutoFractal, …)
  -



FractScript (ii)

- Motivation
  - Composant Fractal développé avec un langage de script
  - Composer des composants AoKell/Julia
- Inspir



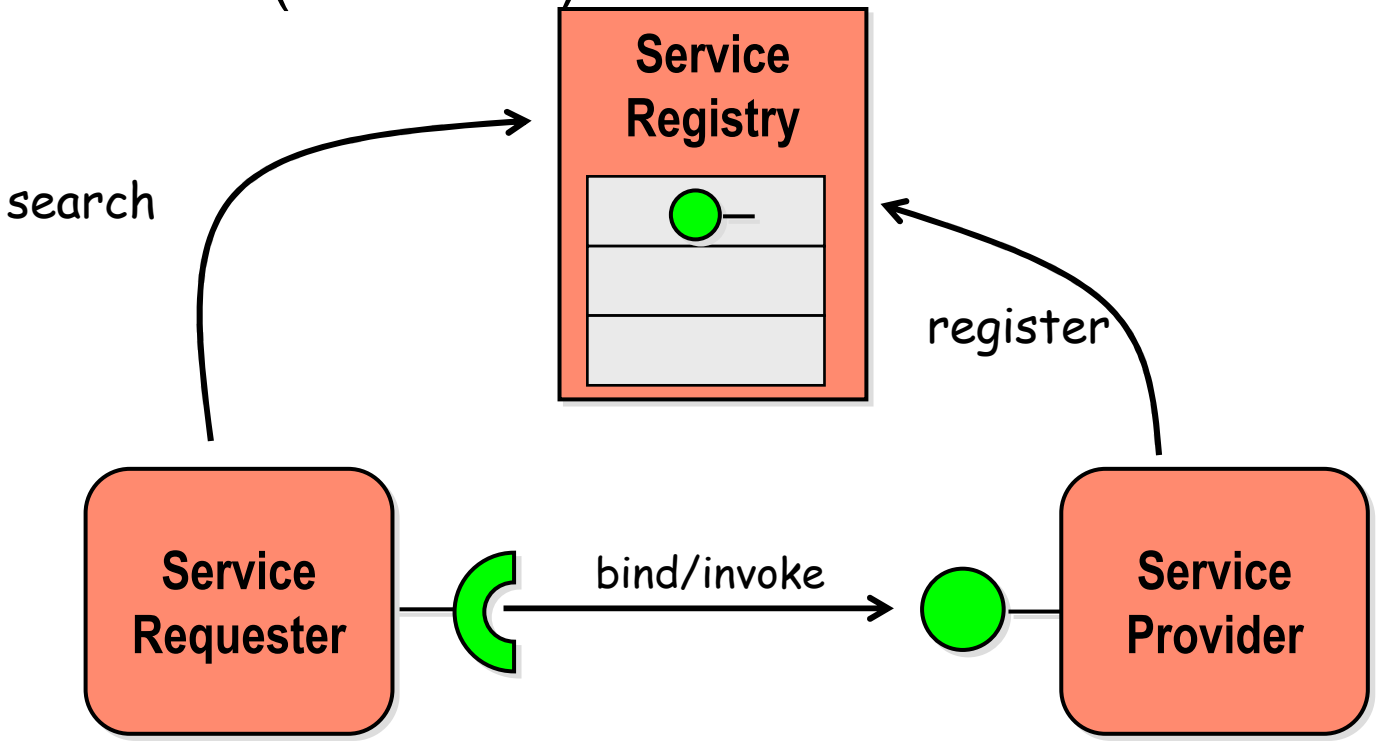
  - Pas de type de composant, mais des prototypes de composant
  - Langages : JSR 223 (JavaScript, PHP, Groovy, …)

SCRScript

- OSGi
  - Plateforme de déploiement en continu (dynamique)
    - ie « *Super-ClassLoader* »
  - Plateforme SOA dynamique
    - Services colocalisées sur une JVM
  - Success story : BMW, Eclipse, Lotus, … JEE, JSR 277
- SCR (Service Component Runtime)
  - Modèle de composants orientés services
    introduit dans la spécification OSGi R4 (JSR 291)
  - Reconfiguration dynamique + Gestion du cycle de vie

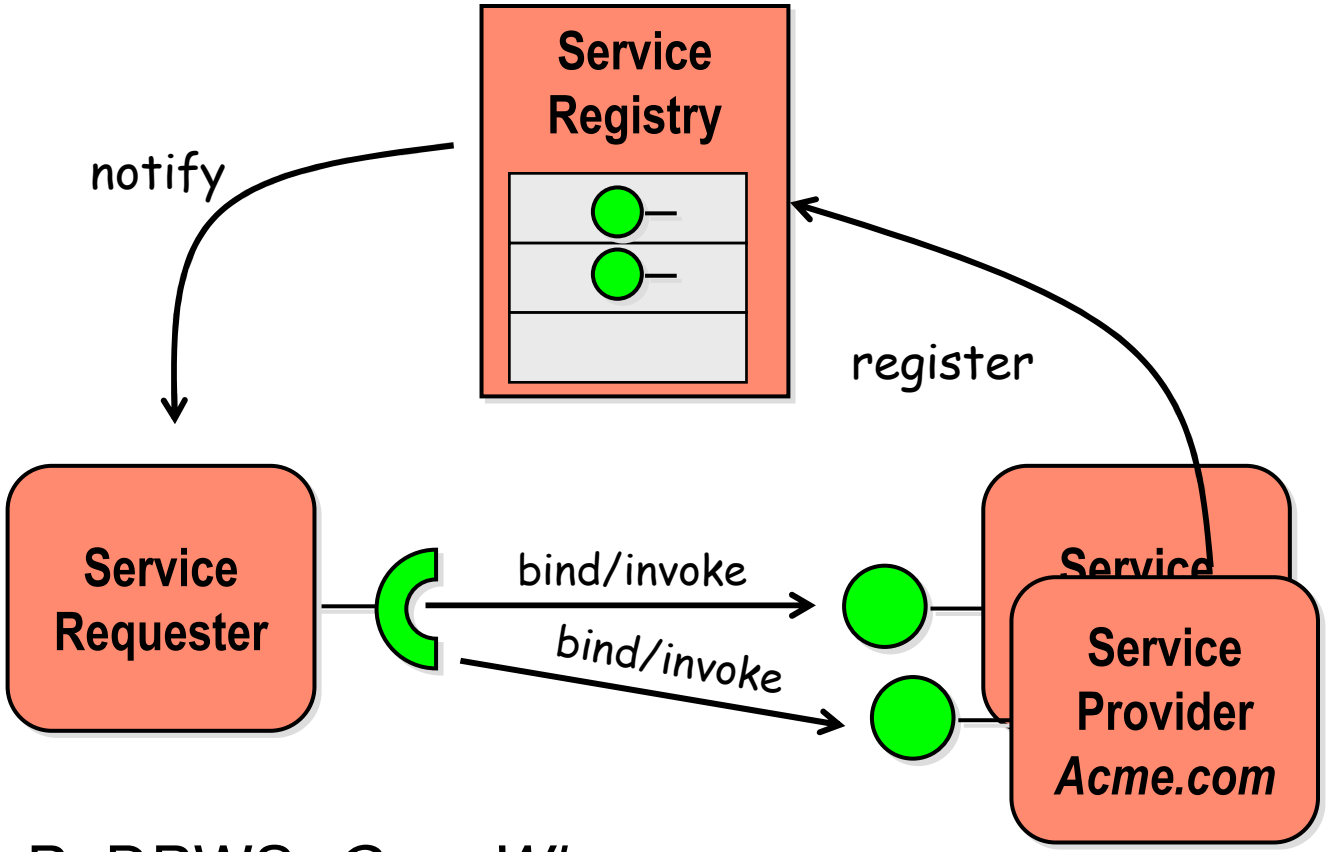
Rappel: Architecture orienté service (SOA)

- Les services (contrats) sont « invariants »



Rappel: Dynamic SOA

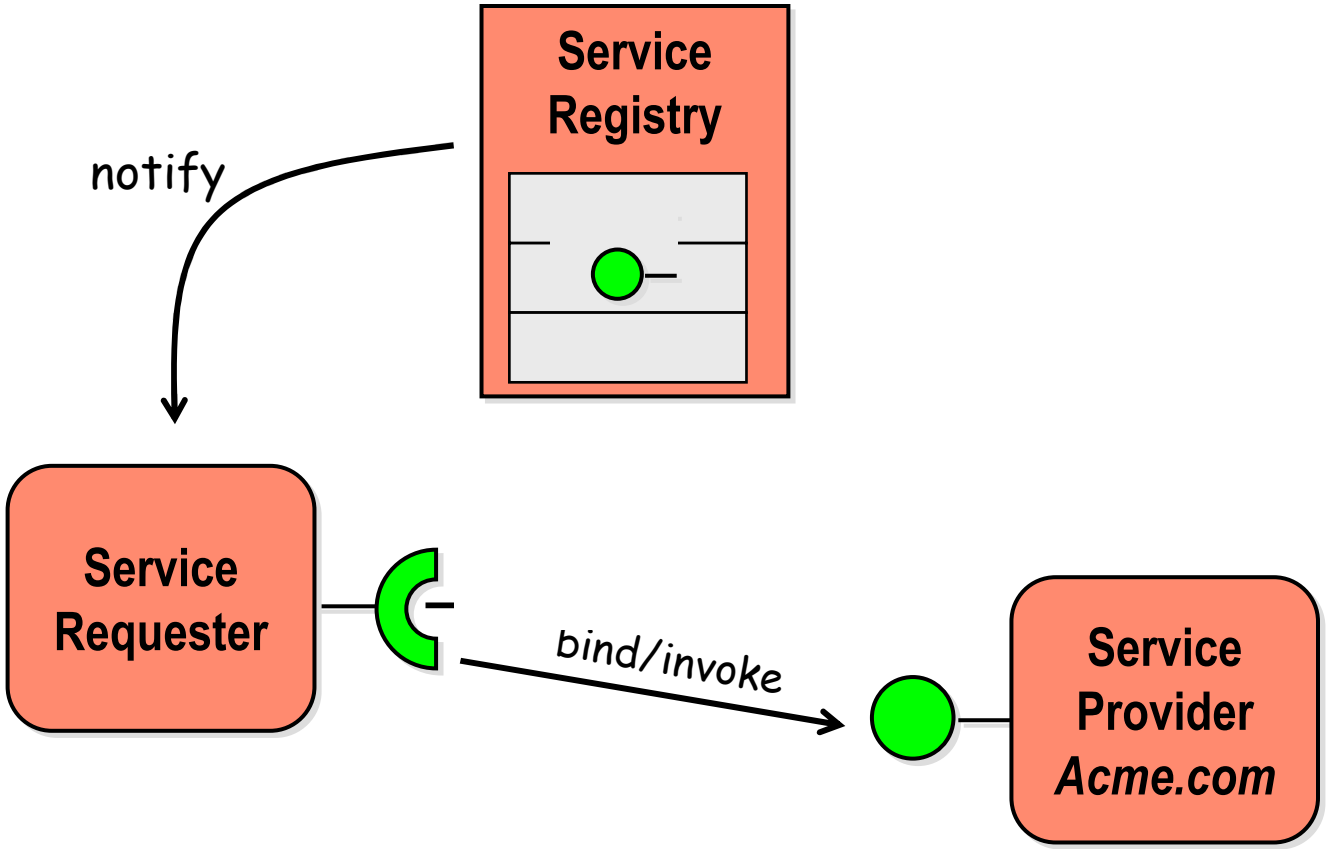
- Arrivée dynamique de nouveaux services



- JINI, UPnP, DPWS, *OpenWings*
- OSGi

Rappel: Dynamic SOA

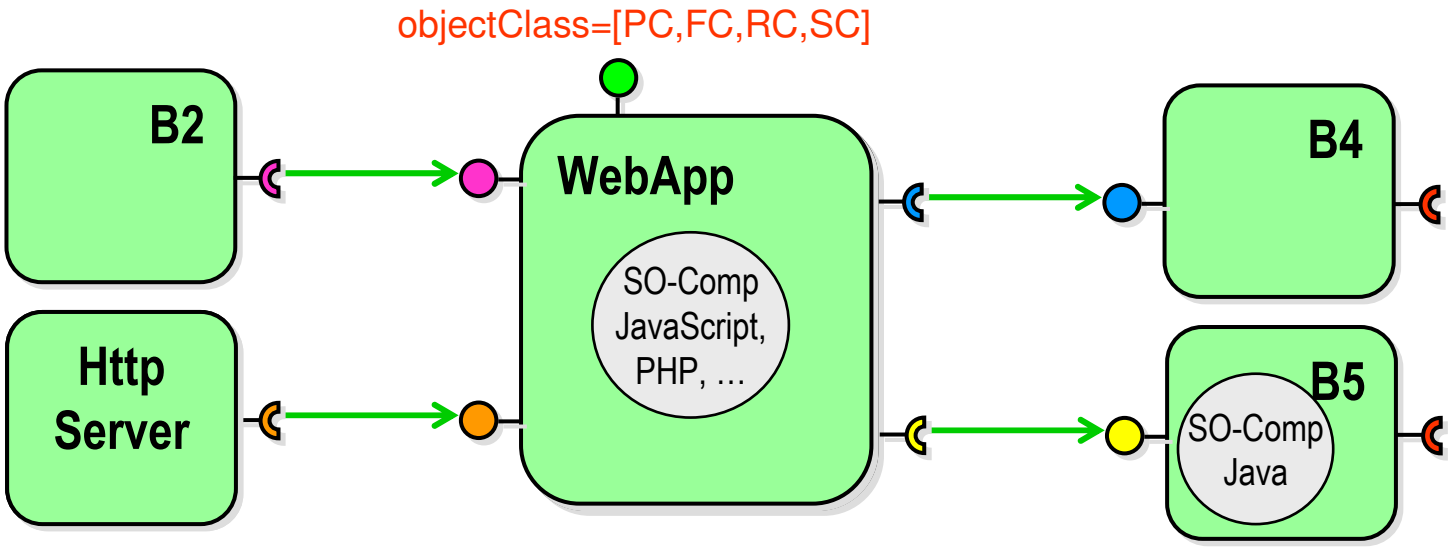
- Retrait dynamique de services utilisés



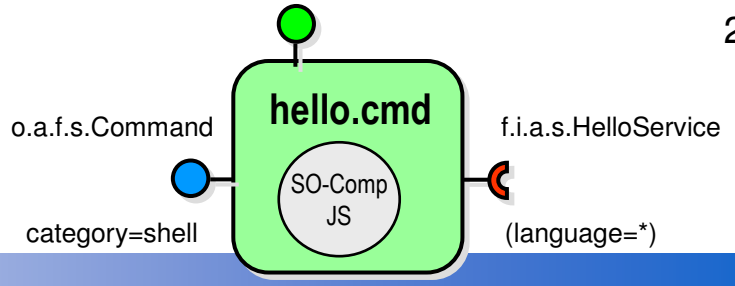
- JINI, UPnP, DPWS, *OpenWings*
- OSGi

SCRScript

- SCR (Service Component Runtime)
  - Modèle de composants orientés services
    introduit dans la spécification OSGi R4 (JSR 291)
  - Reconfiguration dynamique + Gestion du cycle de vie
- SCRScript



Snippet SCRScript




```
<?xml version="1.0" encoding="UTF-8"?>
<component name="hello.cmd">
 <implementation language="javascript"
  entry="SCR-INF/hellcmd.js"
 >

 <property name="category" value="shell"
     type="String"/>
 <service>
  <provide interface
            ="org.apache.felix. shell.Command"/>
 </service>

 <reference name="HELLO"
  interface="fr.imag.adele.service.HelloService"
  target="(language=*)"
  cardinality="1..n"
  policy="dynamic"
  bind="bindHelloService"
  unbind="unbindHelloService"
 />
</component>
```

```
// the Java reference to the HelloService
var services;

// Command functions
function getName(){
  return "hello";
}

function getUsage(){
  return "hello <msg>";
}

function getShortDescription(){
  return "invoke a HelloService";
}

function execute(commandLine,out,err){
  if(services!=undefined){
    for(s in services){
      msg=services[s].sayHello(commandLine.substring(
                java.lang.String(getName()).length()));
      out.println(msg);
    }
  }
}

// Binding/unbinding functions for 0..n or 1..n
function bindHelloServiceMultiple(svcref,svc){
  services[svcref]=svc;
}

function unbindHelloServiceMultiple(svcref,svc){
  services[svcref]=undefined;
}
```

Performances

JACBenchmark

- Intel Centrino 1,7 Ghz, 1 Go de RAM, sur Windows XP avec JVM 1.5.0 de SUN

Canevas	Temps (ms)	Ratio /Java Direct
Fractal/AOKell 2.0 (Spoon based)	261	1,5
Fractal/Julia 2.1.1 (option MERGEALL)	467	2,7
Fractal/AOKell 1.0 (AspectJ based)	523	3,0
FractScript/JavaScript (Rhino engine)	7234	42,1
Pure Java direct (JVM 1.5.0)	172	1,0
Pure Java proxy (JVM 1.5.0)	203	1,2
Pure Java Direct (AOT compilation with cygwin gcj 3.4	4879	28,4

Conclusion

- On s'y attendait
- Plutôt à utiliser pour à les débuts de pile des applications
- Ou alors quand le frontal est très couteux à « traverser »
  - HTTP Service, SOAP Servlet, JMX MBeanServer, RMI Local entre domaine d'application …

Conclusion & Perspectives

Conclusion

- Evolution dynamique des composants
- 2 implémentations
- Disponibilité
  - http://www-adele.imag.fr/users/Didier.Donsez/dev/fractscript
  - http://www-adele.imag.fr/users/Didier.Donsez/dev/osgi/scrscript

Perspectives

- **Fort potentiel des langages dynamiques**
- « Java Language <> Java Platform » *Gilad Bracha, JavaOne 2005*
- Java 6.0 → JSR 223 javax.script
- Java 7.0 → support JVM pour les langages dynamiques (*invokedynamic*)

Question(s) ?

Bonus Track

Livres sur Amazon.com

- Java 11636
- JavaScript 4117
- PHP 3572
- Ruby 1092
- Groovy 357

Perspective : SCRScript dans Cocoon 3.0 / OSGi