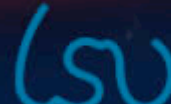


Alligator - plate-forme d'intégration Logicielle pour outils de vérification

Clément DÉMOULINS
clement.demoulins@lip6.fr

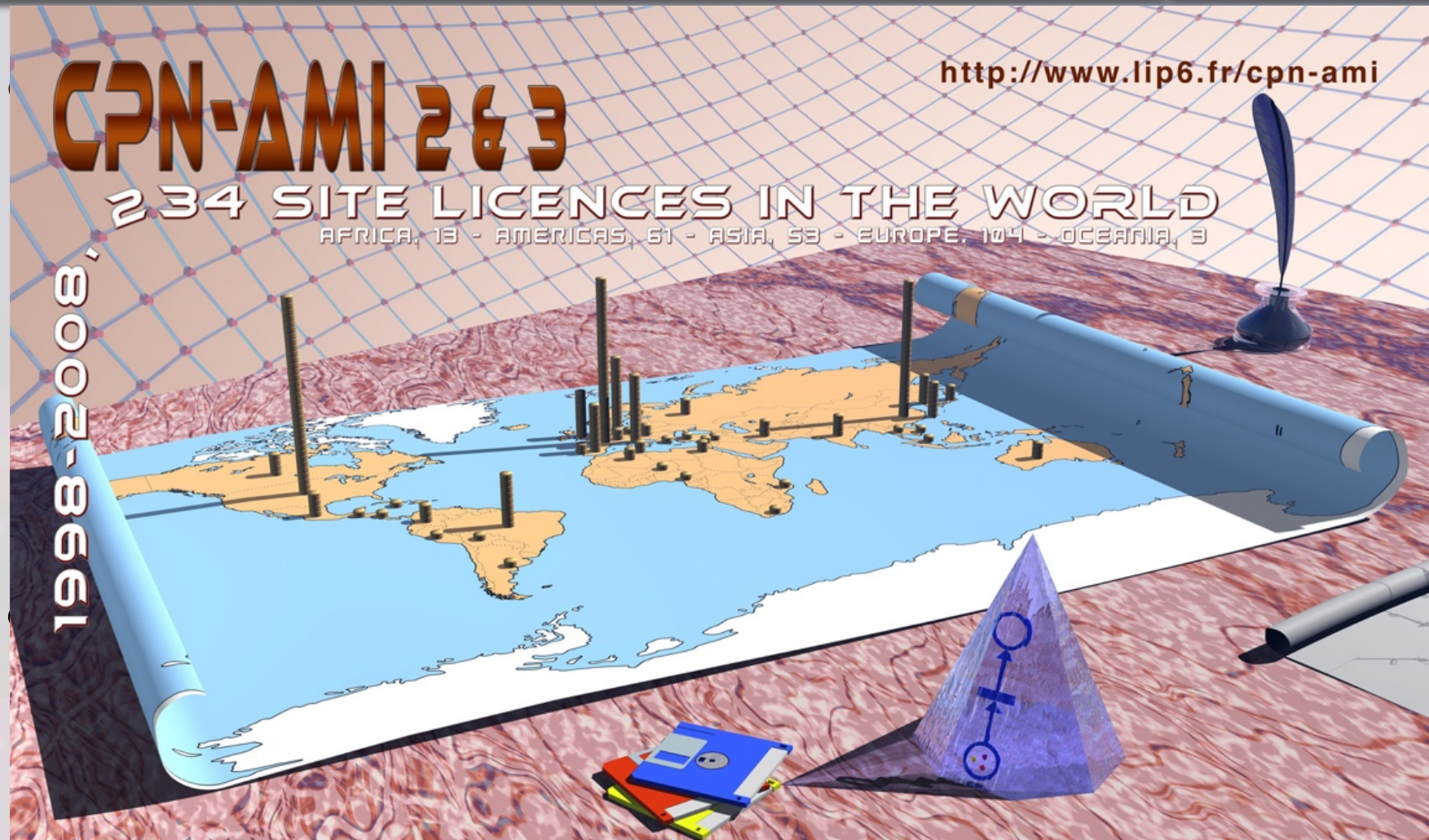
Lip6 — MoVe — NEOPPOD

30 septembre 2011





Introduction



CPN-AMI is a free Petri Net based CASE environment gathering tools designed at LIP6 and tools designed by academic partners: Universities of Genova and Torino (GreatSPN), Helsinki University of Technology (prod), Bell Labs (dot), Universität Humbolt zu Berlin (LoLA), Technische Universität München (PEP), Université Paris 13 (modsog & mcsog).





Introduction

- Historique: AMI (1987-1994) FrameKit (1995-2011)

CPN-AMI 3.1
Mastering Large Specifications
<http://www.lip6.fr/cpn-ami>

Model & Assemble
Configure your Model for Verification
Use Verification Tools
Use P/T nets only Tools
PNML

PetriScript
Language to build PN modules and assemble them
Optimized model checker on the symbolic reachability graph
SnowSpot
Optimized Unfolder
Uses Data Decision Diagrams to scale up with unfolded models
PNML Converter
Implements the ISO/IEC 15909-2 standard (PNML)

CPN-AMI
Modeling and Verification of Very Large Systems with Symmetric Petri nets
<http://www.lip6.fr/cpn-ami>

Macao User Interface on Mac OS X
Coloane User Interface (Eclipse plug-in) on Windows, Linux and Mac OS X

Tool server (Mac OS X and Linux)

Structural Analysis
P & T Invariants
Structural Bounds
Synchon and Traps
Prefix Graph Unfolding

Model Checking
Symbolic State Space
Symmetry Reduction
Decision Diagrams
Temporal Logics (LTL/CTL)
Parallel Model Checking

Script-based Modeling
PNML import and export (ISO/IEC-15909)

Symbolic State Space Generation
up to 50x speedup on a 20 bi-Xeon Hyper-Threaded Cluster

CPN-AMI
Complex Systems Verification Platform
<http://www.lip6.fr/cpn-ami>

MC-SOG: LTL Model Checker based on Symbolic Observation Graphs
Hybrid structure preserving LTL-X
Explicit graph + symbolic nodes

Mod-SOG: Modular Construction of the Symbolic Observation Graph
Three phases algorithm
Building SOG of individual modules
Product of SOGs
Reduction with respect to the formula

New release of the Coloane User Interface
Open Source & Multi-Platform
Windows, MacOS, Linux
Graphical editing support
Grid, Guide, Zoom, Alignment, etc.
Import/Export in PNML, CAMI, DOT, PGF
Integrated to the Eclipse project structure
Automated update system

More Efficient Symbolic Model Checking with libddd
Hierarchical Set Decision Diagrams
Features
Automatic saturation: 1 to 3 orders of magnitude over CUDD
Hierarchical and/or recursive descriptions enabled
Logarithmic solution to some problems: 2^{20000} dining philosophers
Easy construction of efficient model checkers using homomorphisms
Open source C++ library

- ouverture vers une «approche cloud»



Introduction

- Historique: AMI (1987-1994), FrameKit (1995-2011)
 - Expression d'un besoin de factorisation
 - 20 ans de (presque) accumulation de services: CPN-AMI
 - ▶ 2+17 versions entre janvier 1993 et octobre 2010
 - Approche client/serveur
- Mais...
 - Limite du système + sclérose de vieux code
 - ouverture vers une «approche cloud»



Introduction

- Alligator, le futur?
 - Besoin d'une «plate-forme MeFoSyLoMa»
 - Factorisation des outils développés en Île-de-France
- Elaboration dans le cadre du projet Neoppod
 - Utilisation de «technologies standards»
 - Premières expérimentations en vue d'une démonstration en Juin 2011 (PETRI NETS)



Introduction

- Enjeux
 - Finaliser Alligator
 - Enrichir Alligator
 - ▶ Nouveaux formalismes & nouveaux outils
 - Étendre Alligator
 - En faire une base pour la visibilité des travaux dans MeFoSyLoMA



Plan

- I. Introduction
- II. Formats
- III. Architecture
- IV. Service Crocodile
- V. TP



Objectifs

- Évolutions de la plateforme FrameKit (base de CPN-AMI) :
 - Extension des formalismes
 - Simplification du protocole de communication
 - Décentralisation



Formats



Pourquoi un nouveau format ?

- CAMI
 - Non structuré
 - Complexe
 - «Maison» et ancien (soucis de maintenance)
- PNML
 - Réseau de Petri uniquement



Nouveaux formats

Meta-Meta

FML

conform to

Meta

**User
Formalism**

specialize

GML

instance of

Instance

User Model

conform to
is structured by



Nouveaux formats

Meta-Meta

FML

conform to

Meta

**User
Formalism**

specialize

GML

instance of

Instance

User Model

conform to
is structured by



Nouveaux formats

Meta-Meta

FML

Relax-NG

conform to

Meta

User Formalism

specialize

GML

instance of

Instance

User Model

conform to
is structured by



Nouveaux formats

Meta-Meta

FML

Relax-NG

conform to

Meta

**User
Formalism**

specialize

GML

instance of

Instance

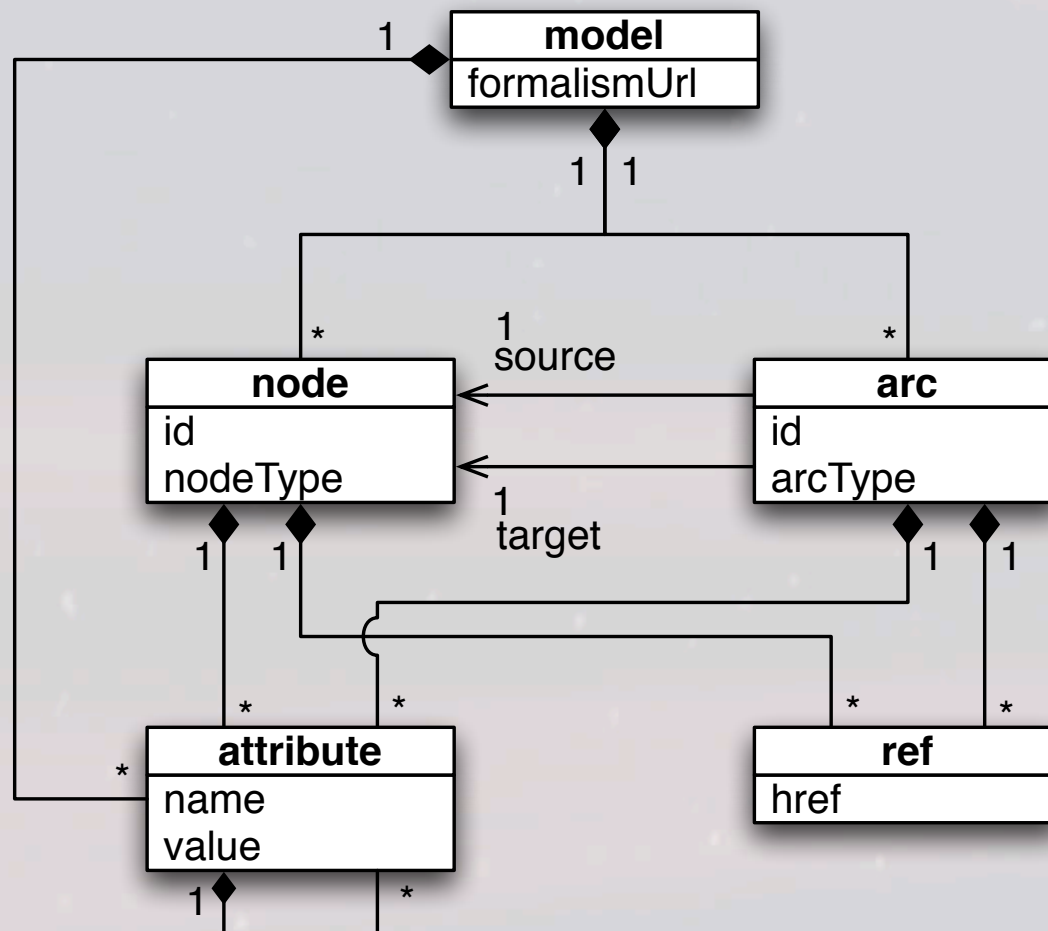
User Model

GML-Check

conform to
is structured by

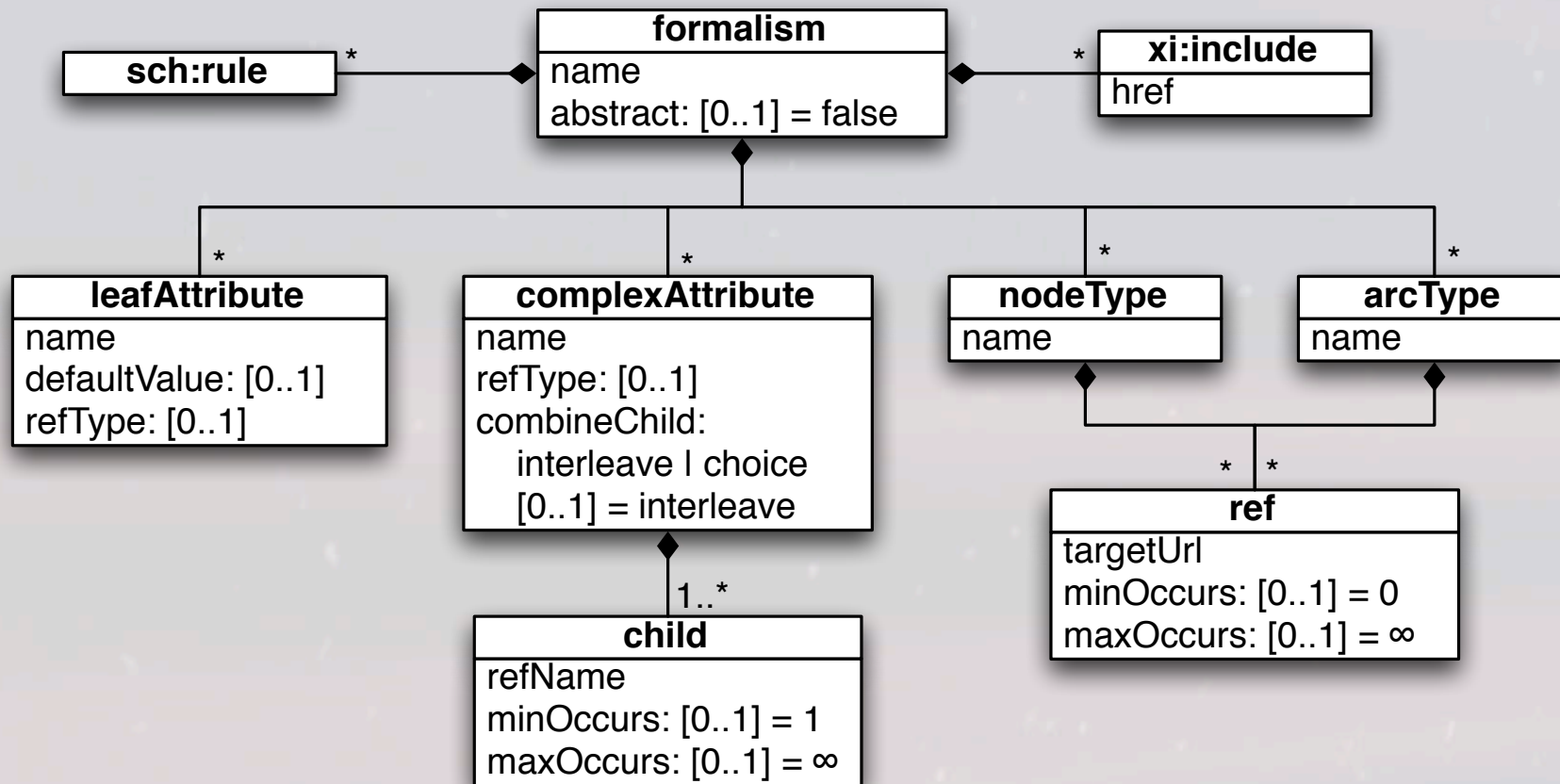


Graph Markup Language (GML)





Formalism Markup Language (FML)



Documentation : <https://srcdev.lip6.fr/trac/research/NEOPPOD/wiki/HowToDefineNewFormalism>



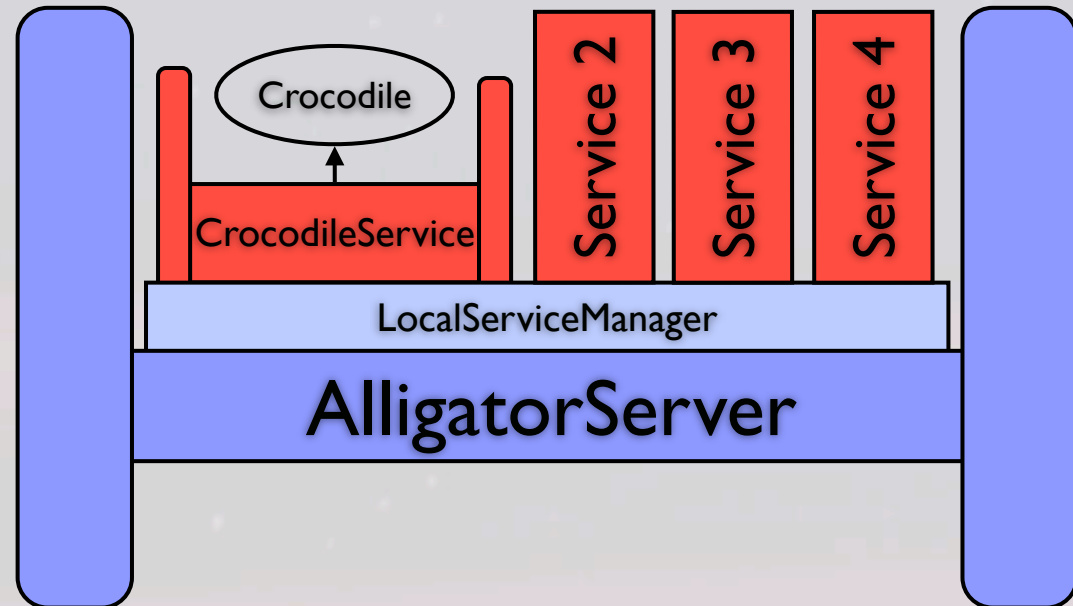
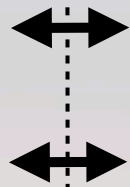
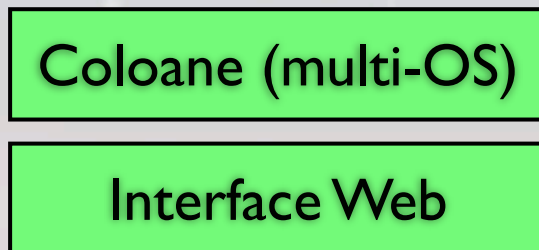
Architecture



Flux

Clients

Serveurs

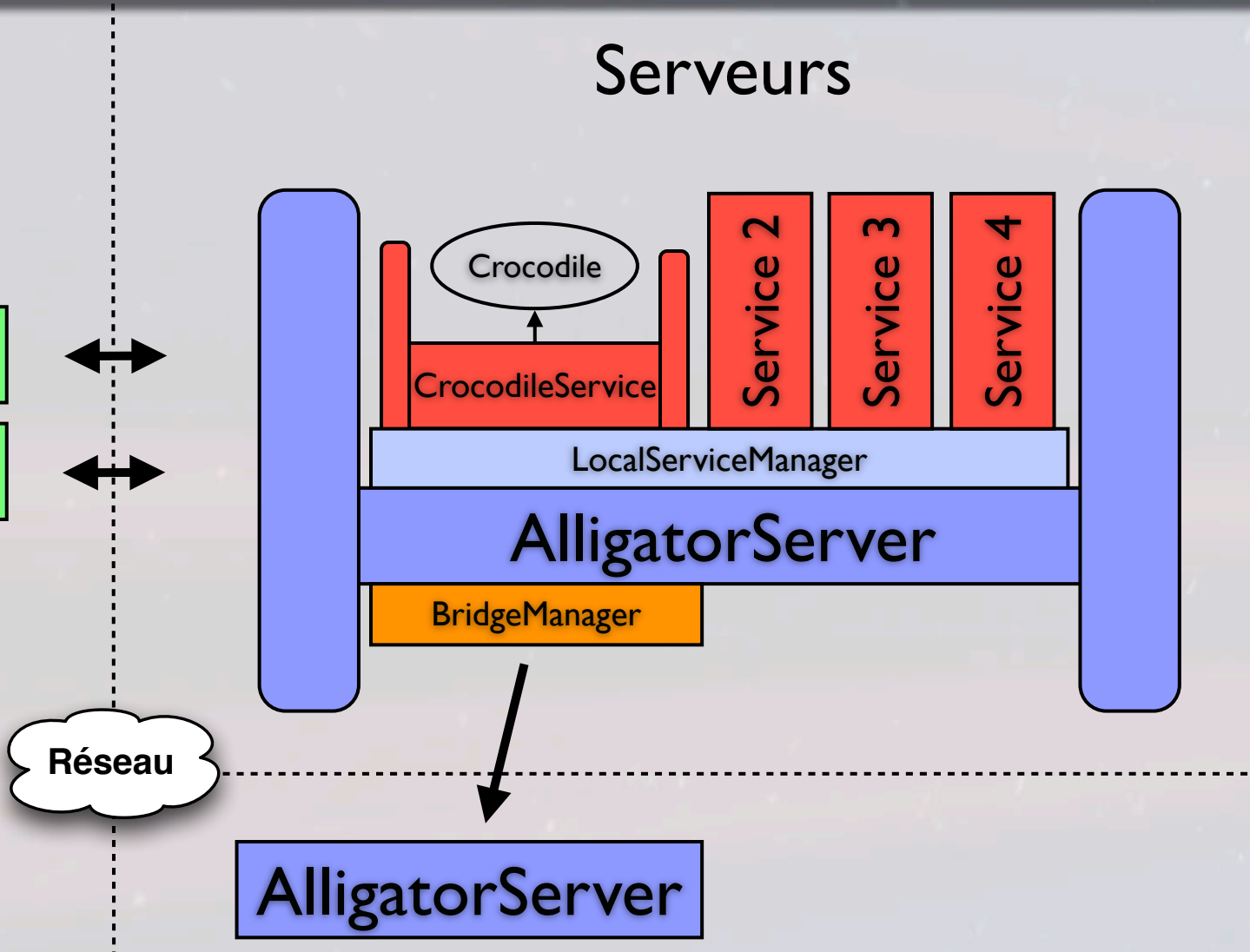
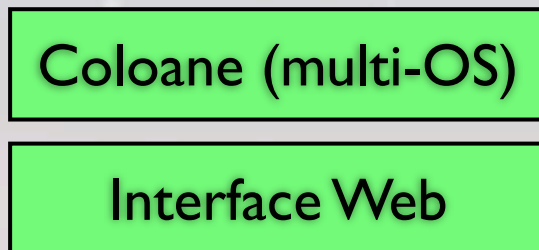




Flux

Clients

Serveurs

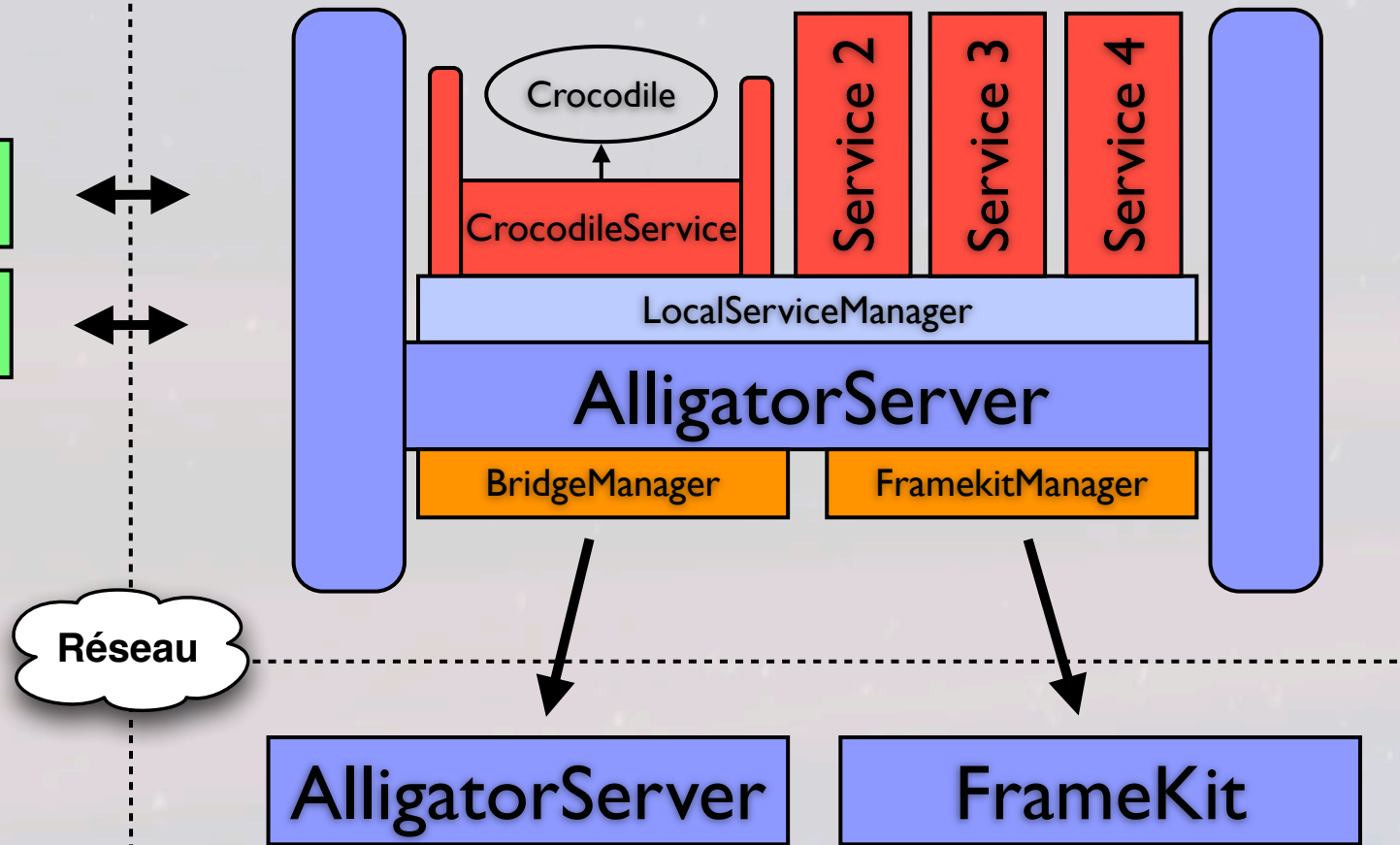
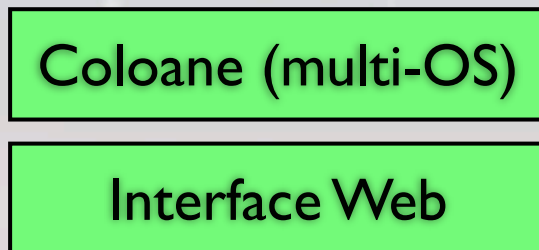




Flux

Clients

Serveurs

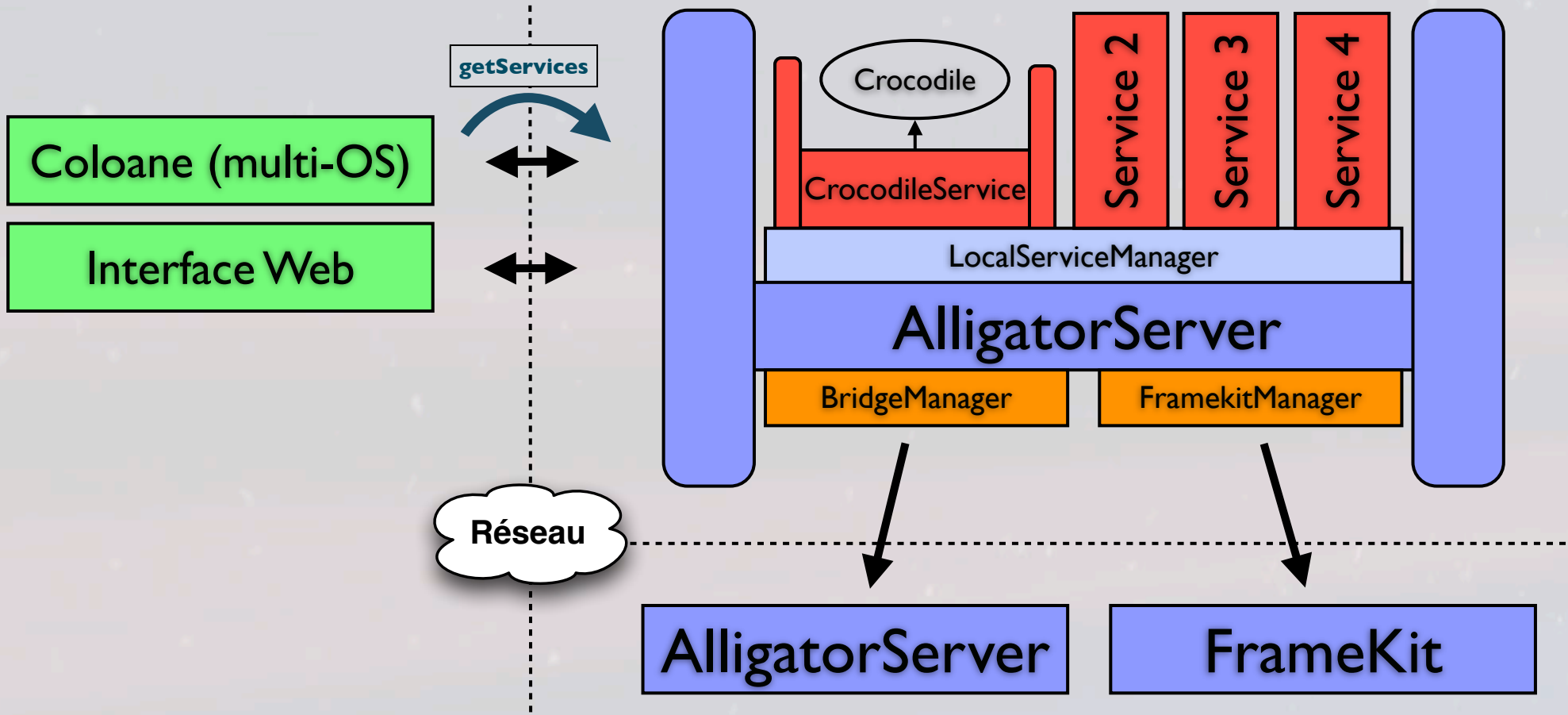




Flux

Clients

Serveurs

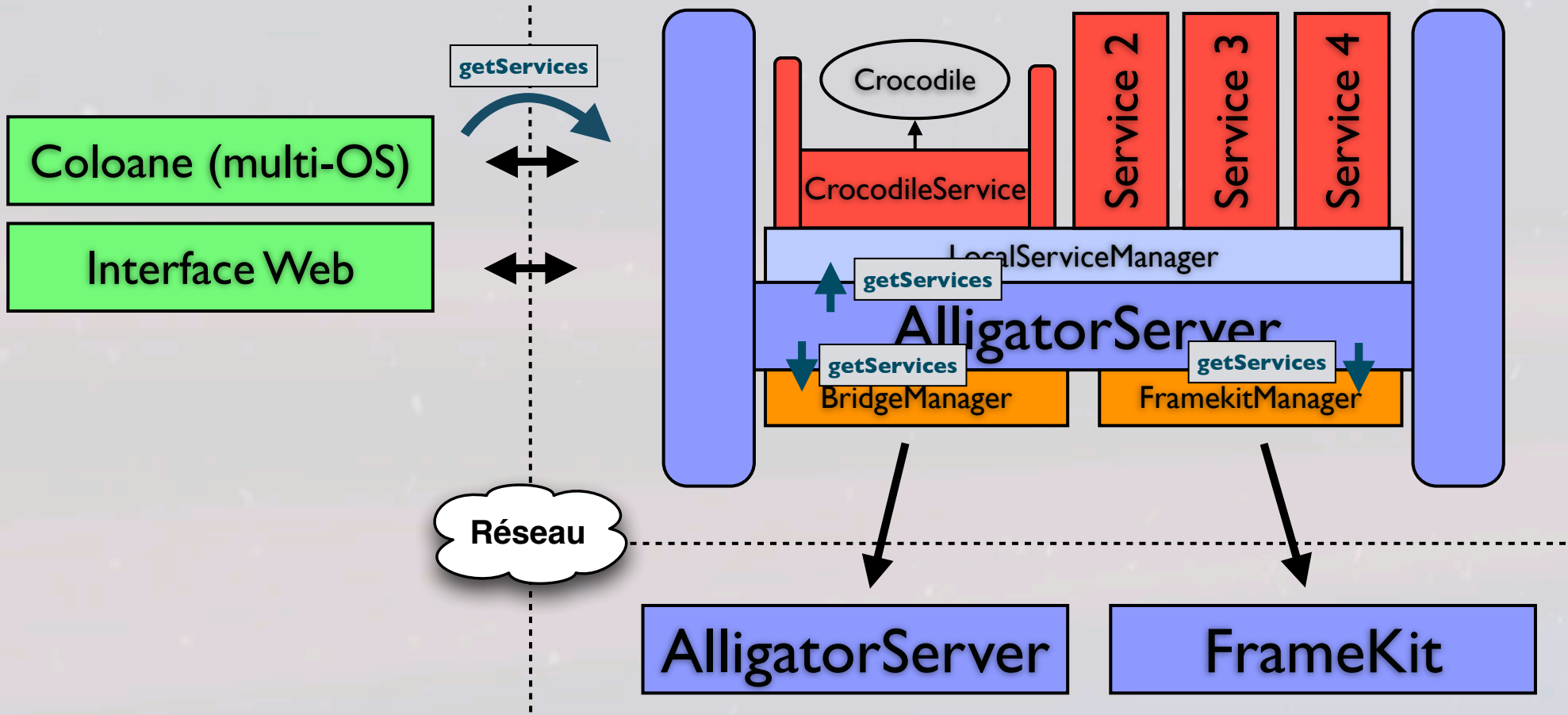




Flux

Clients

Serveurs

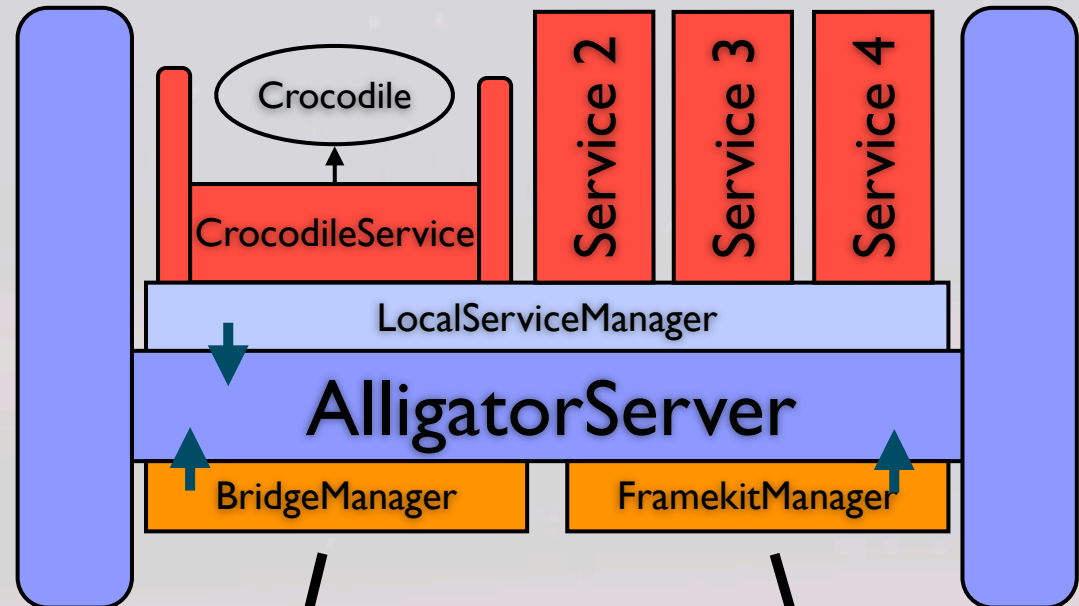
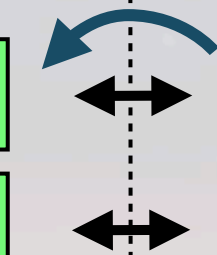
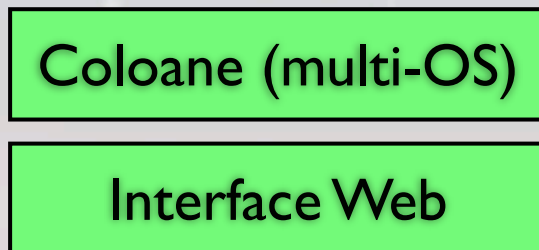




Flux

Clients

Serveurs



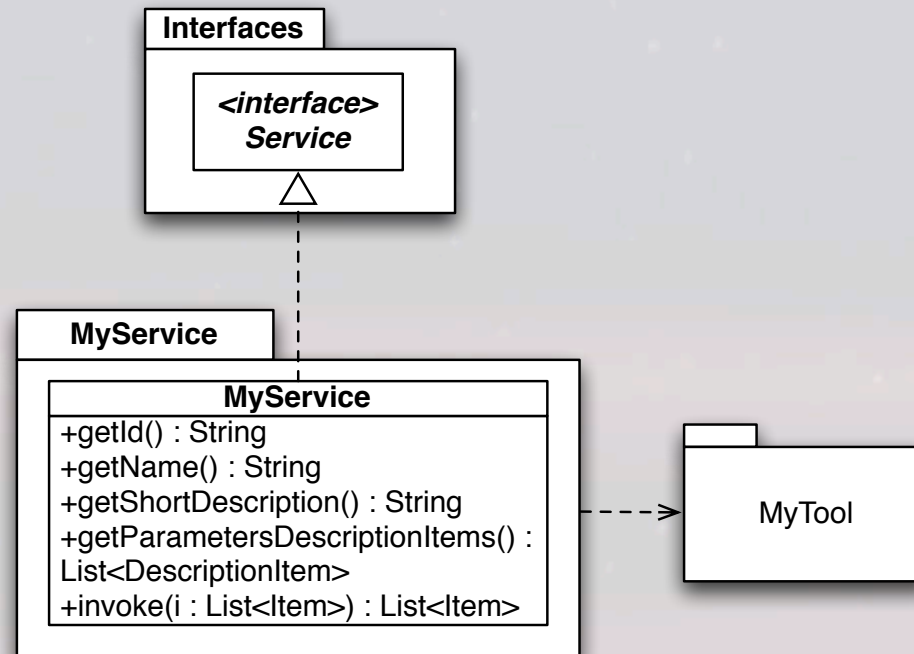


Technologies

- WebService : Communications (clients, serveurs)
- Maven : Gestion configuration (compilation, dépendances, assemblage, déploiement)
- Java : Langage de base
 - ServiceLoader : Extensions (*Service*, *ServiceManager*)

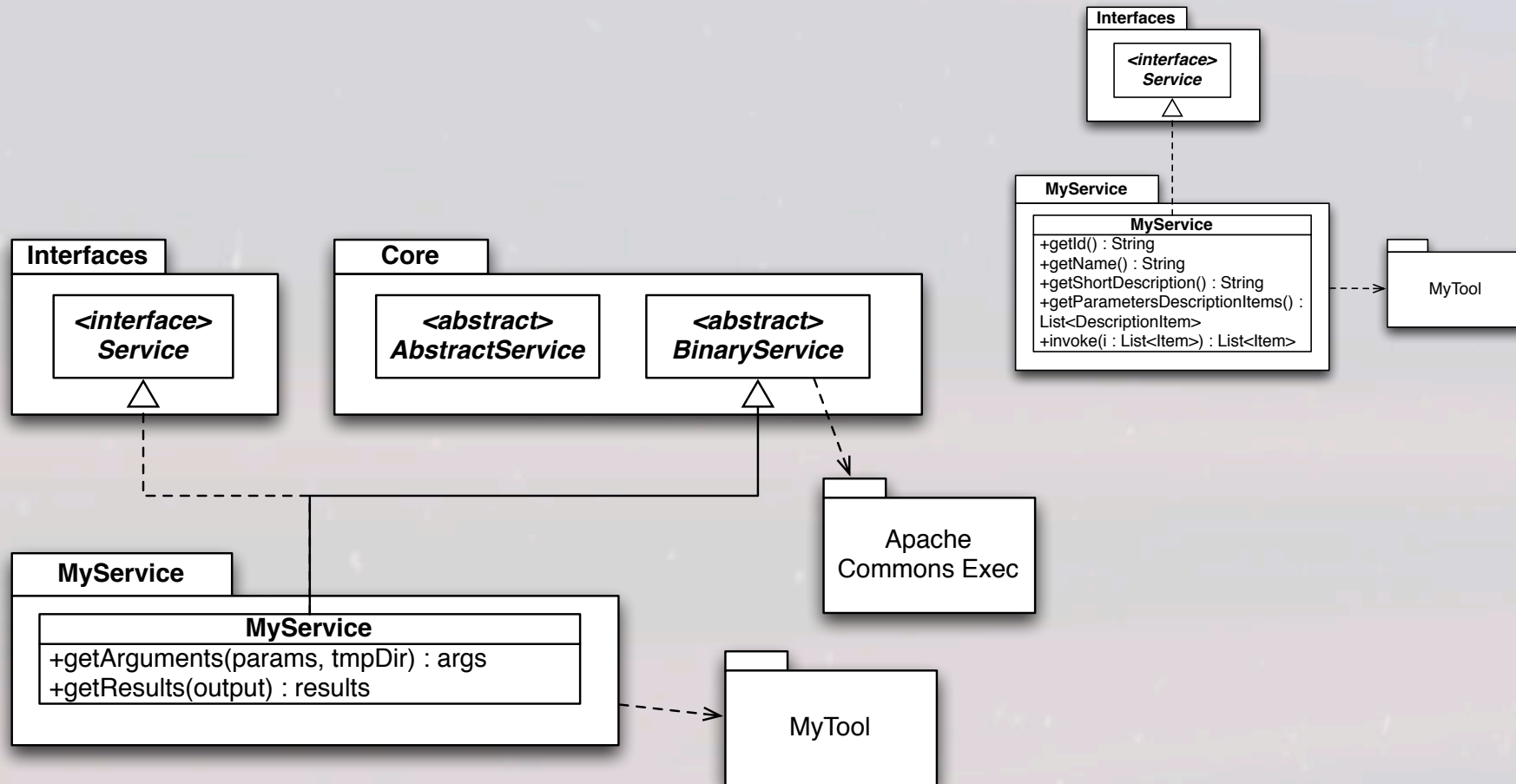


Architecture : Service





Architecture : Service





Service Crocodile

Générateur d'espace d'état



Service Crocodile

- Informations nécessaires (I) :
 1. Identificateur maven : *groupId:artifactId:version*
 2. Identificateur de service
 3. Nom
 4. Description
 5. Liens vers l'outil compiler pour les différentes plateforme gérées



Service Crocodile

New Maven project
Specify Archetype parameters

Group Id:

Artifact Id:

Version:

Package:

Properties available from archetype:

Name	Value
serviceId	crocodile
serviceName	crocodile
serviceClassName	CrocodileService
serviceDescription	Symbolic state space generator
binaryUrlLinux32	https://teamcity-systeme.lip6.fr/guestAuth/repository/download/bt67/.lastSuc
binaryUrlLinux64	https://teamcity-systeme.lip6.fr/guestAuth/repository/download/bt66/.lastSuc
binaryUrlDarwin	https://teamcity-systeme.lip6.fr/guestAuth/repository/download/bt63/.lastSuc

Advanced



Service Crocodile

- Informations nécessaires (2) :
 1. Gérer les arguments (ligne de commande)
 2. Gérer les résultats (sortie standard)



Pause



Séance de TP

Objectif :

Intégrer un outil basique générant des statistiques sur un modèle.



0 - Pré-requis

- Eclipse 3.6 Helios
- Plugin Eclipse : m2eclipse
- Plugin Eclipse : Coloane
 - Module Core
 - Module External Platforms



I - Configuration

- Ajout du dépôt maven contenant Alligator

~/.m2/settings.xml

- *Preferences / Maven / User Settings*
- Lien : settings.xml

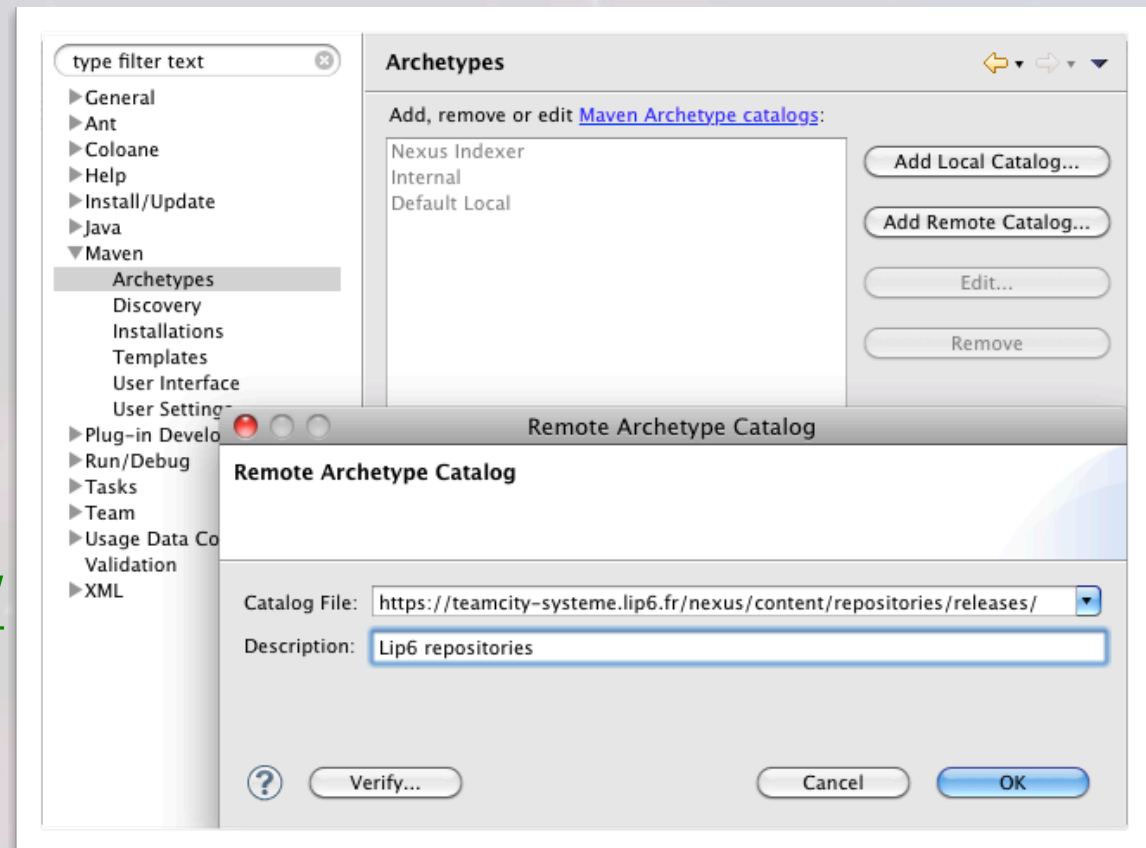
```
<settings>
  <mirrors>
    <mirror>
      <id>nexus</id>
      <name>Repository LIP6</name>
      <url>https://teamcity-systeme.lip6.fr/nexus/content/groups/public</url>
      <mirrorOf>*</mirrorOf>
    </mirror>
  </mirrors>
  <profiles>
    <profile>
      <id>teamcity-systeme</id>
      <activation>
        <activeByDefault>>true</activeByDefault>
      </activation>
      <repositories>
        <repository>
          <id>nexus</id>
          <name>Repository LIP6</name>
          <url>https://teamcity-systeme.lip6.fr/nexus/content/groups/public</url>
          <layout>default</layout>
          <releases>
            <enabled>>true</enabled>
            <updatePolicy>always</updatePolicy>
            <checksumPolicy>warn</checksumPolicy>
          </releases>
          <snapshots>
            <enabled>true</enabled>
            <updatePolicy>always</updatePolicy>
            <checksumPolicy>fail</checksumPolicy>
          </snapshots>
        </repository>
      </repositories>
    </profile>
  </profiles>
</settings>
```



I - Configuration

- Dans les préférences d'eclipse
- Ajout d'un catalogue «d'archetype»

<https://teamcity-systeme.lip6.fr/nexus/content/repositories/releases/>





2 - Création du service

- Nouveau «*Maven Project*»
- Utilisation de l'archetype
`fr.lip6.move.alligator:service-archetype`
 - Création de la structure
 - Configuration basique



2 - Création du service

New Maven project
Specify Archetype parameters

Group Id:

Artifact Id:

Version:

Package:

Properties available from archetype:

Name	Value
serviceId	modelstats
serviceName	modelstats
serviceName	modelstats
serviceClassName	ModelStatsService
serviceDescription	Give some statistics on a model

► Advanced



3 - Récupération de l'outil

- ModelStat est disponible sur le dépôt maven, il faut ajouter la dépendance au fichier de configuration *pom.xml* :

```
<dependency>
  <groupId>fr.lip6.move.psar.ModelStats</groupId>
  <artifactId>modelstats</artifactId>
  <version>1.0.0-SNAPSHOT</version>
</dependency>
```




4 - Invoquer l'outil

- À faire :
 - Récupérer le modèle dans les paramètres
 - Utiliser la méthode *getStatistics* de la classe *ModelStats*.
 - Renvoyer les résultats



4 - Invoquer l'outil

```
@Override
public final List<Item> invoke(List<Item> params) {
    List<Item> results = new ArrayList<Item>();
    try {
        File model = File.createTempFile(ID, ".gml");
        FileWriter writer = new FileWriter(model);
        writer.write(params.get(0).getValue());
        writer.close();
        results.add(new Item(ItemType.TEXT, "Statistics", ModelStats.getStatistics(model)));
        model.delete();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParserConfigurationException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (SAXException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return results;
}
```



4 - Invoquer l'outil

- En cas d'erreurs :
 - Vérifier qu'il ne manque pas d'*import*
 - Toutes les exceptions sont elles gérées :
`try {} catch (Exception e) {}`
 - Utiliser le *quick fix* d'eclipse en laissant le curseur sur l'erreur



4 - Javadocs

Alligator :

<https://teamcity-systeme.lip6.fr/guestAuth/repository/download/bt77/.lastSuccessful/api/index.html>

ModelStat :

<https://teamcity-systeme.lip6.fr/guestAuth/repository/download/bt41/.lastSuccessful/site/apidocs/index.html>



5 - Intégration

```
public ExampleService() {  
    super(ID, NAME, DESCRIPTION);  
    addParameter(new DescriptionItem(ItemType.MODEL, "currentModel"));  
}  
  
{@inheritDoc}  
@see fr.lip6.move.alligator.interfaces.Service#invoke(java.util.List)  
  
Override  
public final List<Item> invoke(List<Item> params) {  
    List<Item> results = new ArrayList<Item>();  
    try {  
        File model = File.createTempFile(ID, ".gml");  
        FileWriter writer = new FileWriter(model);  
        writer.write(params.get(0).getValue());  
        writer.close();  
        results.add(new Item(ItemType.TEXT, "stats", ModelStats.getStatistics(model)));  
        model.delete();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } catch (ParserConfigurationException e) {  
        e.printStackTrace();  
    } catch (SAXException e) {  
        e.printStackTrace();  
    }  
    return results;  
}
```

Run As

- 1 Java Applet
- 2 Java Application
- m2 3 Maven build
- m2 4 Maven build...**
- m2 5 Maven clean
- m2 6 Maven generate-sources
- m2 7 Maven install
- m2 8 Maven test

Date
26/09/11 14:32



5 - Intégration

Edit configuration and launch.

Name:

Main JRE Refresh Environment Common

Base directory:

Goals:

Profiles:

Offline Update Snapshots
 Debug Output Skip Tests Non-recursive
 Resolve Workspace artifacts

Parameter Name	Value



5 - Intégration

- Télécharger et décompresser la plateforme :

<https://teamcity-systeme.lip6.fr/guestAuth/repository/download/bt77/.lastSuccessful/alligator-0.1-SNAPSHOT-bundle.zip>

- Copier le jar 'target/votreservice.jar' dans le dossier 'services'
- Copier l'outil modelstat dans le dossier 'lib'

<https://teamcity-systeme.lip6.fr/guestAuth/repository/download/bt41/.lastSuccessful/modelstats-1.0.0-SNAPSHOT-jar-with-dependencies.jar>

- Démarrer Alligator à l'aide du script alligator.sh



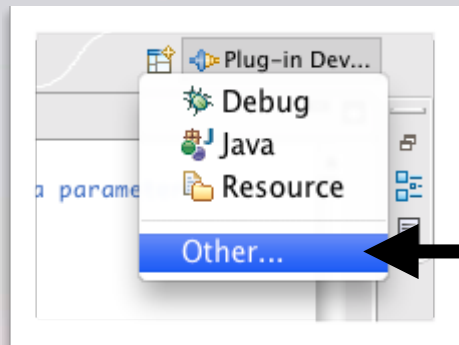
6 - Test

- Passer en perspective «Coloane Modeler»
- Créer un modèle
- Invoquer votre service par le menu «Coloane Services»

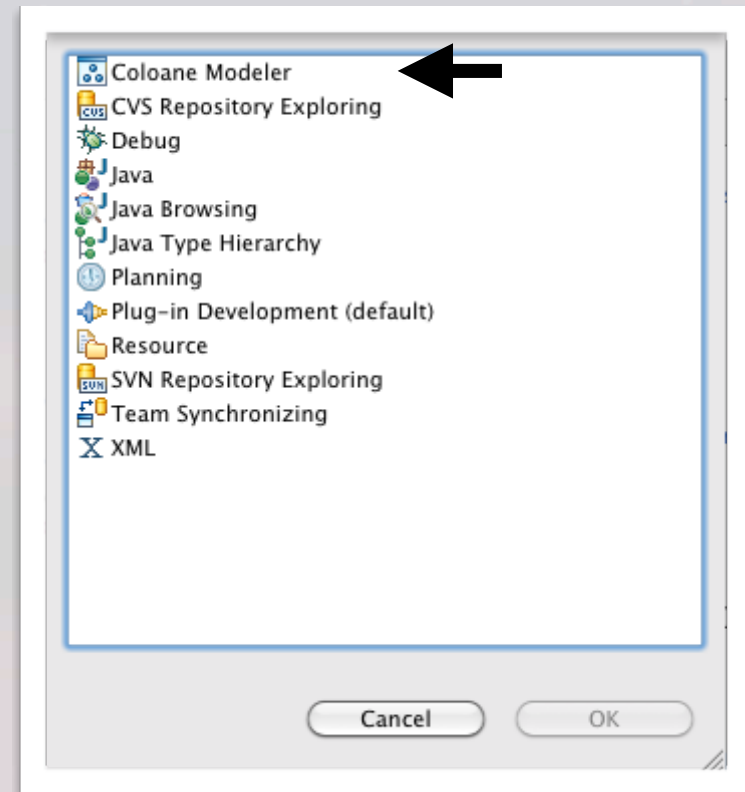


6 - Ouvrir Coloane

1)



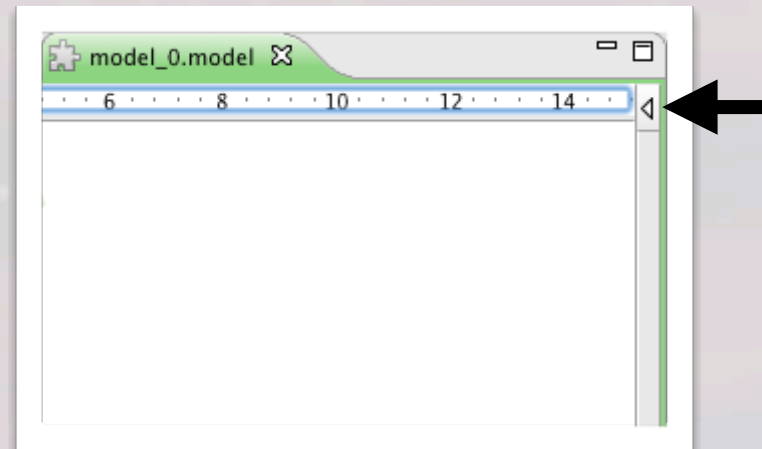
2)





6 - Créer un modèle

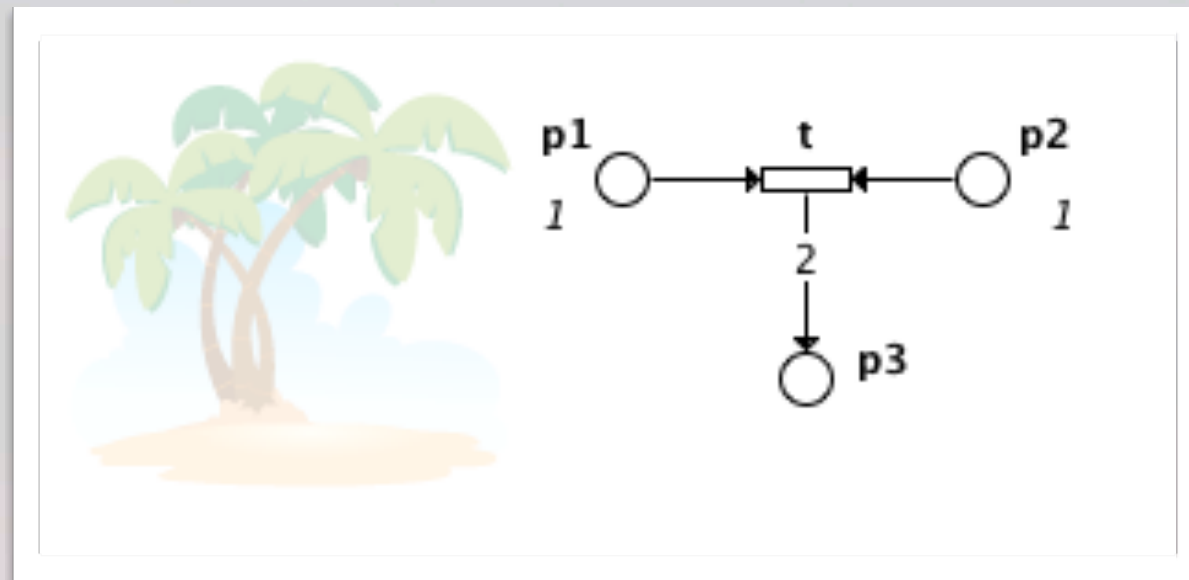
- Créer un nouveau «*Modeling Project*»
- Créer un nouveau «*Model*»
 - Utiliser le formalisme «*Place/Transition*»
- Ouvrir la palette :





6 - Créer un modèle

Exemple :





6 - Test

- Invoquer votre service par le menu «*Coloane Services*»
- Visualiser le résultat dans la vue «*Results*»
- Attention l'affichage n'est pas complet sous MacOS