Soutenance de thèse de Doctorat
le 26 Janvier 2010

Canevas de développement agile pour l'évolution fiable de systèmes logiciels à composants ou orientés services

Guillaume Waignier
Equipe ADAM – INRIA Lille-Nord Europe

Directeur : Laurence Duchien
Encadrant : Anne-Françoise Le Meur

UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE
LIFL - UMR 8022 CNRS
59655 VILLENEUVE D'ASCQ CEDEX
E-mail : Guillaume.Waignier@lifl.fr web : http://www.lifl.fr/~waignier

Plan

- Contexte
- Exemple : Dossier Médical Personnel
- Ma problématique : définir un canevas agile qui permet l'évolution fiable de logiciels à composants ou orientés services
- Ma contribution : CALICO
- Evaluation
- Conclusion et perspectives

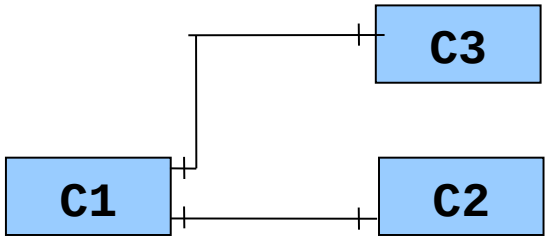
1. Contexte
1.1. Contexte de la thèse

- Construction et évolution des systèmes logiciels
  - Ensemble de fonctionnalités pour répondre aux besoins des utilisateurs
  - Allongement de la durée de vie
  - Besoin d'évoluer rapidement [Lehman85]
    - Nouvelles exigences des utilisateurs
    - Ajout/suppression de fonctionnalités
  - Besoin de fiabilité [ISO9126]
    - Maintien du niveau de service dans des conditions données et une durée de temps fixée
    - Conception & évolution

- Approches de constructions rapides
  - Architecture logicielle
  - Méthodologie de développement agile

1. Contexte
1.2 Architecture logicielle

- Architecture logicielle [Medvidovic00]
  - Brique logicielle = fonctionnalité
    - Composant / service
  - Facilite la conception
    - Assemblage, réutilisation
    - Raisonnement



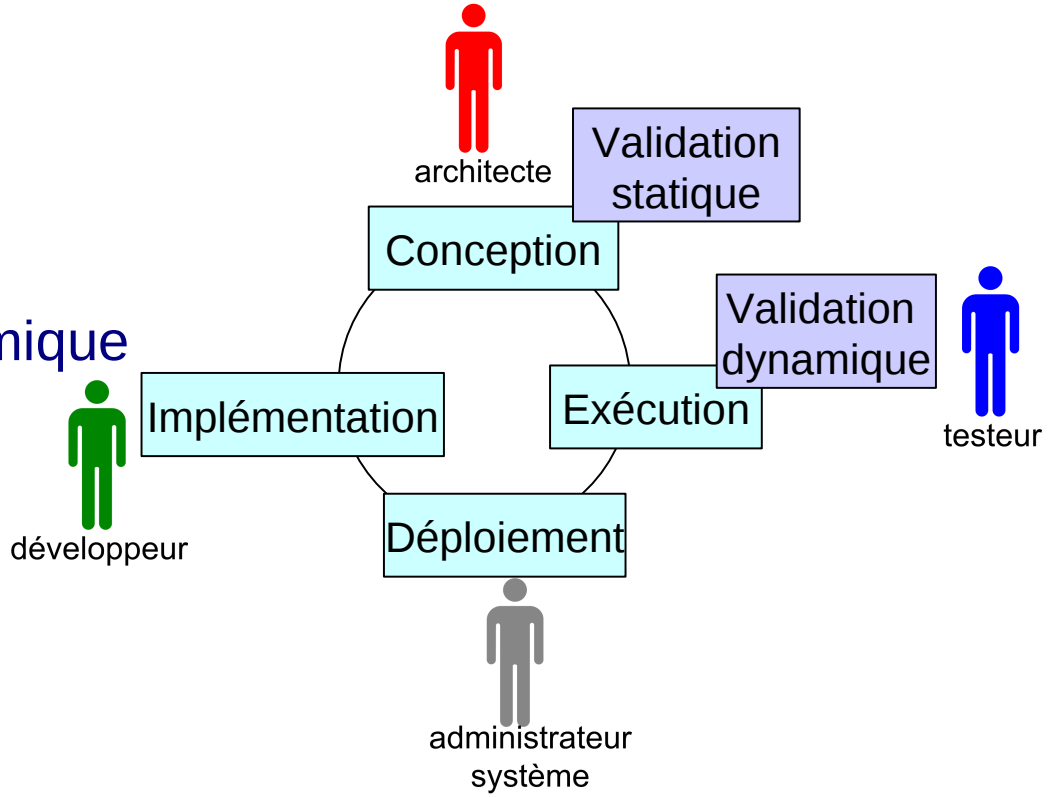
- Conception de systèmes fiables
  - Briques logicielles fiables
  - Assemblage de briques compatibles
    - Interactions compatibles
    - Problématique de conception et d'évolution

1. Contexte
1.3. Méthode agile

- Méthode agile [Manifesto01]
  - Processus de développement logiciel
  - Nombreuses méthodes agiles (Scrum, XP)
  - Réactif face aux évolutions des besoins des utilisateurs
  - Cycle court et rapide (Scrum : de 1 à 4 semaines)
  - Cycle incrémental et itératif
    - Ajout/suppression de fonctionnalités à chaque itération
  - Tests continus

1. Contexte
1.4. Cycle de développement agile

- Conception
- Validation statique
- Implémentation
- Déploiement
- Exécution / validation dynamique



Plan

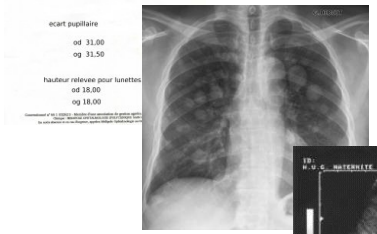
- Contexte
- Exemple : Dossier Médical Personnel
- Ma problématique : définir un canevas agile qui permet l'évolution fiable de logiciels à composants ou orientés services
- Ma contribution : CALICO
- Evaluation
- Conclusion et perspectives

2. Exemple : Dossier Médical Personnel

Financé par ANR

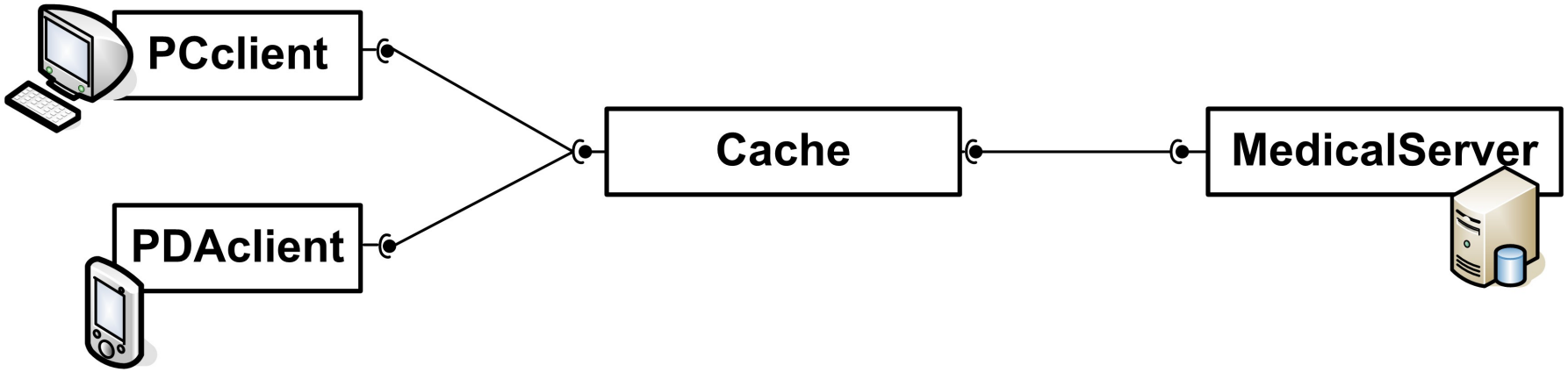
- Exemple fil rouge
  - Objectif : conception et évolution fiable
    - Cycle de développement agile
    - Architecture logicielle
  - Mise en évidence des problèmes
    - Chaque étape du cycle

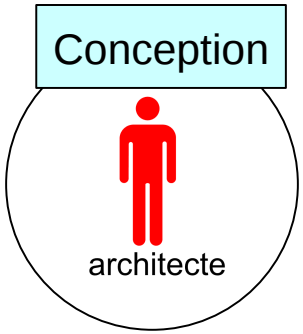
- Application DMP
  - Démonstrateur du projet ANR TLog FAROS
  - Un des rôles
    - Consultation d'informations médicales
  - Utilisateurs : professionnels de santé
  - Informations médicales



2. Exemple : Dossier Médical Personnel

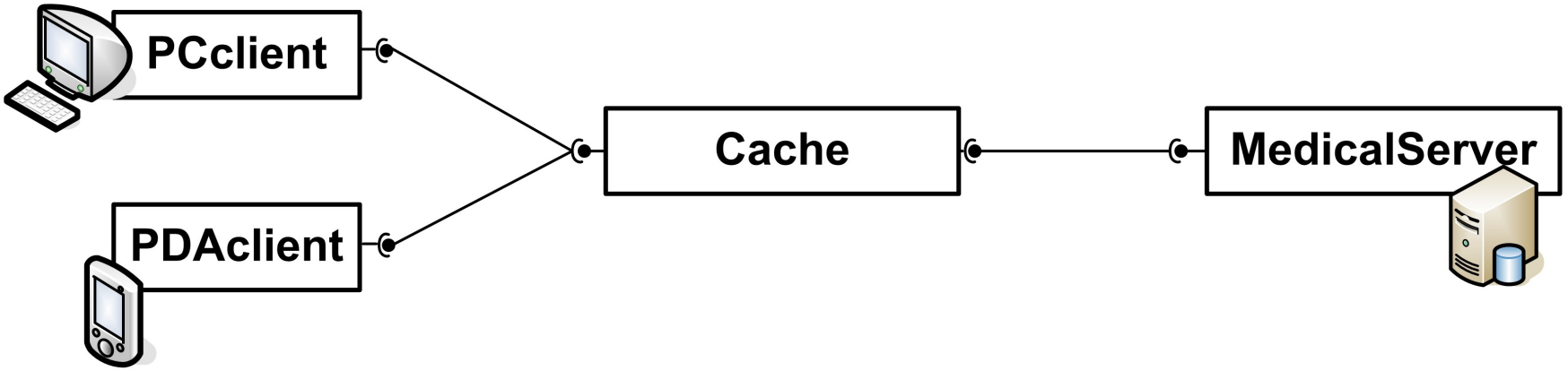
2.1. Conception (structure)





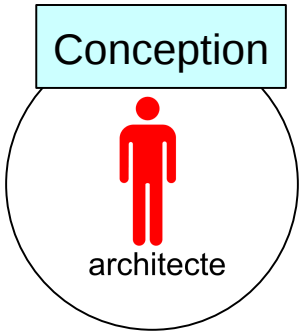
2. Exemple : Dossier Médical Personnel

2.1. Conception (propriétés applicatives)



hypothèse

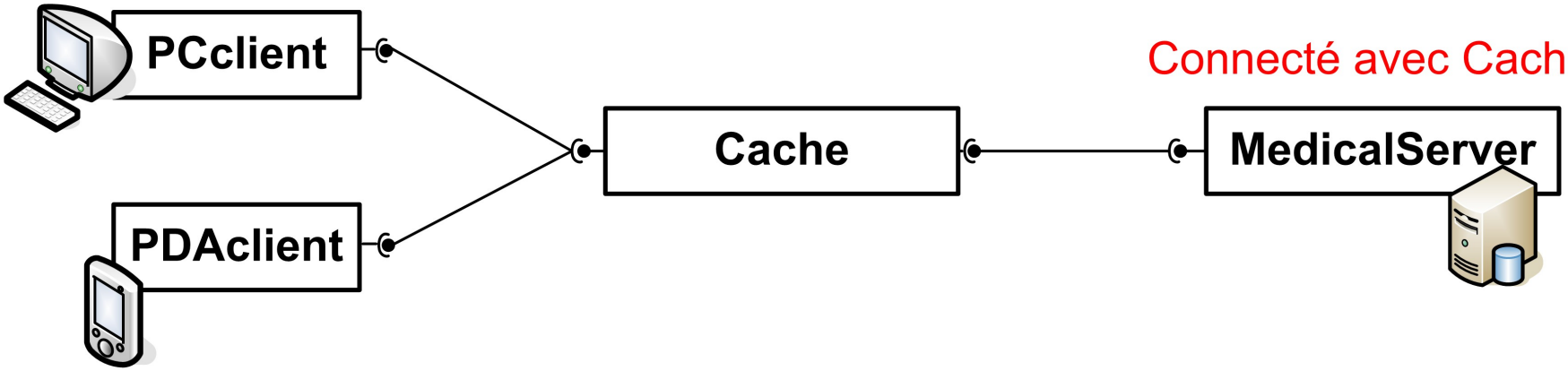
garantie



- Propriétés applicatives

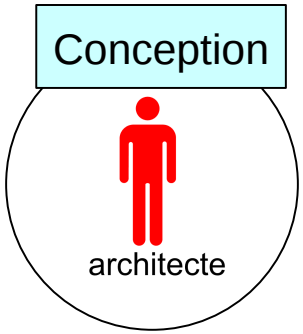
2. Exemple : Dossier Médical Personnel

2.1. Conception (propriétés applicatives)



hypothèse

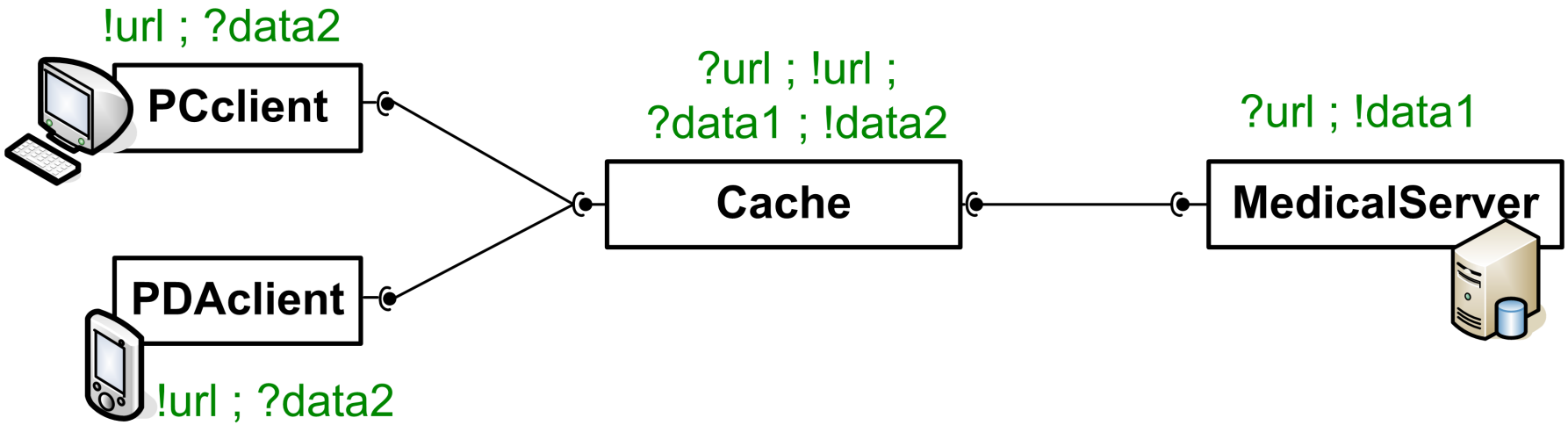
garantie



- Propriétés applicatives
  - Structure : contraint les connexions

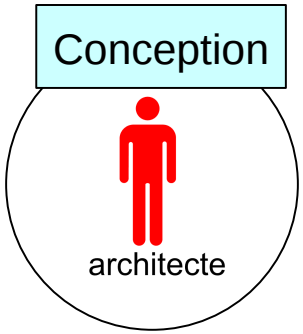
2. Exemple : Dossier Médical Personnel

2.1. Conception (propriétés applicatives)



hypothèse

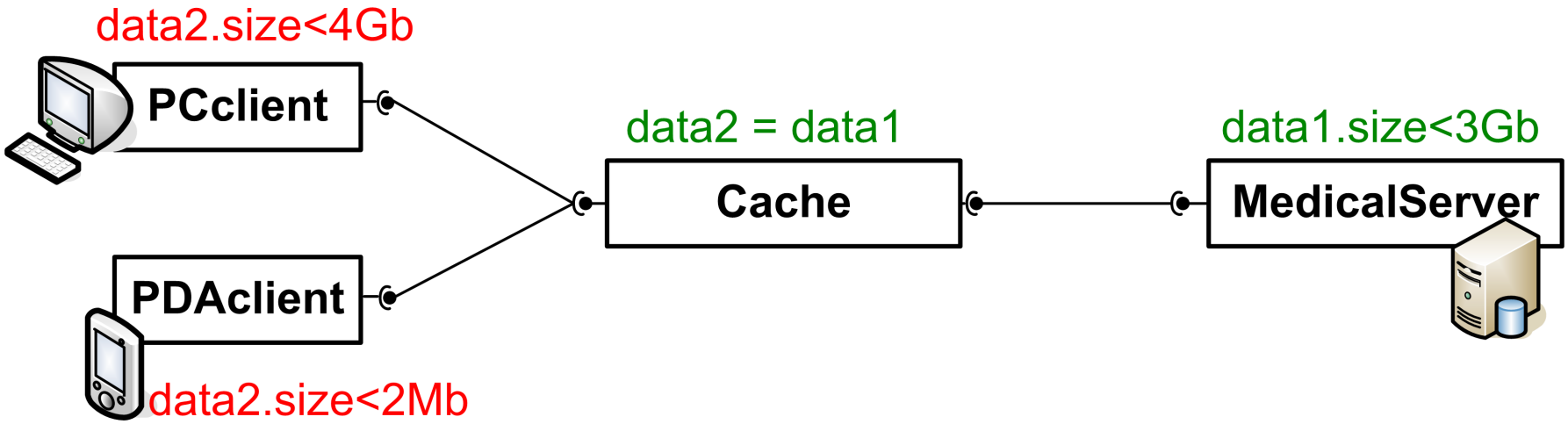
garantie



- Propriétés applicatives
  - Structure : contraint les connexions
  - Comportement : exprime le flot de contrôle

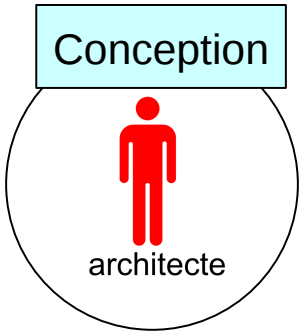
2. Exemple : Dossier Médical Personnel

2.1. Conception (propriétés applicatives)



hypothèse

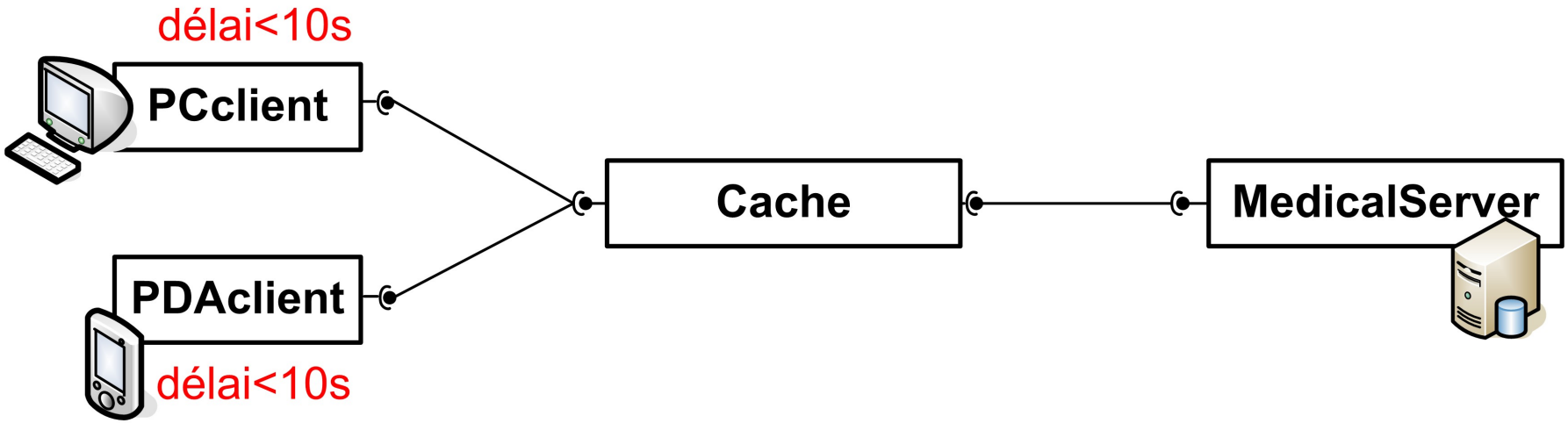
garantie



- Propriétés applicatives
  - Structure : contraint les connexions
  - Comportement : exprime le flot de contrôle
  - Flot de données : renseigne la valeur autorisée des messages reçus et envoyés

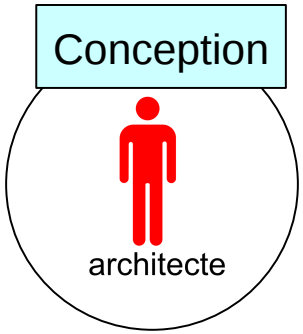
2. Exemple : Dossier Médical Personnel

2.1. Conception (propriétés applicatives)



hypothèse

garantie



- Propriétés applicatives
  - Structure : contraint les connexions
  - Comportement : exprime le flot de contrôle
  - Flot de données : renseigne la valeur autorisée des messages reçus et envoyés
  - QdS : définit des contraintes extra fonctionnelles (CPU)

2. Exemple : Dossier Médical Personnel

2.1. Conception (propriétés applicatives)

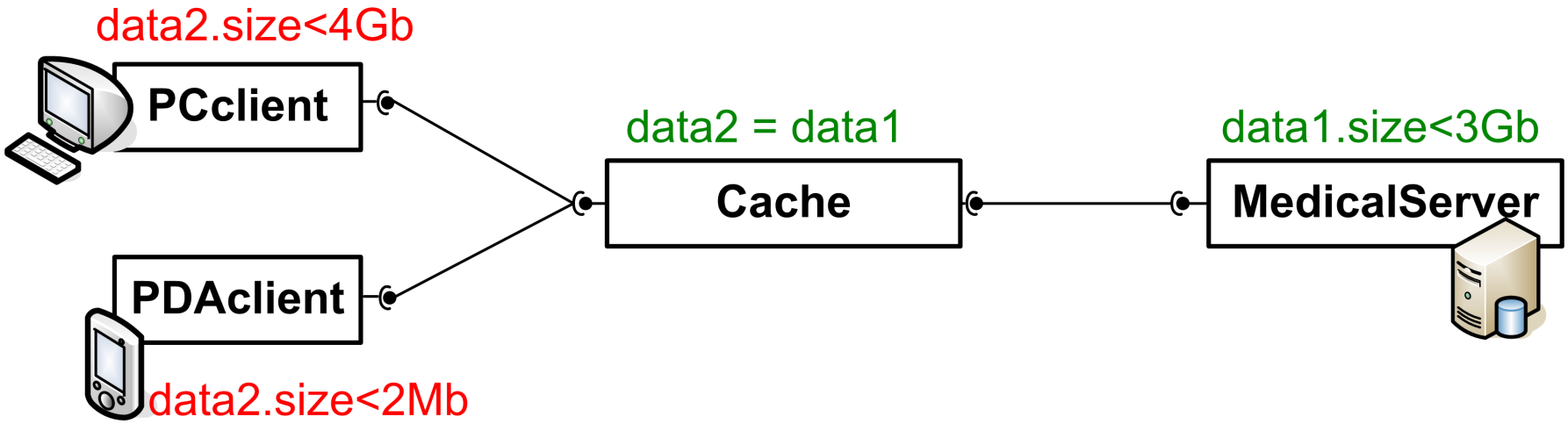
2.1.1. Étude de l'existant

- Support de spécification limité
  - Aucun support (Fractal Julia, CCM, SCA)
  - Pas de support des 4 catégories
    - Hétérogène et dispersé

	Fractal, CCM, SCA	Wright	SOFA, Fractal-BPC	Acme/ Armani	Confract	SafArchie	WebServices
Structure	limité	Style	limité	Armani language	invariant	Type	limité
Comportement	n/a	CSP	Behavioral protocol	n/a	n/a	SFSP	BPMN/BPEL
Flot de données	n/a	n/a	n/a	pre/post conditions	pre/post conditions	pre/post conditions	n/a
QdS	n/a	n/a	n/a	n/a	n/a	n/a	SLA

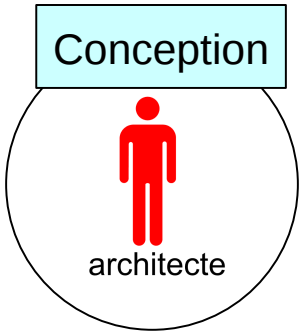
2. Exemple : Dossier Médical Personnel

2.1. Conception (propriétés applicatives)



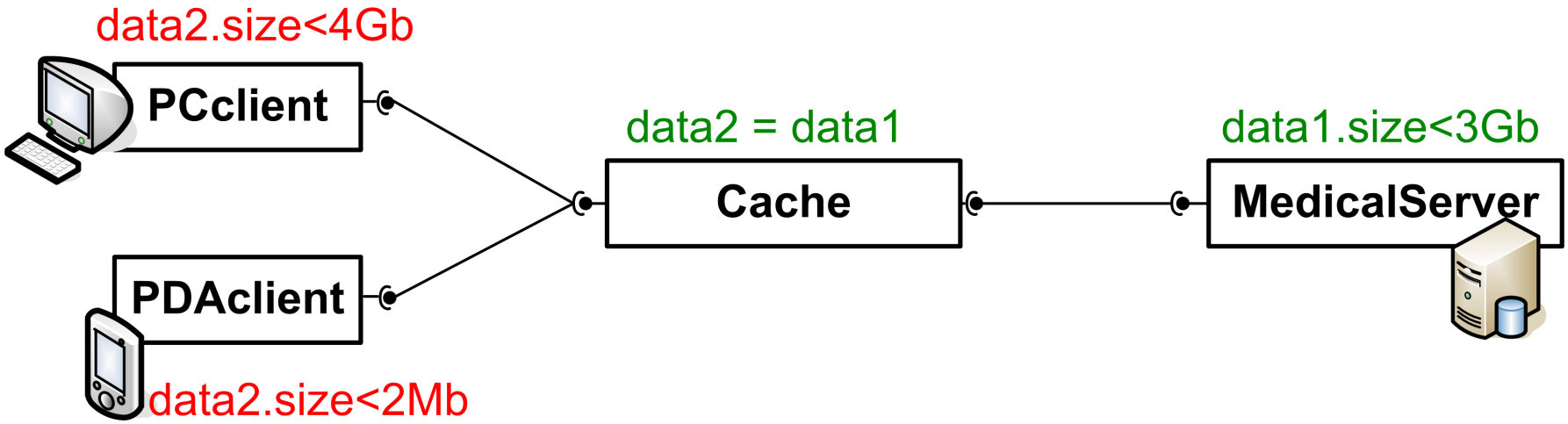
hypothèse

garantie



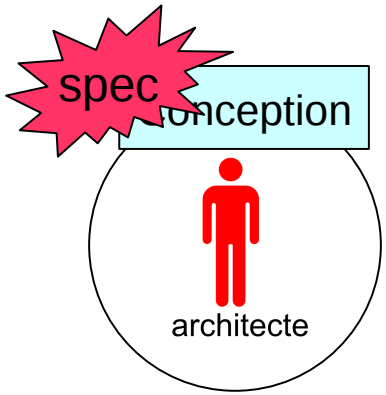
2. Exemple : Dossier Médical Personnel

2.1. Conception (propriétés applicatives)



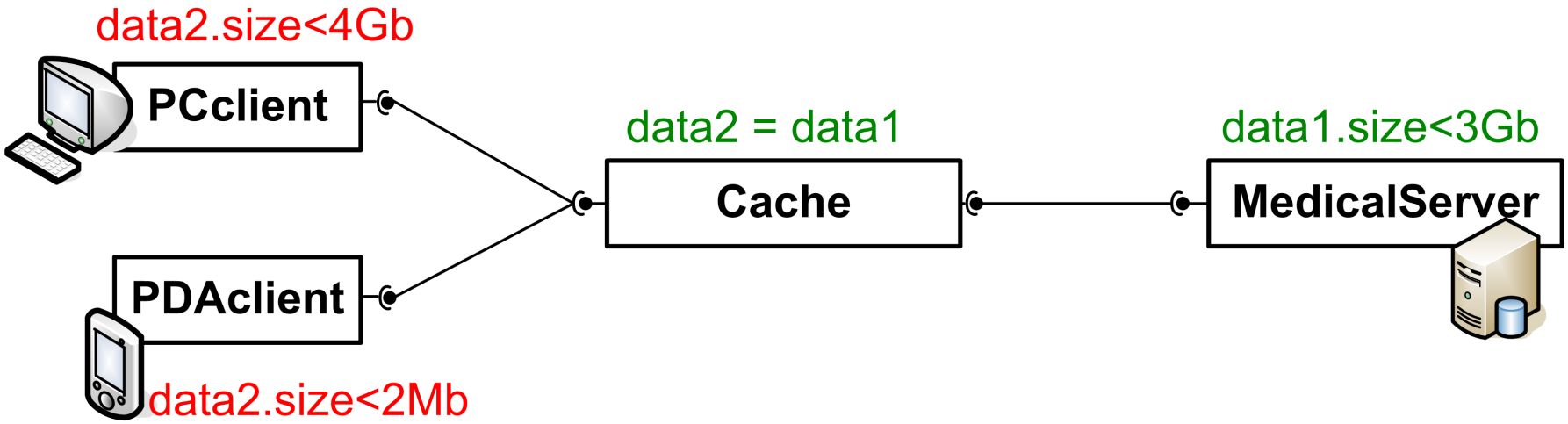
hypothèse

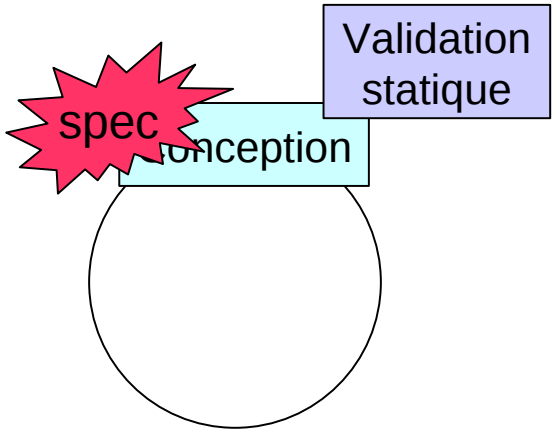
garantie



2. Exemple : Dossier Médical Personnel

2.2. Validation statique



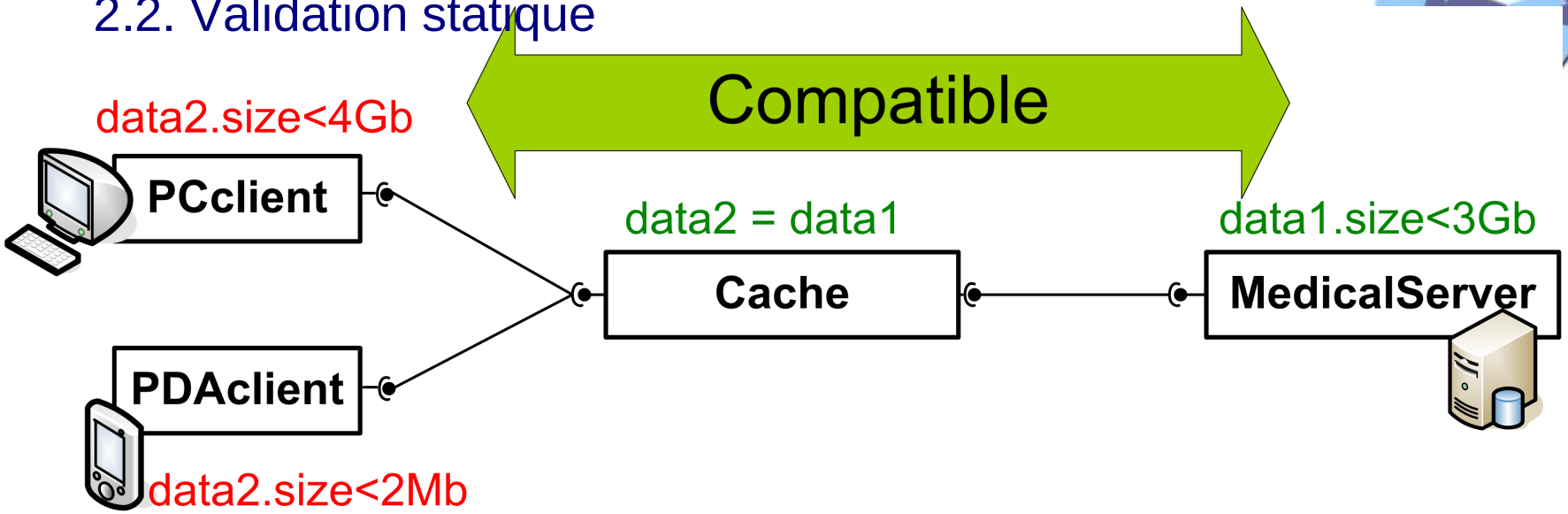


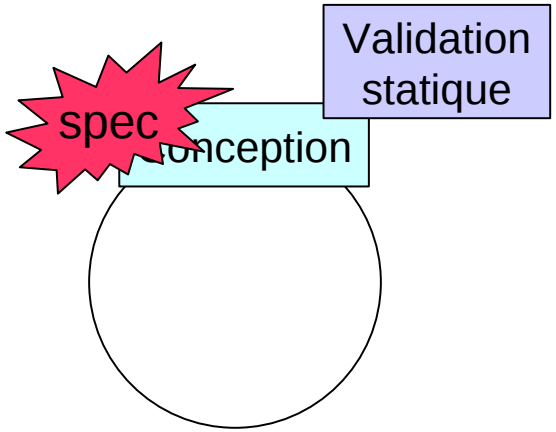
hypothèse

garantie

2. Exemple : Dossier Médical Personnel

2.2. Validation statique



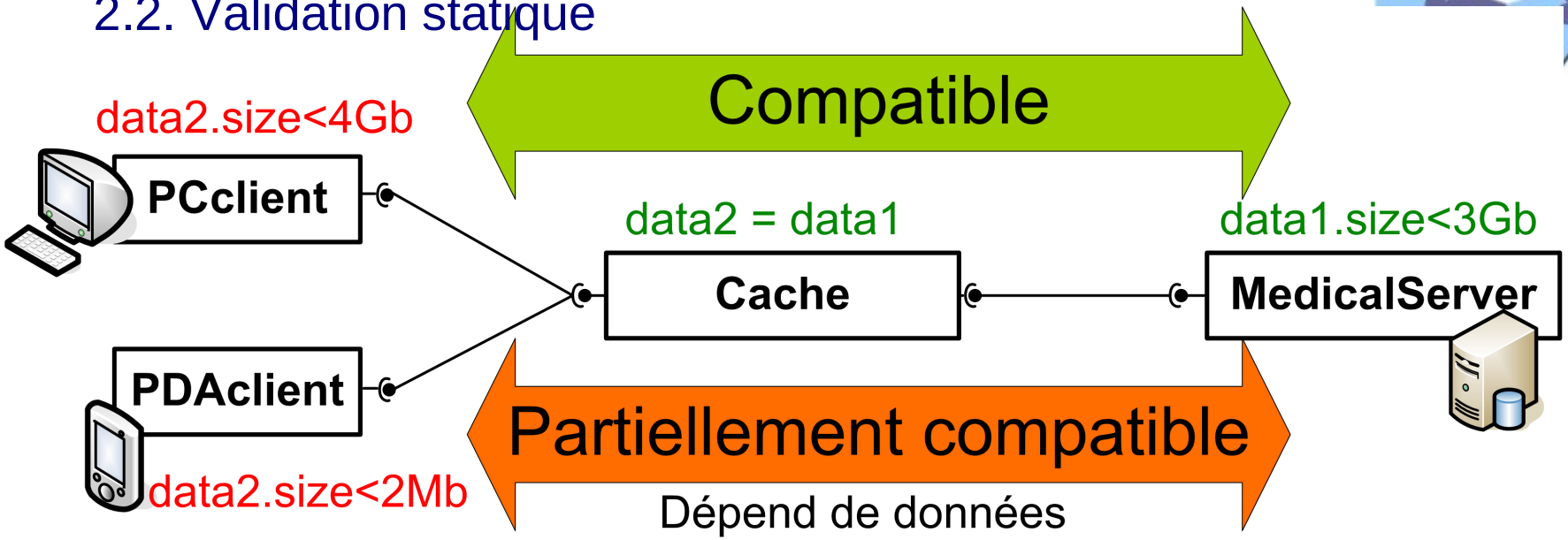


hypothèse

garantie

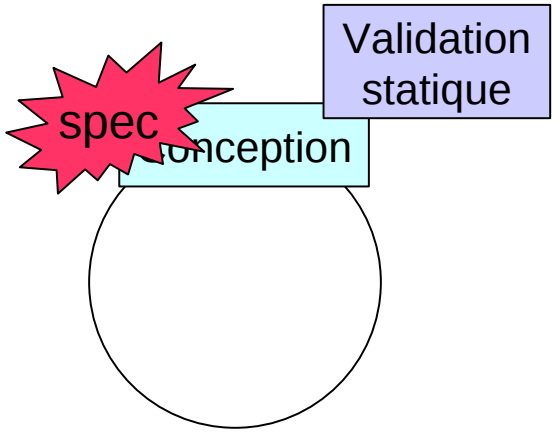
2. Exemple : Dossier Médical Personnel

2.2. Validation statique



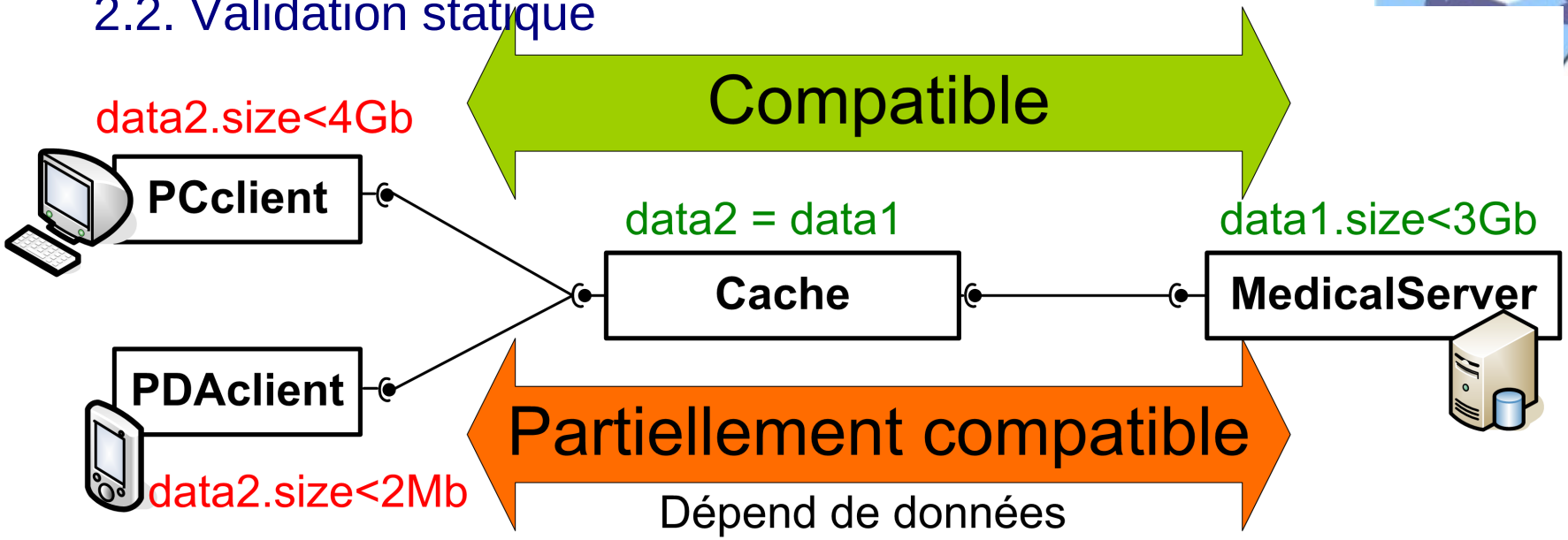
hypothèse

garantie



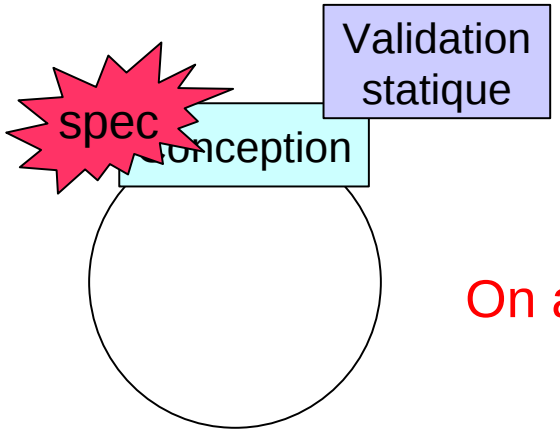
2. Exemple : Dossier Médical Personnel

2.2. Validation statique



hypothèse

garantie



On aura besoin d'effectuer une validation dynamique

2. Exemple : Dossier Médical Personnel

2.2. Validation statique

2.2.1. Étude de l'existant

- Analyse d'une interaction [Barais05]
  - Compatible
  - Incompatible
  - Partiellement compatible

2. Exemple : Dossier Médical Personnel

2.2. Validation statique

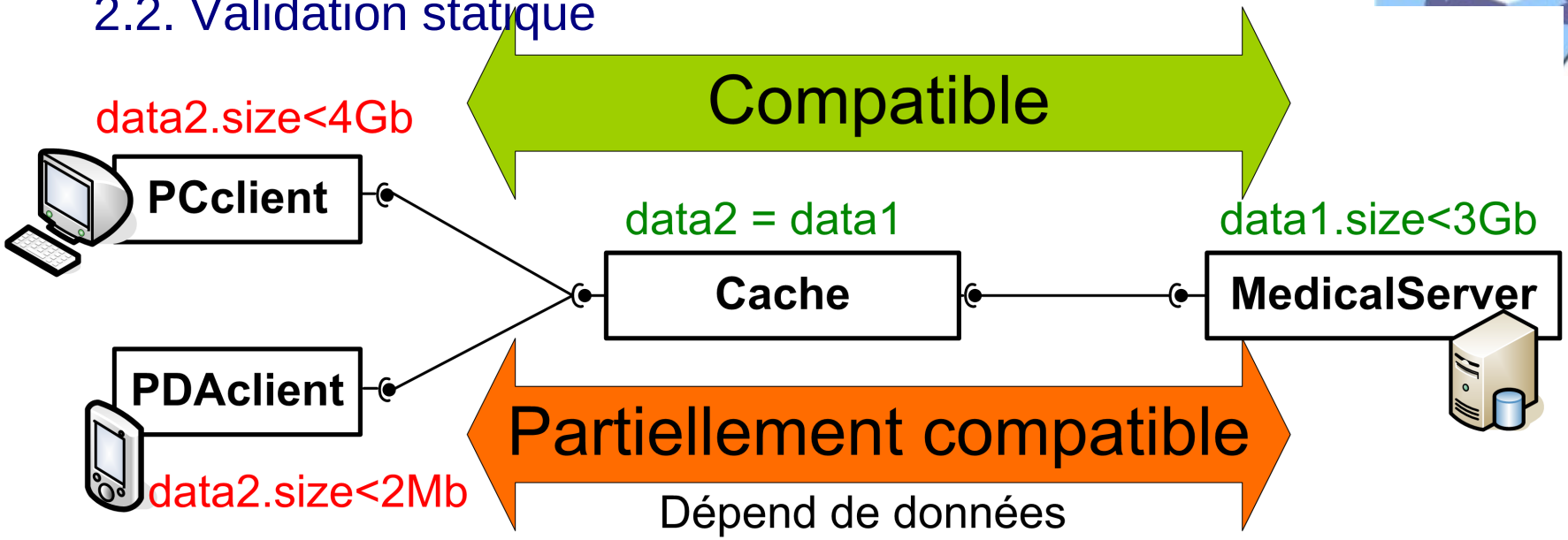
2.2.1. Étude de l'existant

- Analyse d'une interaction [Barais05]
  - Compatible
  - Incompatible
  - Partiellement compatible

- Manque d'intégration d'outil d'analyse statique
  - Outils fortement couplés avec une plate-forme d'exécution
  - Prise en charge d'un sous ensemble des propriétés applicatives
  - Pas de prise en compte des interactions partiellement compatibles
    - Considère comme incompatible (Wright)
    - Affichage d'un avertissement (SafArchie)
      - Analyses dynamiques découplées

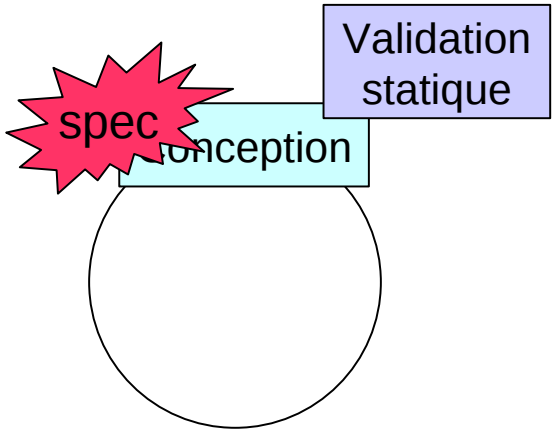
2. Exemple : Dossier Médical Personnel

2.2. Validation statique



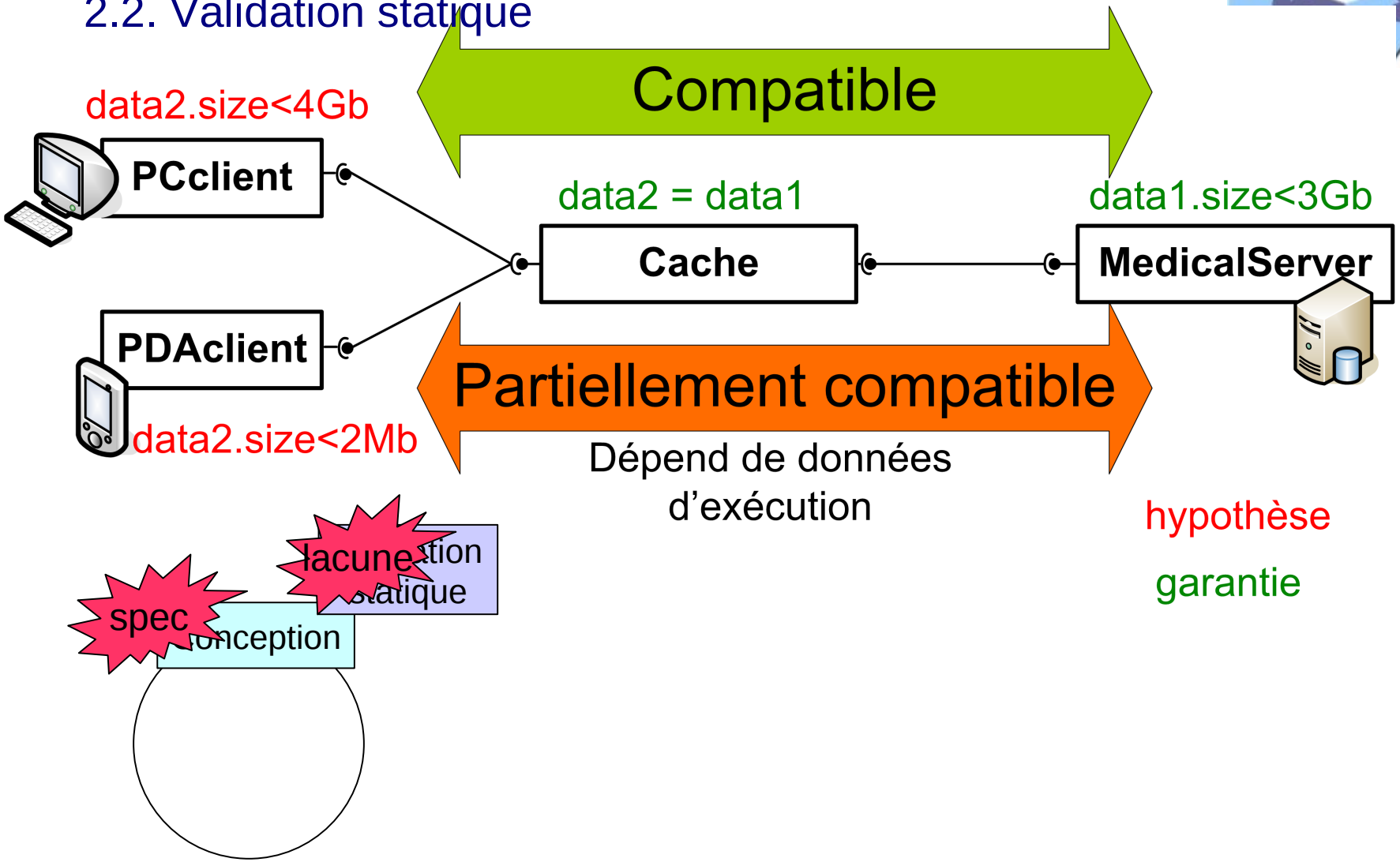
hypothèse

garantie



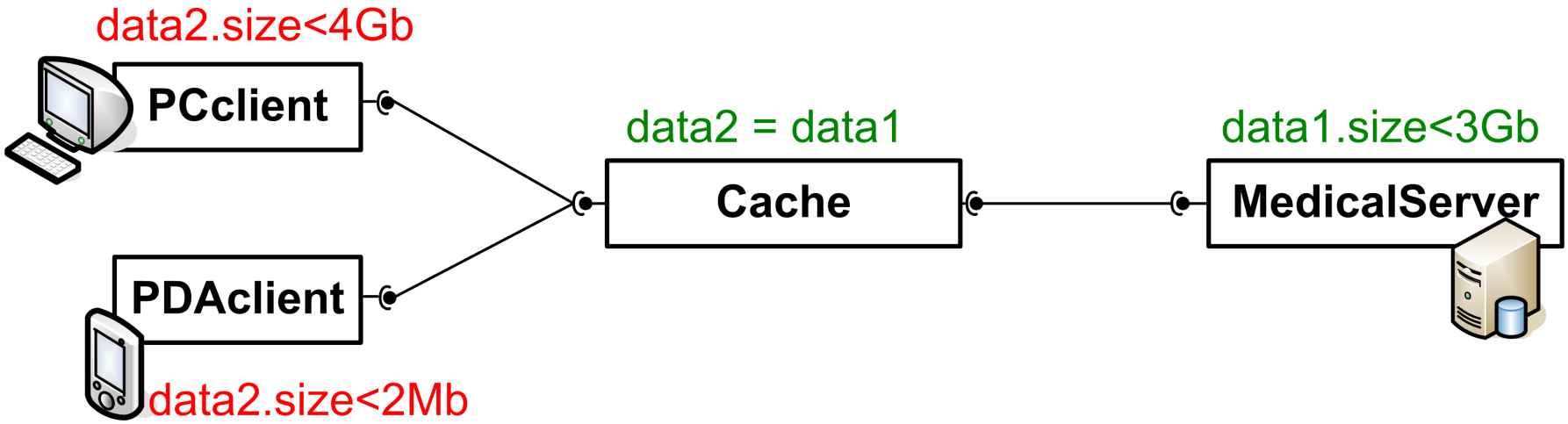
2. Exemple : Dossier Médical Personnel

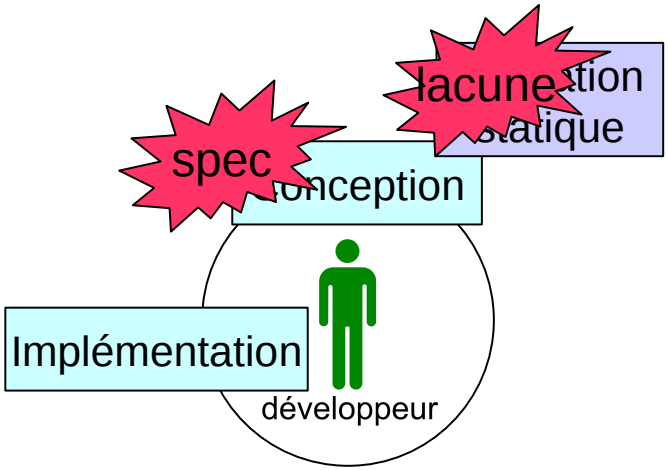
2.2. Validation statique



2. Exemple : Dossier Médical Personnel

2.3. Implémentation





hypothèse

garantie

2. Exemple : Dossier Médical Personnel

2.3. Implémentation

data2.size<4Gb

1.size<3Gb

dicalServer

ypothèse

garantie

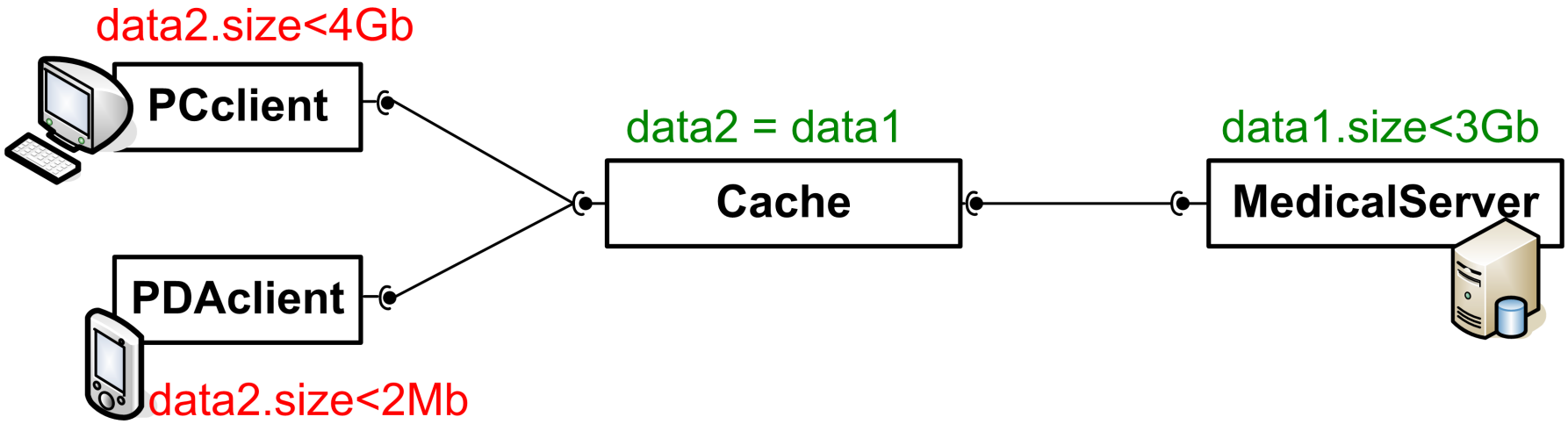
```
@Component(name="Cache", provides=@Interface(...))
public class CacheImpl implements SearchDataItf
{
  @Requires(name="out")
  private ImageServer.SearchDataItf p2;

  public Image get(URL url)
  {
    ...
  }
}
```

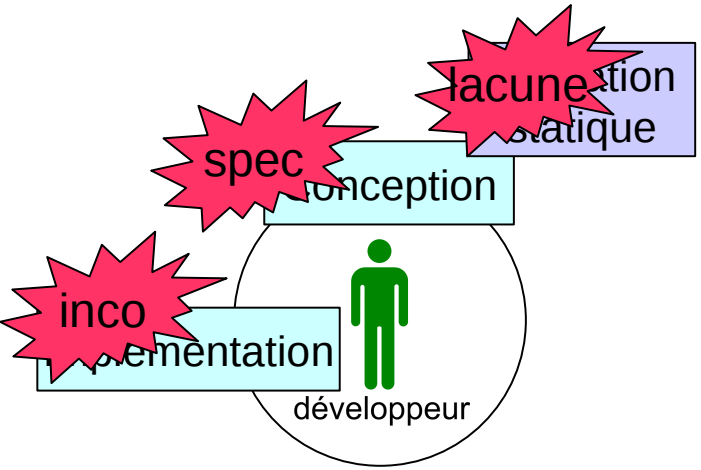
Implémentation

développeur

2. Exemple : Dossier Médical Personnel

2.3. Implémentation



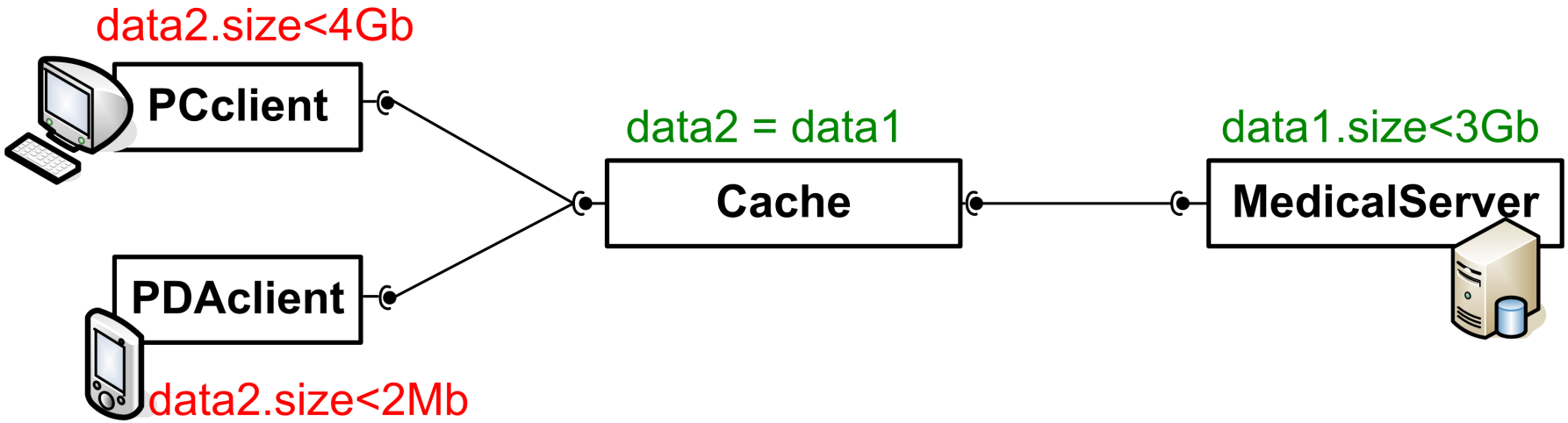


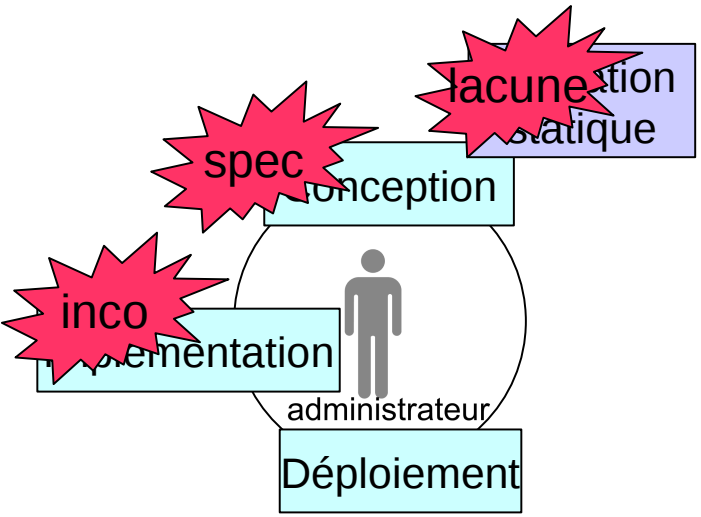
hypothèse

garantie

2. Exemple : Dossier Médical Personnel

2.4. Déploiement



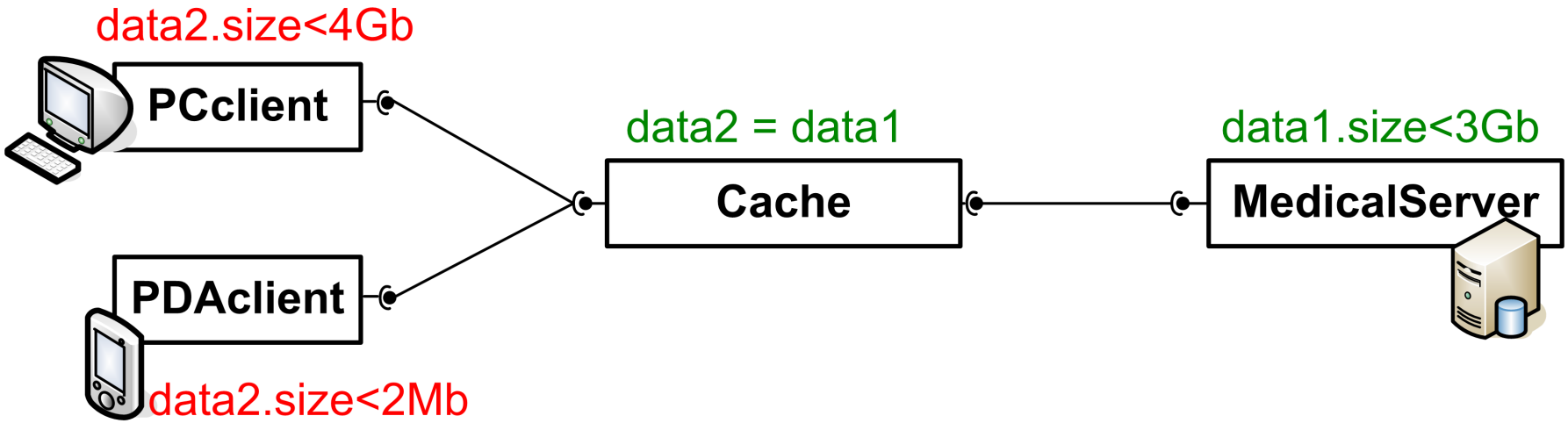


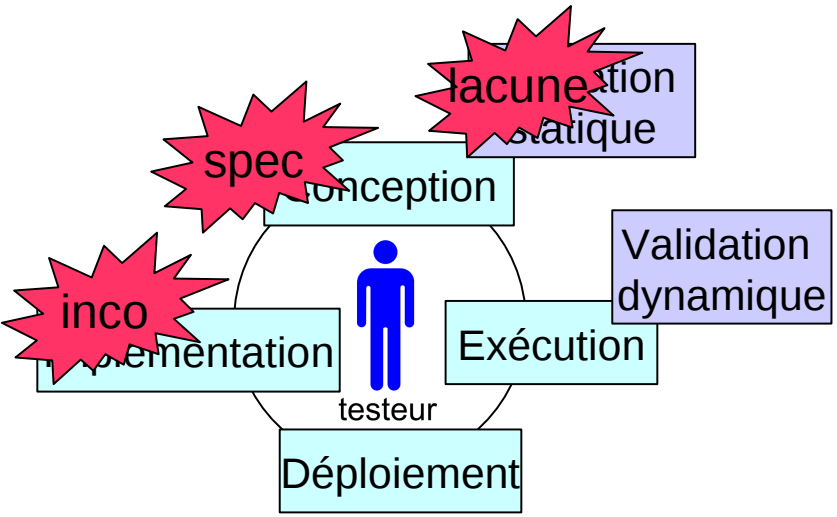
hypothèse

garantie

2. Exemple : Dossier Médical Personnel

2.5. Exécution / Validation dynamique



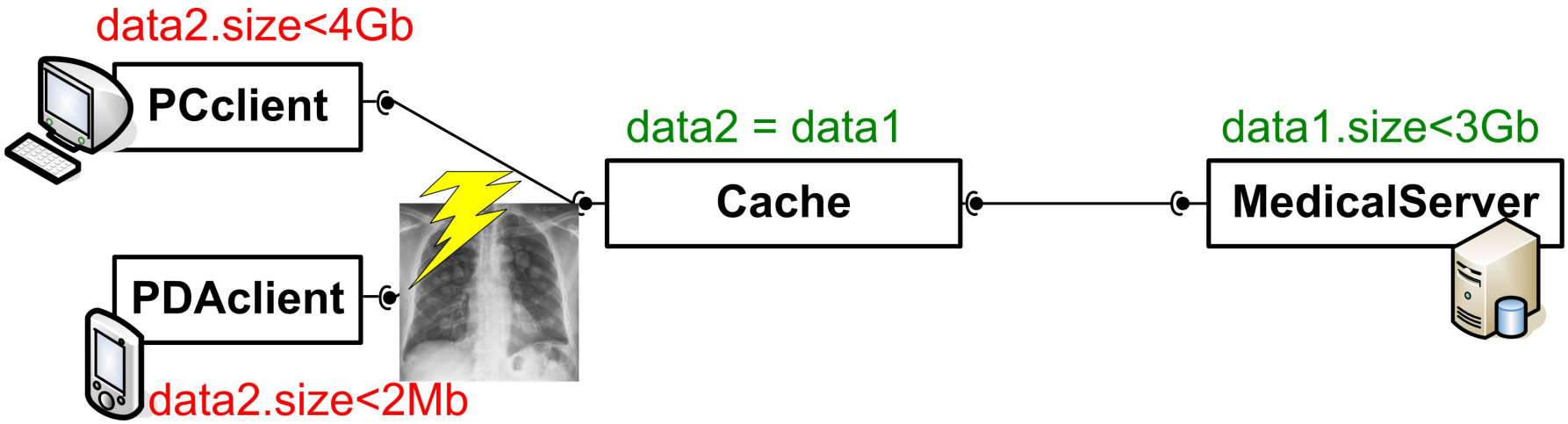


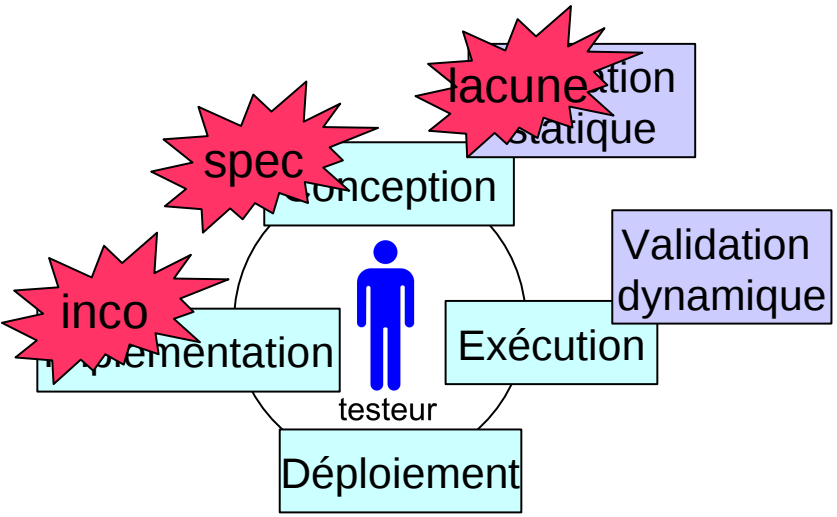
hypothèse

garantie

2. Exemple : Dossier Médical Personnel

2.5. Exécution / Validation dynamique





hypothèse

garantie

2. Exemple : Dossier Médical Personnel

2.5. Exécution / Validation dynamique

2.5.1. Étude de l'existant

- Besoin d'observer les données d'exécution
  - Valeur des messages échangés
  - QdS

2. Exemple : Dossier Médical Personnel

2.5. Exécution / Validation dynamique

2.5.1. Étude de l'existant

- Besoin d'observer les données d'exécution
  - Valeurs des messages échangés
  - QdS

- Manque de support d'observation
  - Mise en oeuvre manuelle du mécanisme d'observation
    - Modification du code source des composants/services
    - Intégration de sondes de QdS
  - Tâche difficile
  - Source d'erreur

2. Exemple : Dossier Médical Personnel

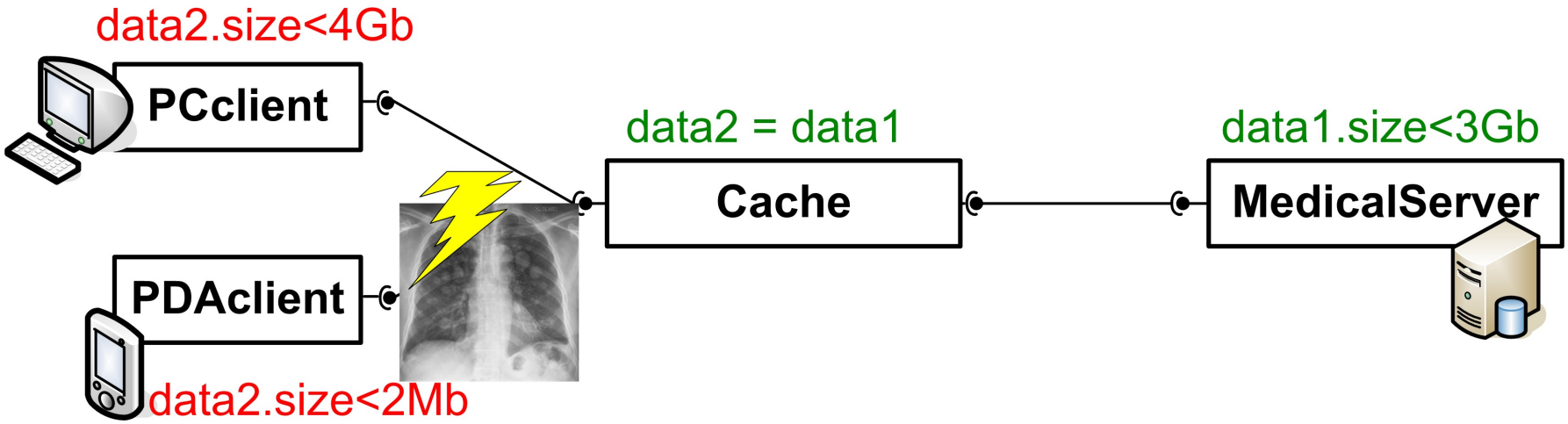
2.5. Exécution / Validation dynamique

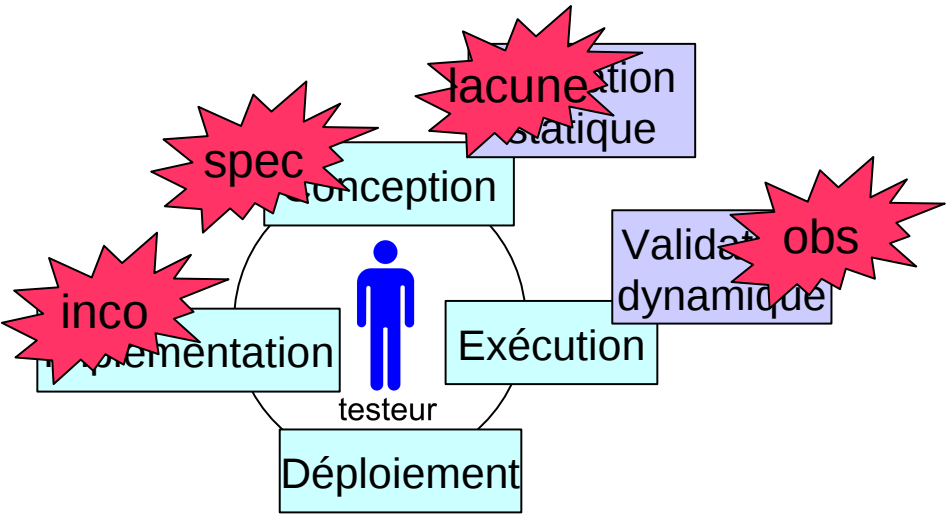
2.5.1. Étude de l'existant

- Besoin d'observer les données d'exécution
  - Valeur des messages échangés
  - QdS
- Manque de support d'observation
  - Mise en oeuvre manuelle du mécanisme d'observation
    - Modification du code source des composants/services
    - Intégration de sondes de QdS
  - Tâche difficile
  - Source d'erreur
- Identification manuelle de la source de l'erreur
  - Message d'erreur au niveau du code
  - Perte du lien avec les propriétes applicatives

2. Exemple : Dossier Médical Personnel

2.5. Exécution / Validation dynamique



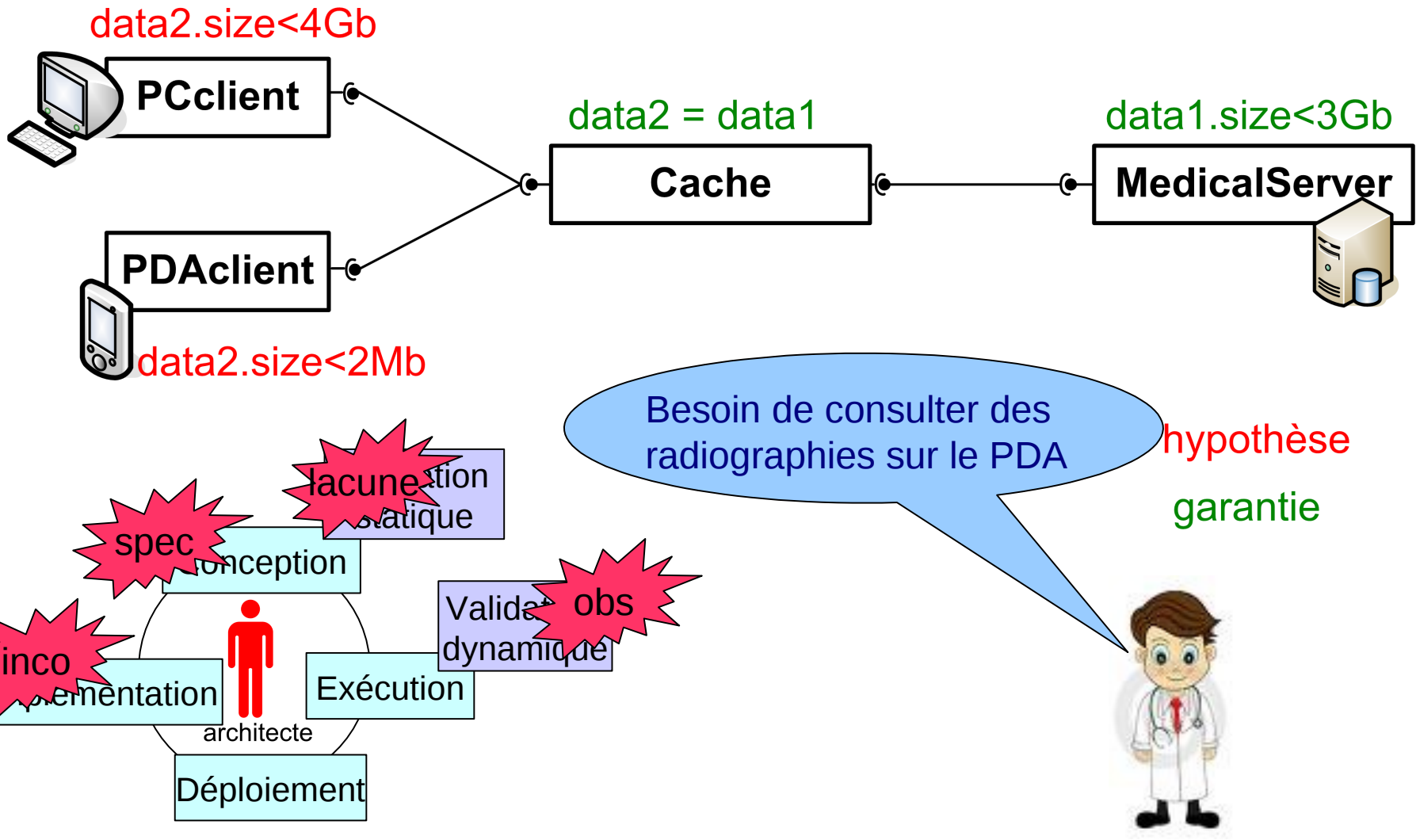


hypothèse

garantie

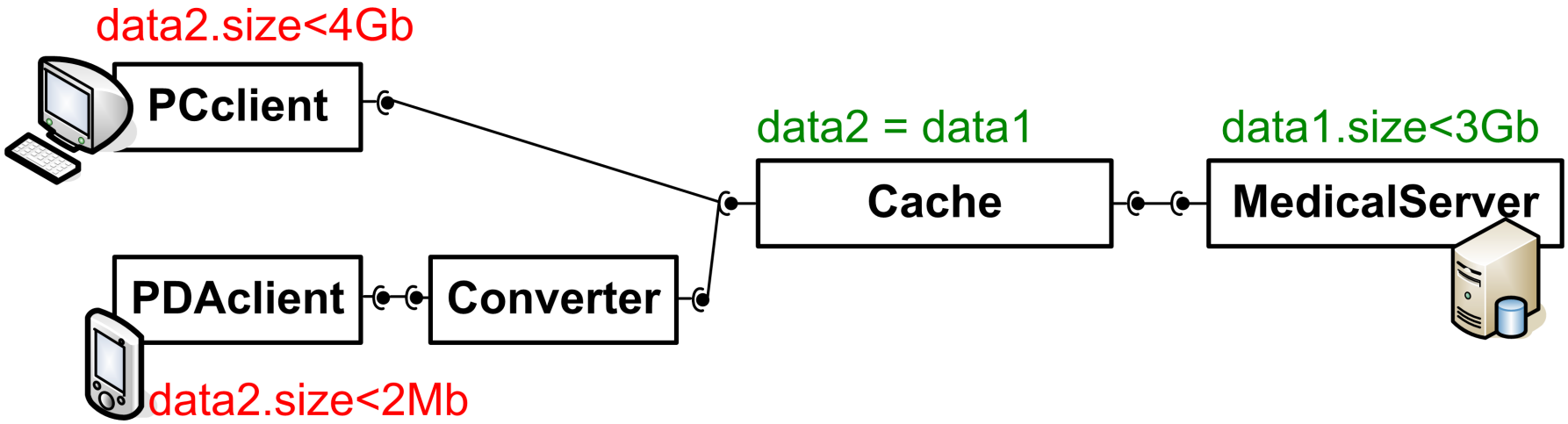
2. Exemple : Dossier Médical Personnel

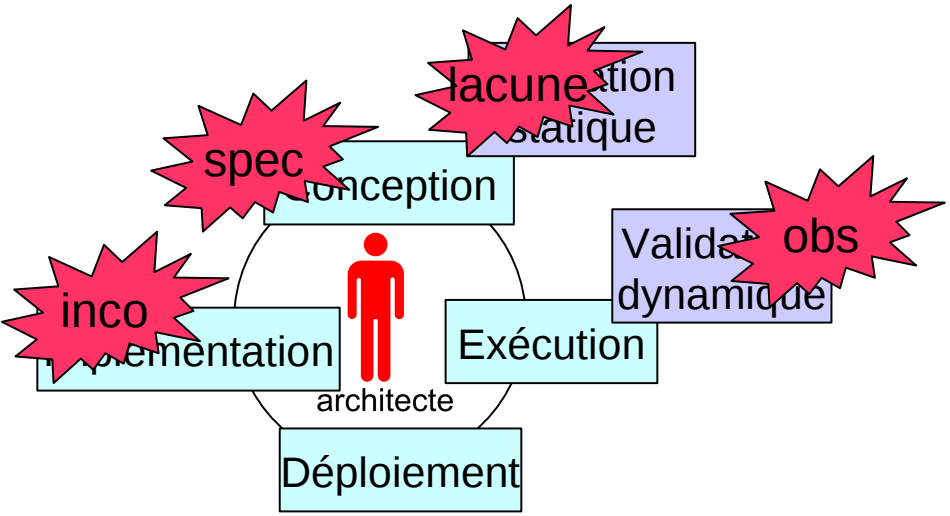
2.6. Évolution



2. Exemple : Dossier Médical Personnel

2.6. Évolution



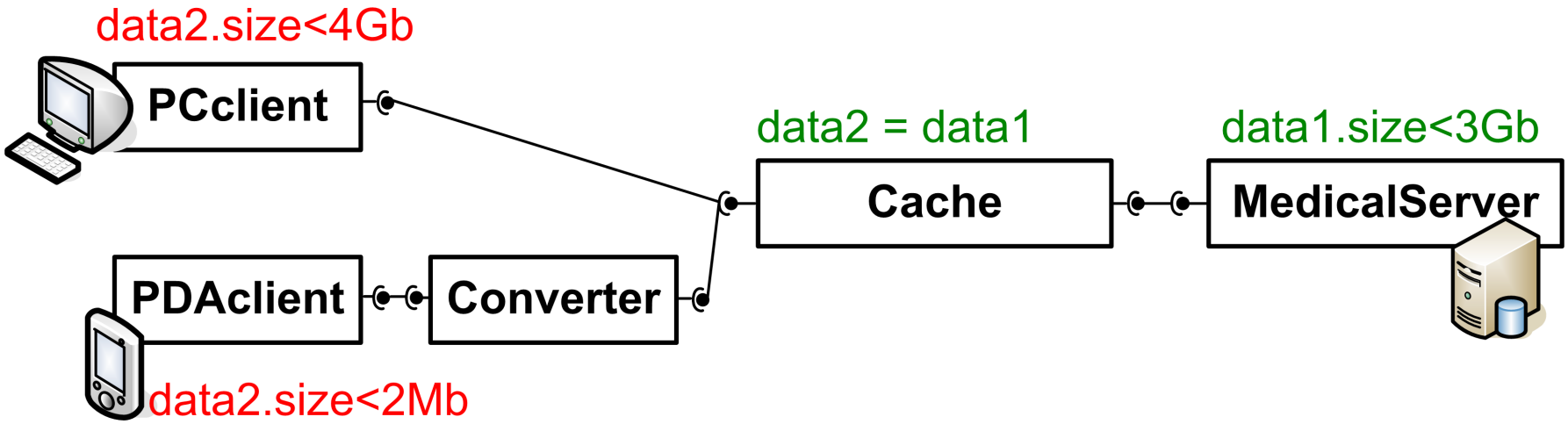


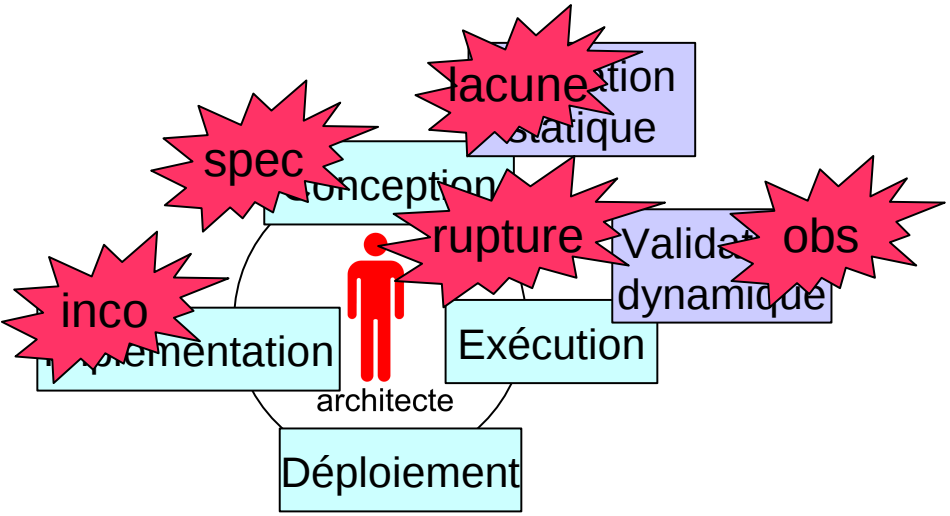
hypothèse

garantie

2. Exemple : Dossier Médical Personnel

2.6. Évolution





hypothèse

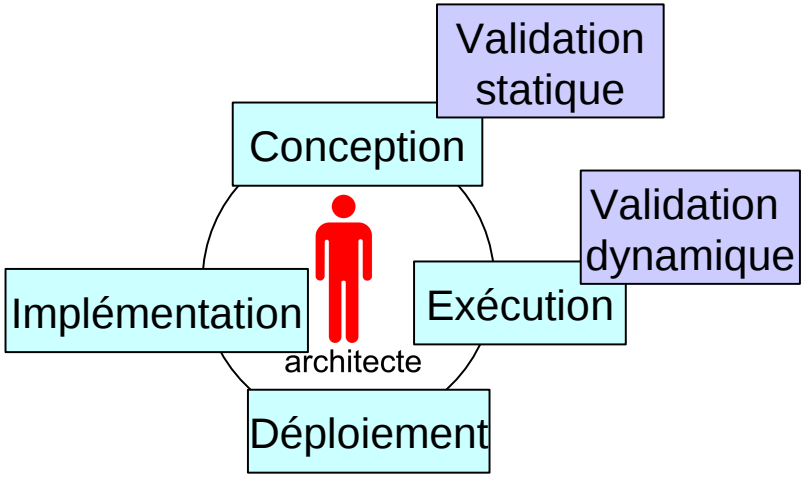
garantie

Plan

- Contexte
- Exemple : Dossier Médical Personnel
- Ma problématique : définir un canevas agile qui permet l'évolution fiable de logiciels à composants ou orientés services
- Ma contribution : CALICO
- Evaluation
- Conclusion et perspectives

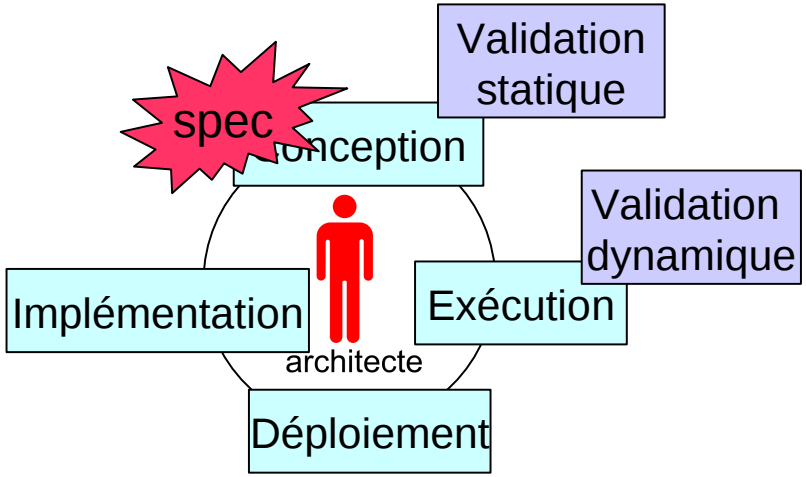
3. Ma problématique : évolution fiable

- Conception



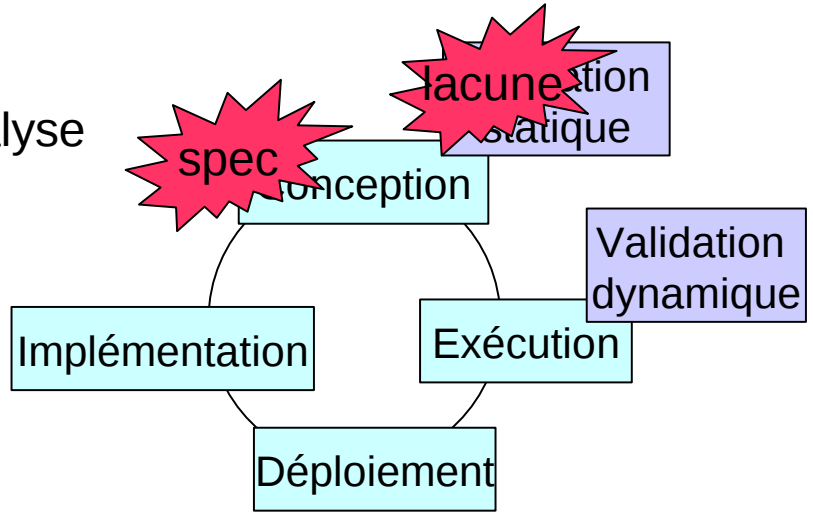
3. Ma problématique : évolution fiable

- Conception
  - Manque de moyen de spécification des propriétés applicatives



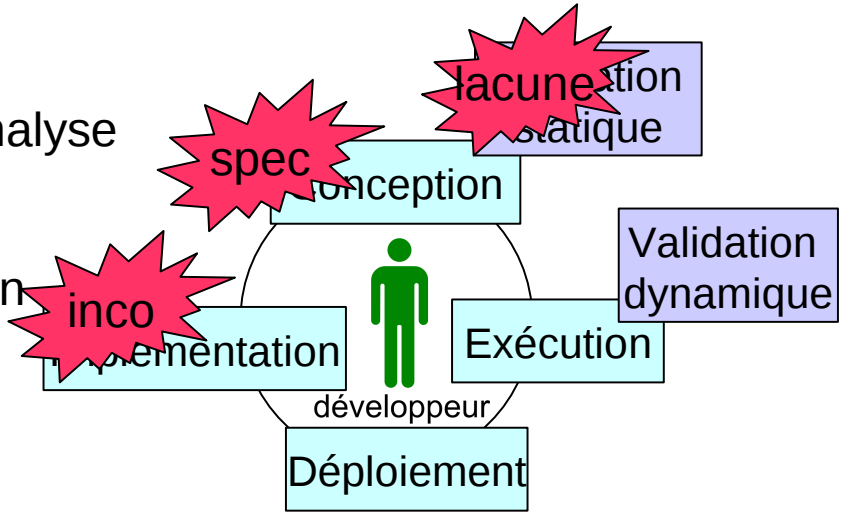
3. Ma problématique : évolution fiable

- Conception
  - Manque de moyen de spécification des propriétés applicatives
- Validation statique
  - Manque d'intégration d'outil d'analyse



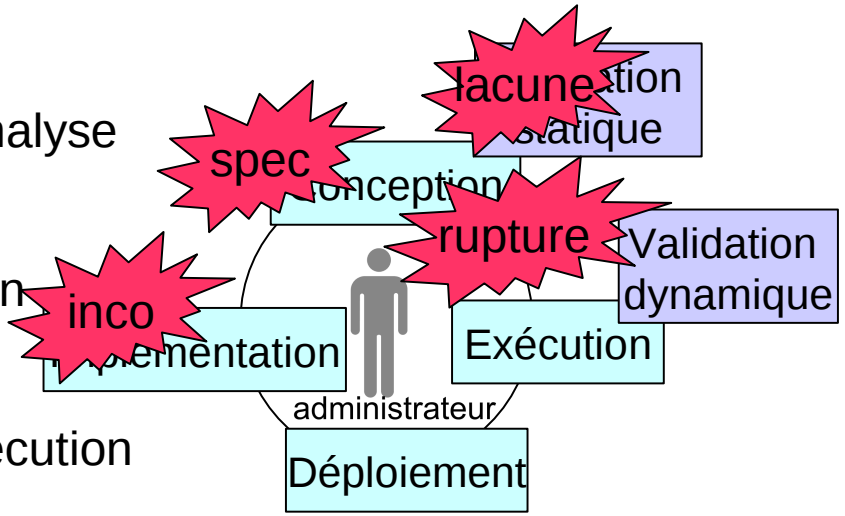
3. Ma problématique : évolution fiable

- Conception
  - Manque de moyen de spécification des propriétés applicatives
- Validation statique
  - Manque d'intégration d'outil d'analyse
- Implémentation
  - Incohérence avec la spécification



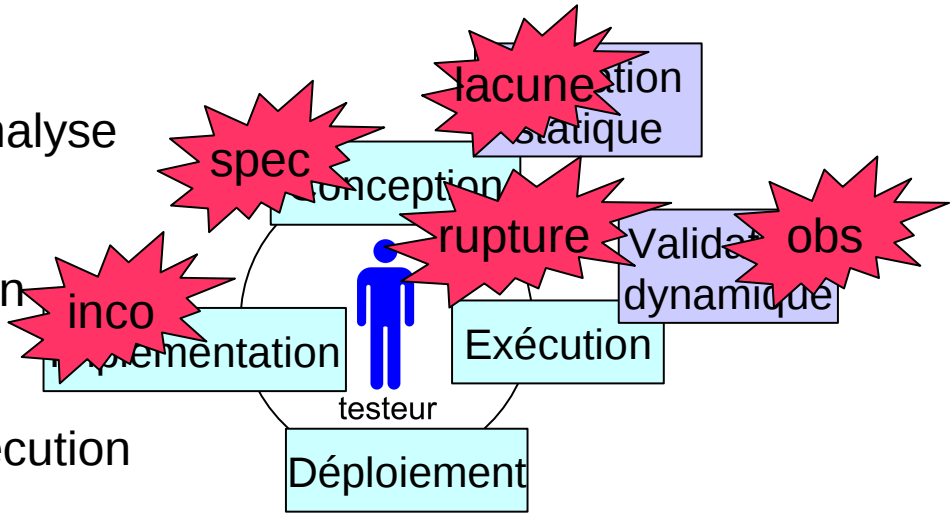
3. Ma problématique : évolution fiable

- Conception
  - Manque de moyen de spécification des propriétés applicatives
- Validation statique
  - Manque d'intégration d'outil d'analyse
- Implémentation
  - Incohérence avec la spécification
- Déploiement
  - Rupture entre conception ↔ exécution



3. Ma problématique : évolution fiable

- Conception
  - Manque de moyen de spécification des propriétés applicatives
- Validation statique
  - Manque d'intégration d'outil d'analyse
- Implémentation
  - Incohérence avec la spécification
- Déploiement
  - Rupture entre conception ↔ exécution
- Exécution / validation dynamique
  - Manque de mécanismes d'observation

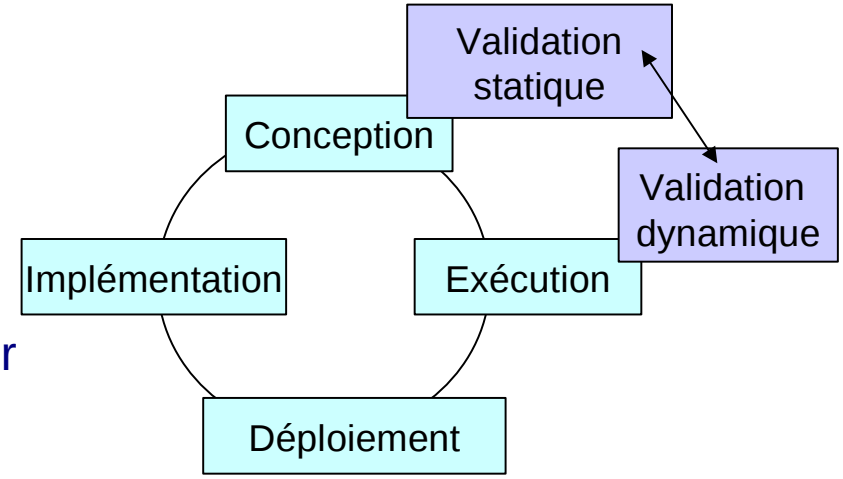


Plan

- Contexte
- Exemple : Dossier Médical Personnel
- Ma problématique : définir un canevas agile qui permet l'évolution fiable de logiciels à composants ou orientés services
- Ma contribution : CALICO
- Evaluation
- Conclusion et perspectives

4. Cycle de développement de CALICO

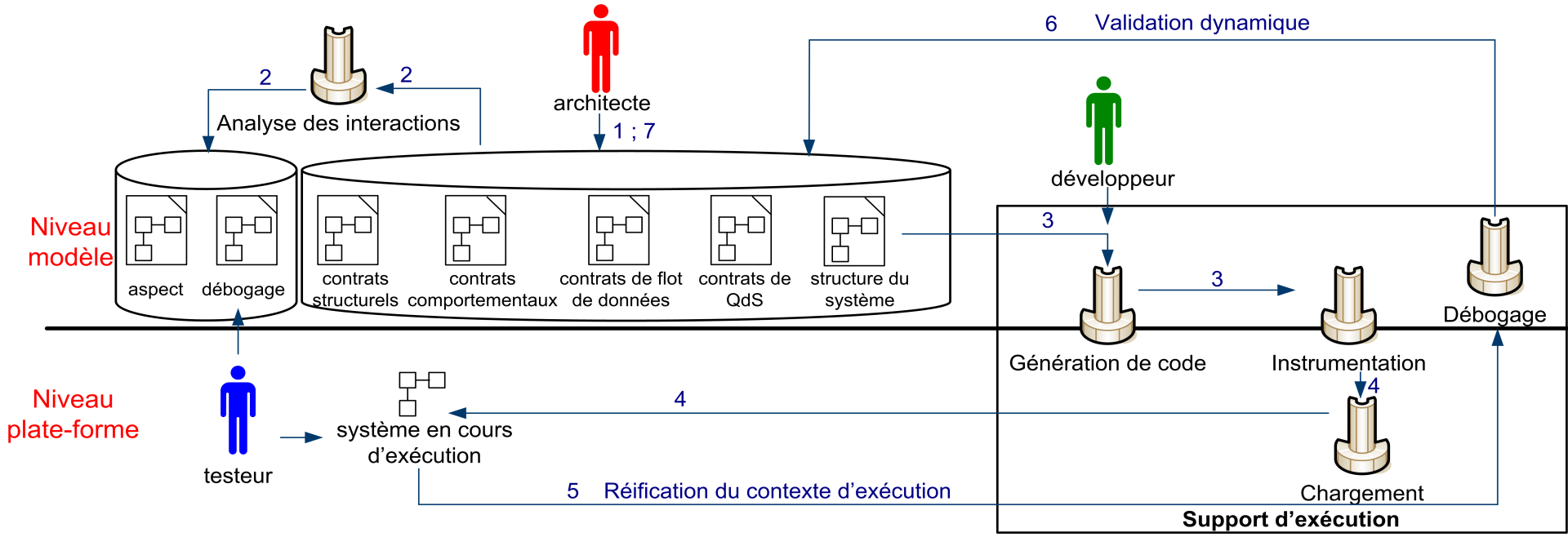
- Développement agile
  - Cyle itératif et incrémental
    - 6 étapes
  - Guide les acteurs
    - Architecte, développeur, testeur
  - Vue conceptuelle qui reflète le logiciel en cours d'exécution



- Fiabilisation des évolutions
  - Couplage des validations statiques et dynamiques des évolutions

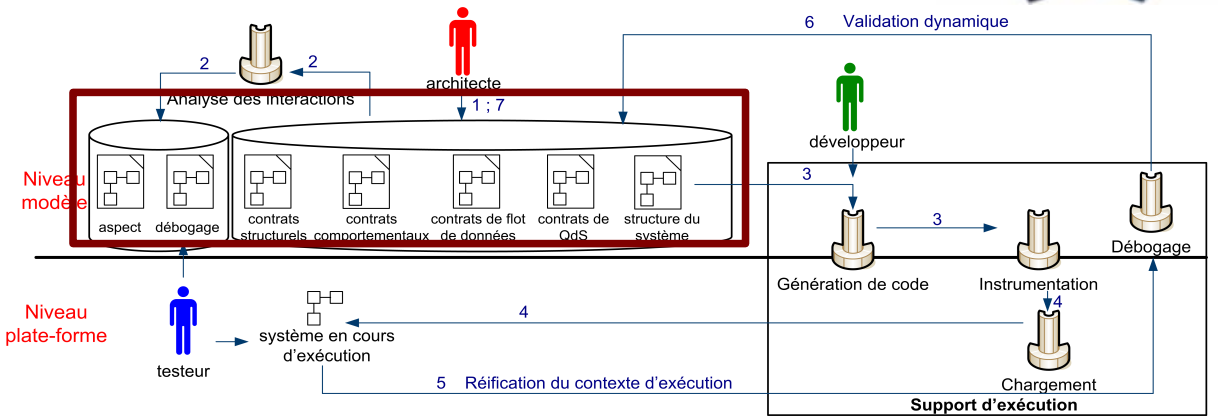
- Multi plates-formes
  - Intégration transparente des mécanismes de validation dans le logiciel
  - Extensible et générique
    - Composants / services

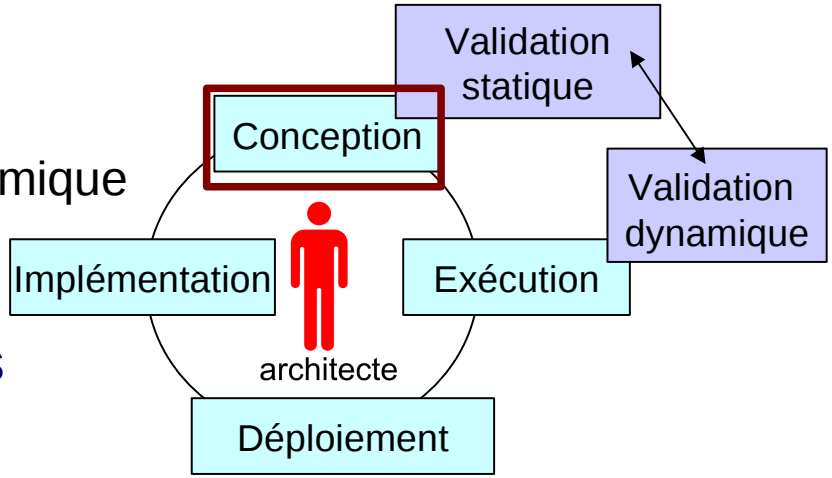
4. Canevas CALICO



Plan

- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives





4. Cycle de développement de CALICO

4.1. Conception

- Problématique
  - Spécifier l'assemblage de composants /services
  - Décrire les propriétés applicatives

4. Cycle de développement de CALICO

4.1. Conception

- Problématique
  - Spécifier l'assemblage de composants /services
  - Décrire les propriétés applicatives

- CALICO
  - Offre une large palette de spécifications
    - Structure, comportement, flot de données et de QdS
  - Intégration uniforme des catégories de propriétés applicatives
    - Paradigme hypothèse/garantie

4. Cycle de développement de CALICO

4.1. Conception

- Problématique
  - Spécifier l'assemblage de composants /services
  - Décrire les propriétés applicatives

- CALICO
  - Offre une large palette de spécifications
    - Structure, comportement, flot de données et de QdS
  - Intégration uniforme des catégories de propriétés applicatives
    - Paradigme hypothèse/garantie

Avantages

- Extensible
- Indépendant de la plate-forme d'exécution (composants /services)
  - Permet la spécification de propriétés applicatives sur des plates-formes qui ne la supportent pas nativement

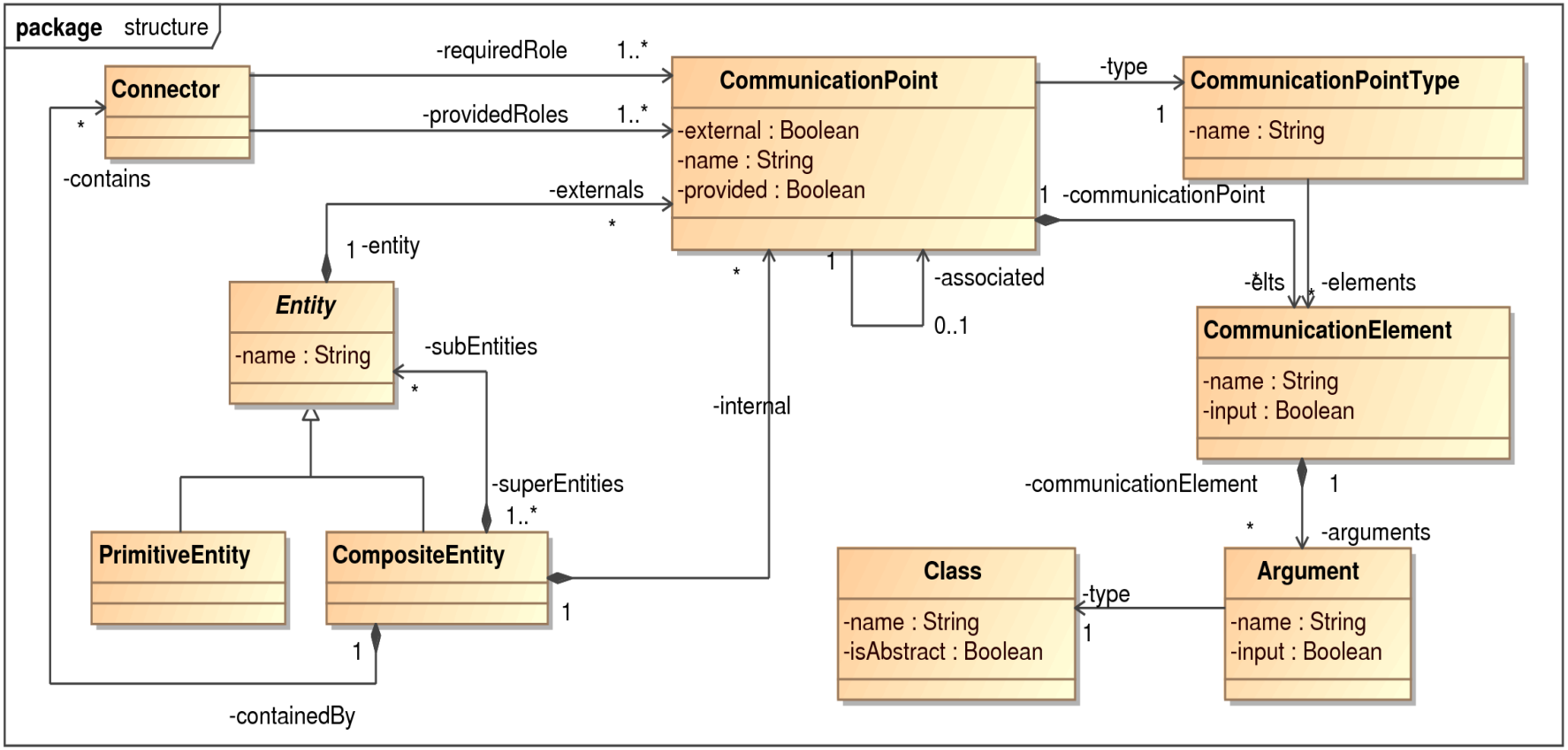
architecte

4. Cycle de développement de CALICO

4.1. Conception

4.1.1. Métamodèle de structure du système

- Assemblage de composants et de services



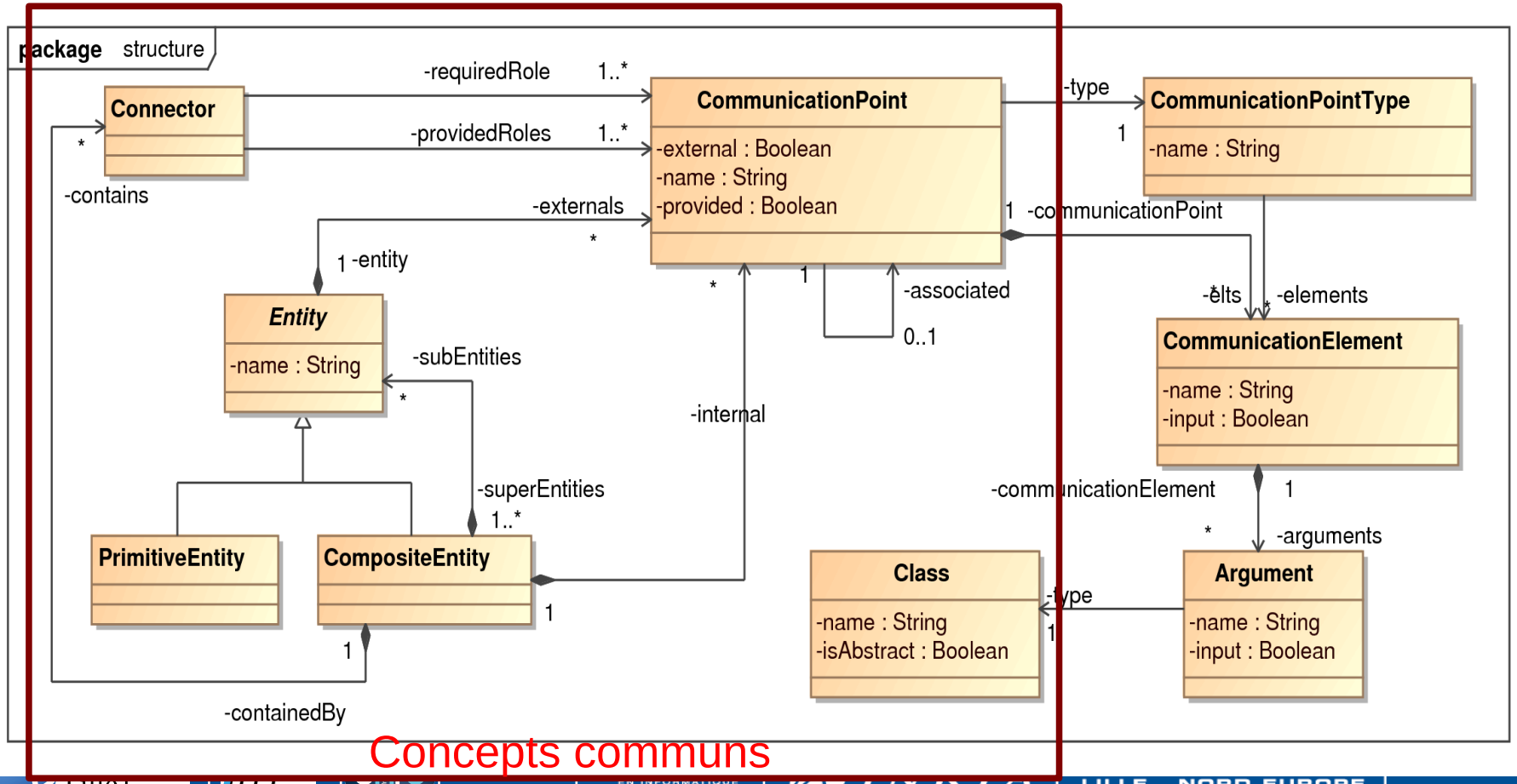
architecte

4. Cycle de développement de CALICO

4.1. Conception

4.1.1. Métamodèle de structure du système

- Assemblage de composants et de services
  - Structure basée sur l'existant (16 modèles à composants/services)



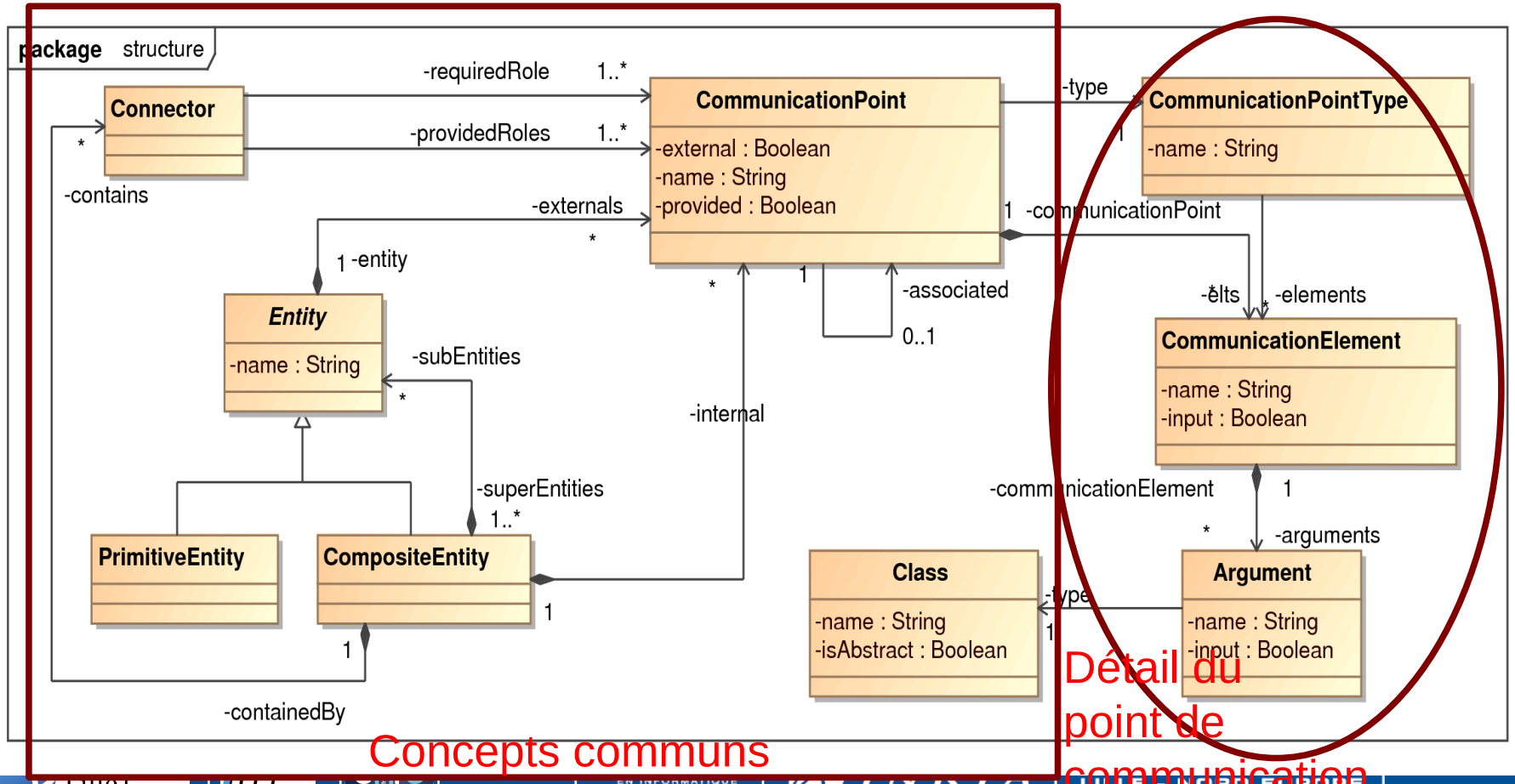
architecte

4. Cycle de développement de CALICO

4.1. Conception

4.1.1. Métamodèle de structure du système

- Assemblage de composants et de services
  - Structure basée sur l'existant (16 modèles à composants/services)
  - Raisonnement sur les interactions entre composants/services



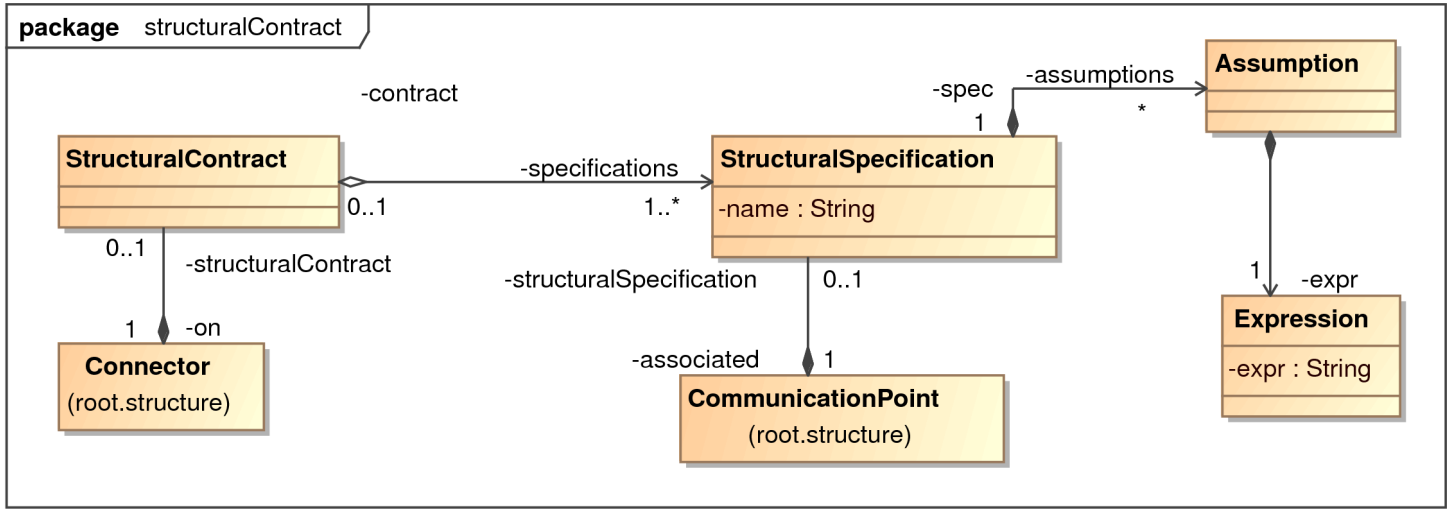
architecte

4. Cycle de développement de CALICO

4.1. Conception

4.1.2. Métamodèle des propriétés structurelles

- Contraintes structurelles
  - Basé sur un langage de contraintes bien connu (OCL)
    - Spécialisation pour les architectures logicielles



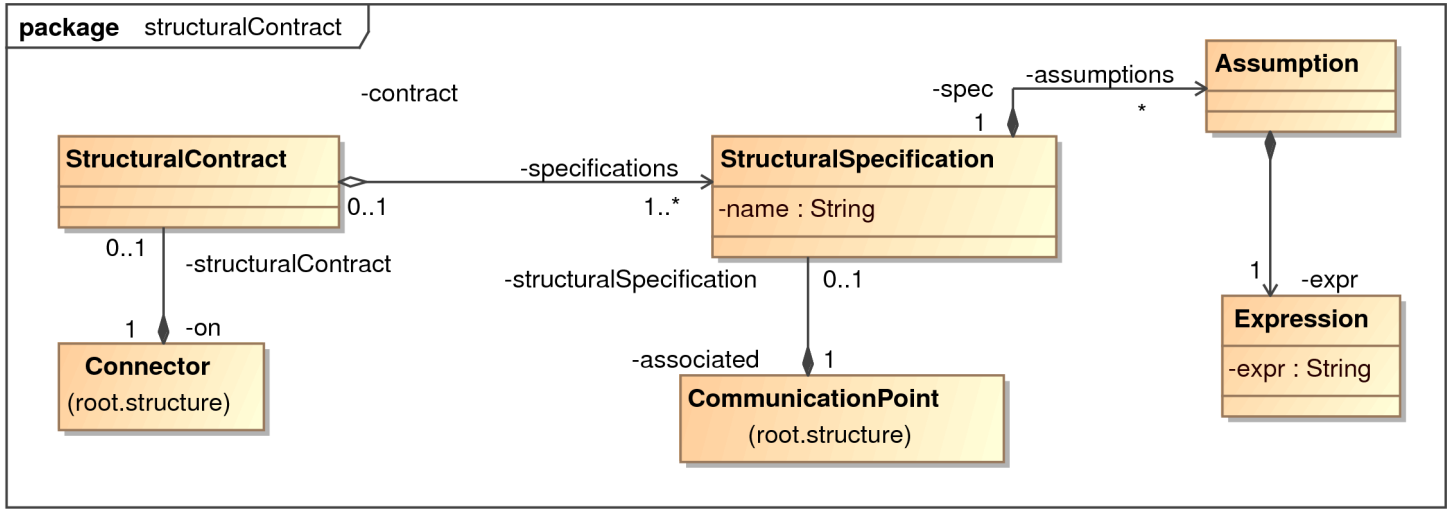
architecte

4. Cycle de développement de CALICO

4.1. Conception

4.1.2. Métamodèle des propriétés structurelles

- Contraintes structurelles
  - Basé sur un langage de contraintes bien connu (OCL)
    - Spécialisation pour les architectures logicielles



Exemple DMP:

on MedicalServer.in:

other.entity.name=”Cache”

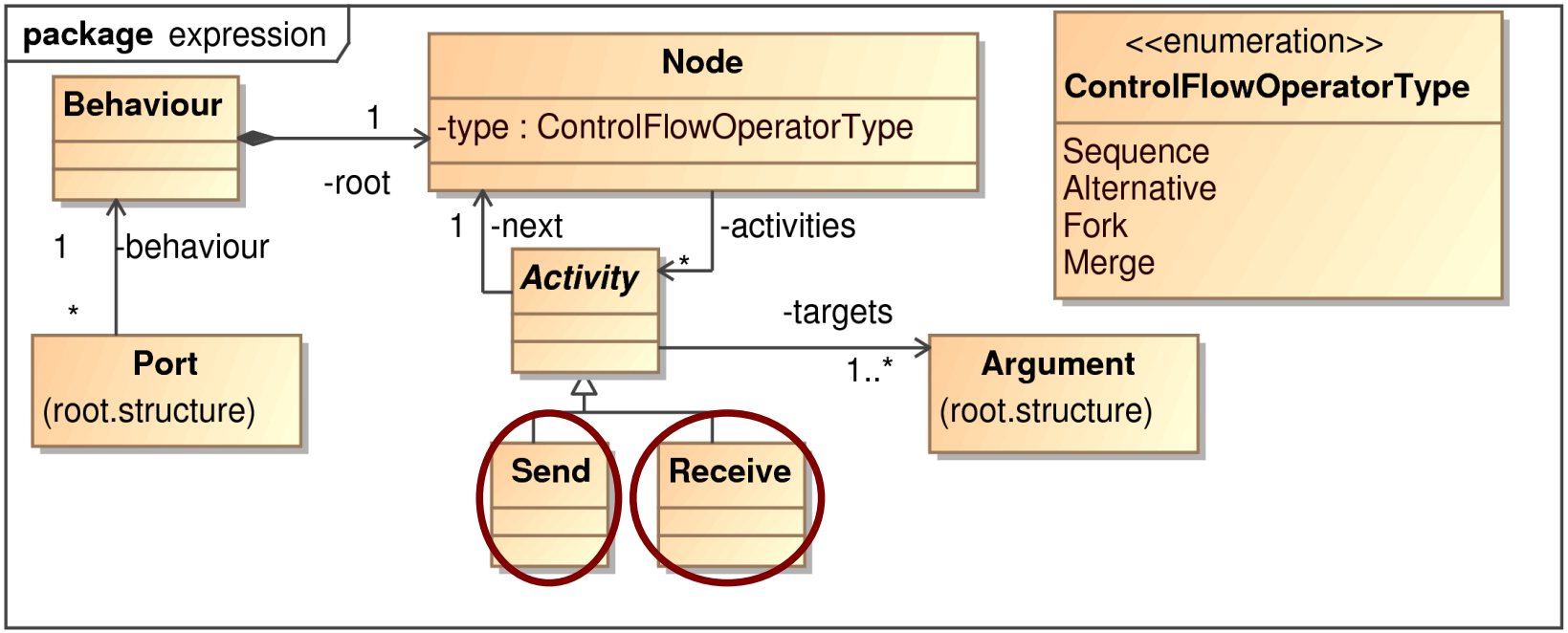
architecte

4. Cycle de développement de CALICO

4.1. Conception

4.1.3. Métamodèle des propriétés comportementales

- Flot de contrôle entrant et sortant des points de communication
  - Détecter les interblocages
  - Baser sur le graphe de flot de contrôle (comme BPMN)
  - Utilise les opérateurs de composition du flot de contrôle non structurés [Aalst07]



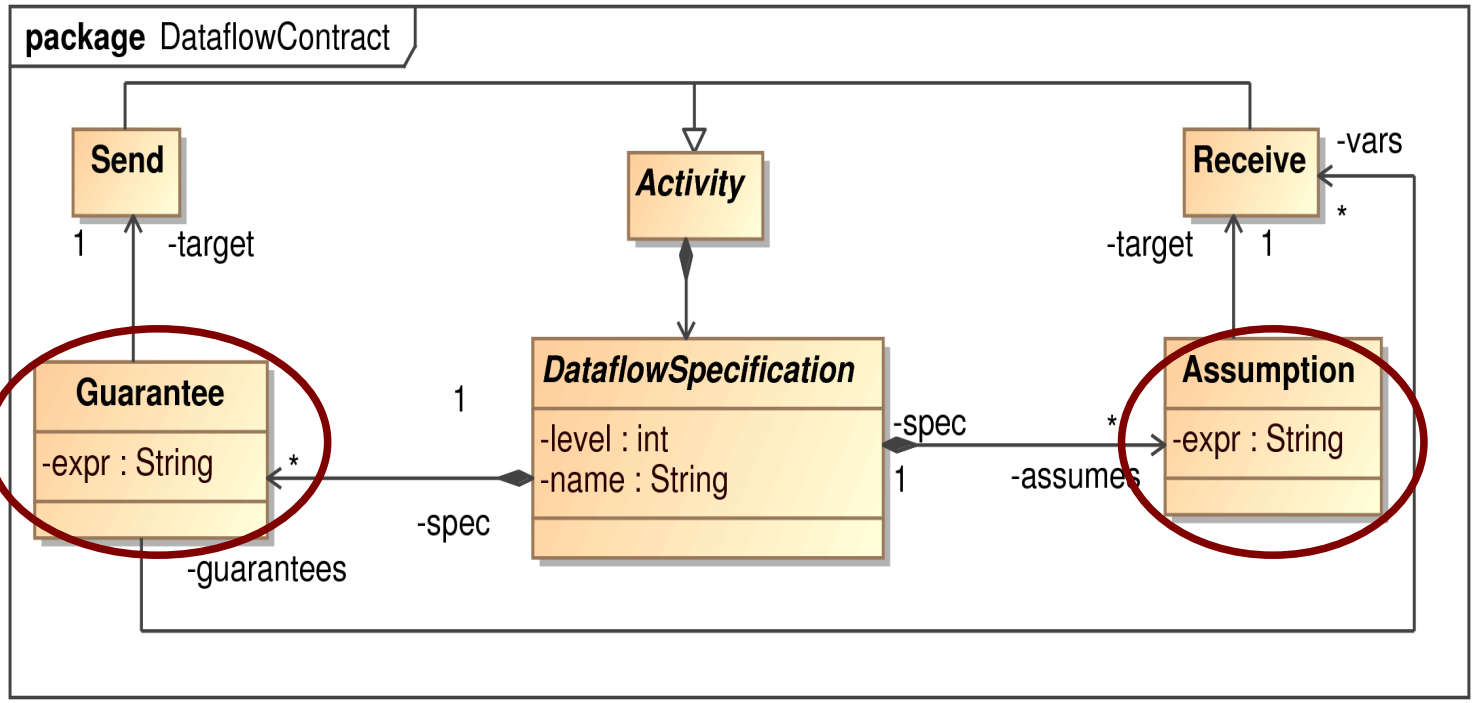
architecte

4. Cycle de développement de CALICO

4.1. Conception

4.1.4. Métamodèle des propriétés de flot de données

- Intervalle de valeurs autorisées des messages entrants et sortants
  - Requis pour raisonner sur des applications de diffusion d'information
  - Repose sur le concept de pre/post conditions

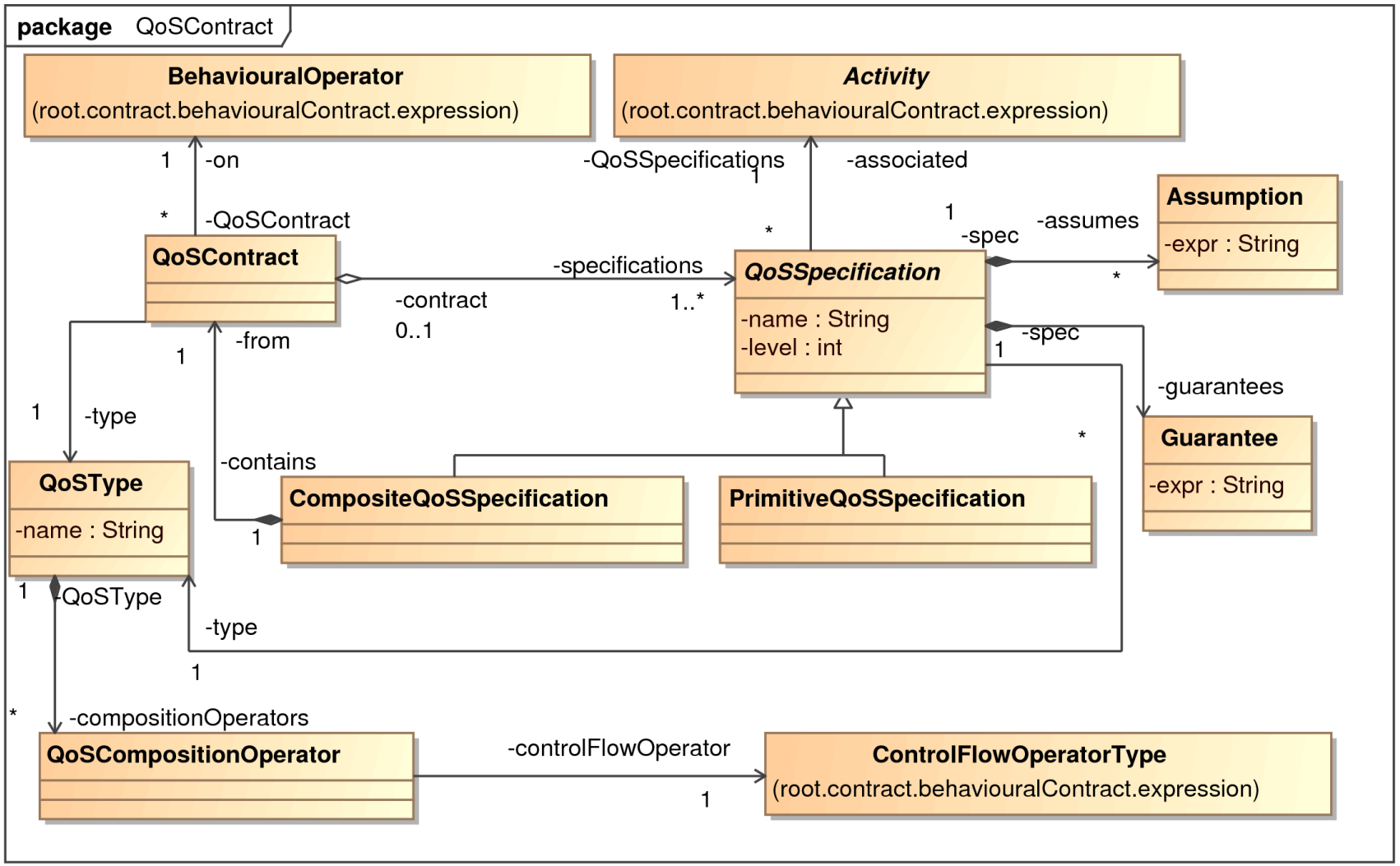


4. Cycle de développement de CALICO

4.1. Conception

4.1.5. Métamodèle des propriétés de QdS

architecte

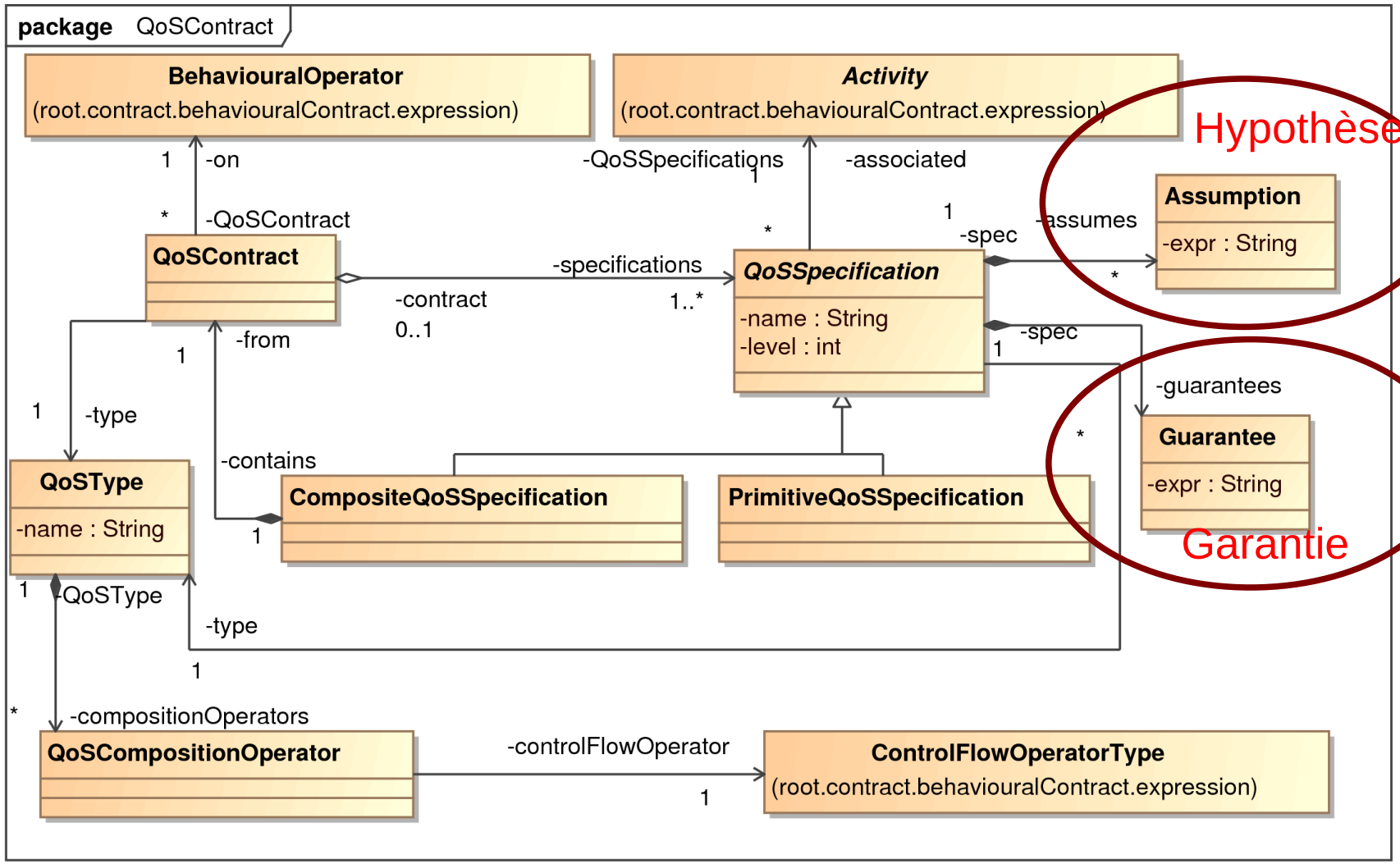


architecte

4. Cycle de développement de CALICO

4.1. Conception

4.1.5. Métamodèle des propriétés de QdS

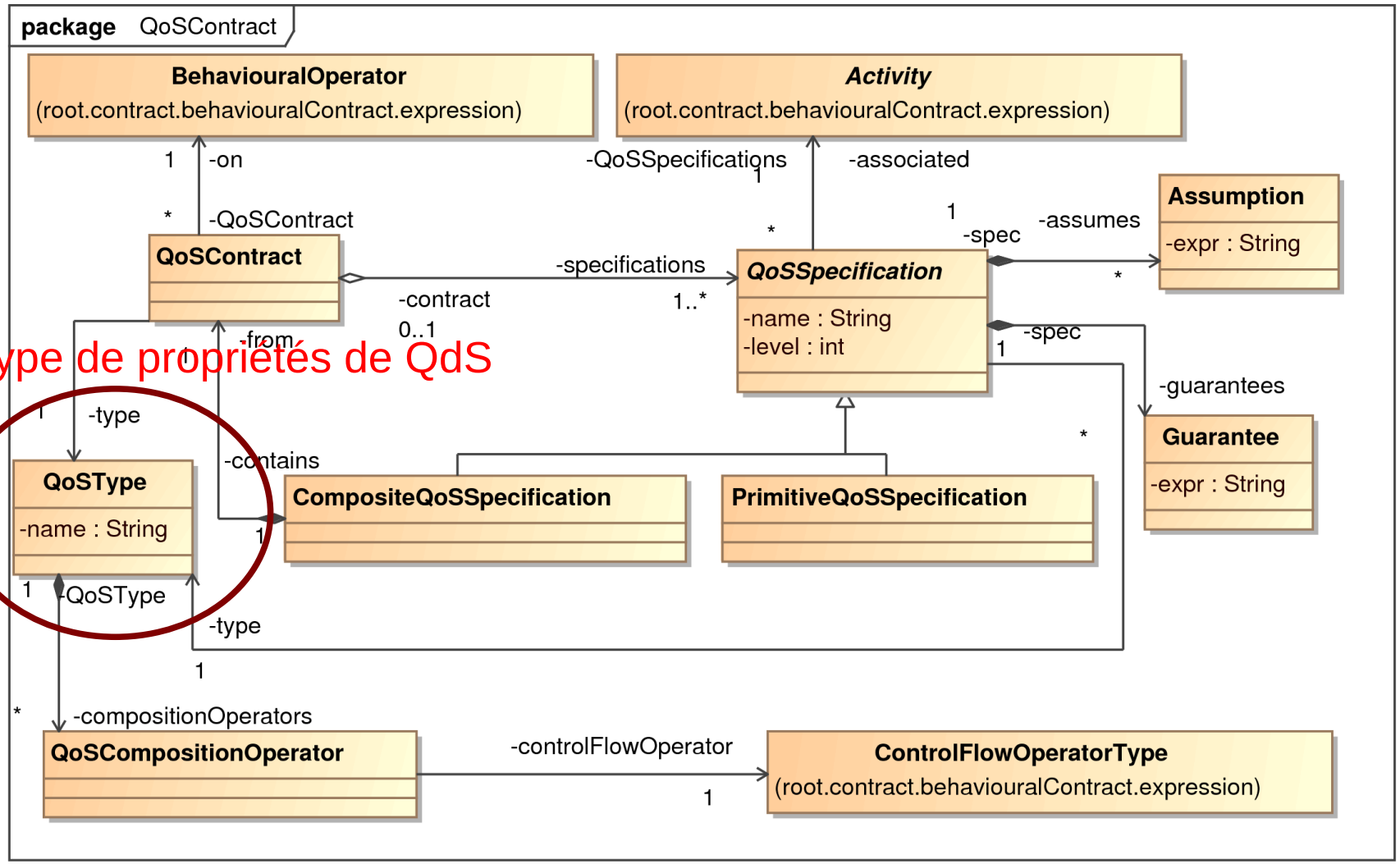


4. Cycle de développement de CALICO

4.1. Conception

4.1.5. Métamodèle des propriétés de QdS

architecte

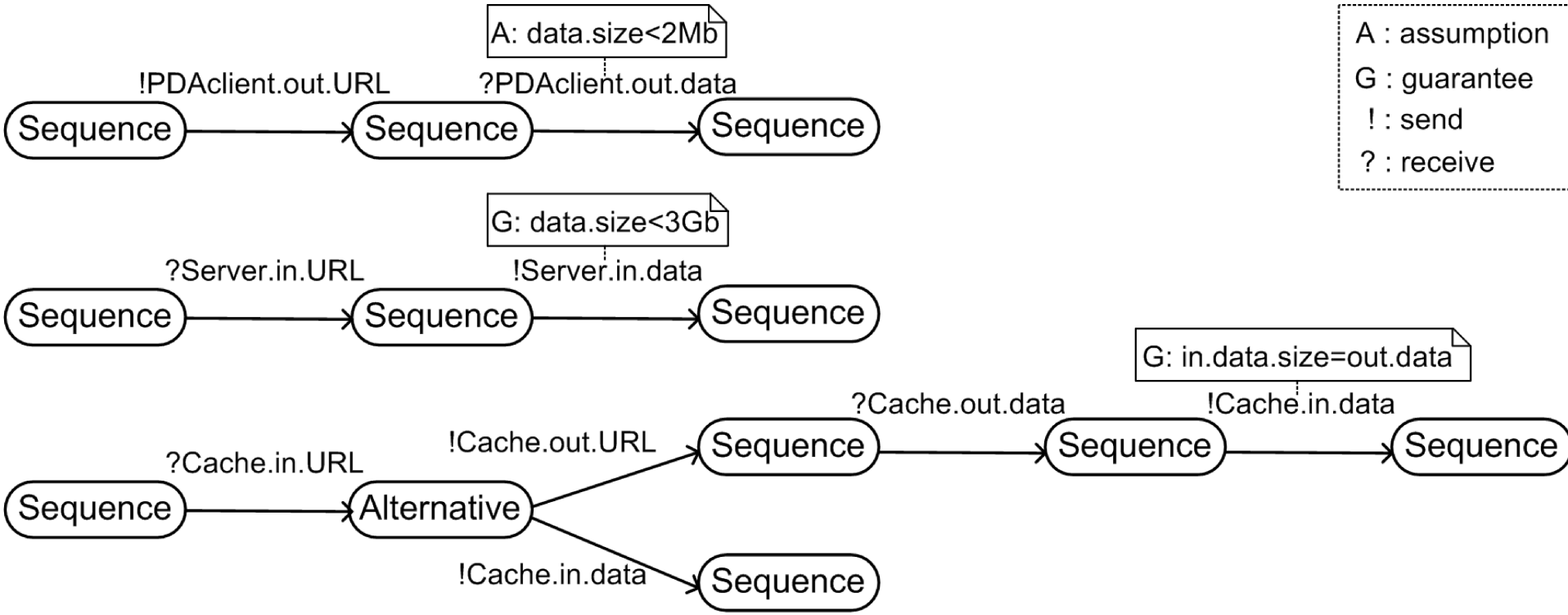


architecte

4. Cycle de développement de CALICO

4.1. Conception

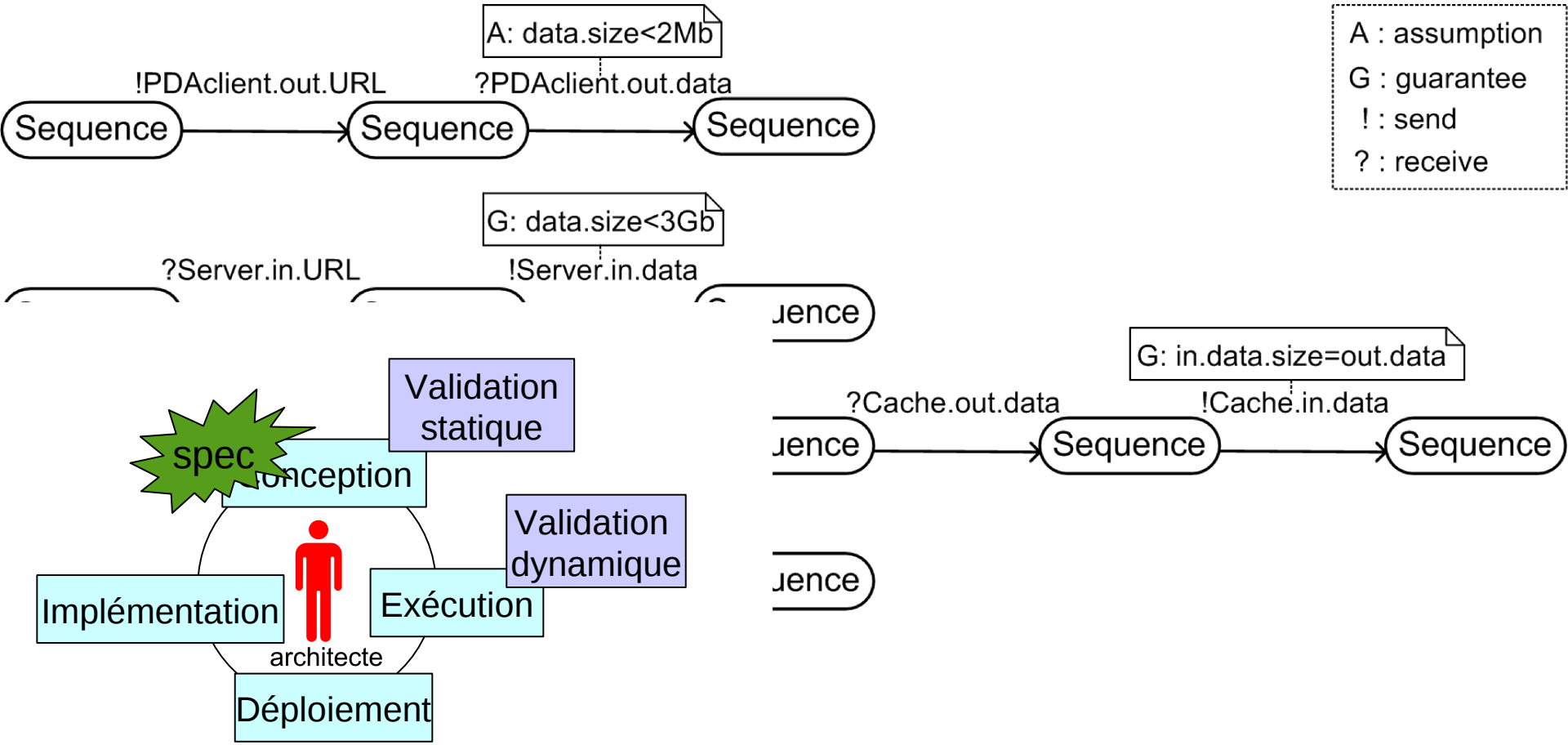
4.1.6. Exemple DMP



4. Cycle de développement de CALICO

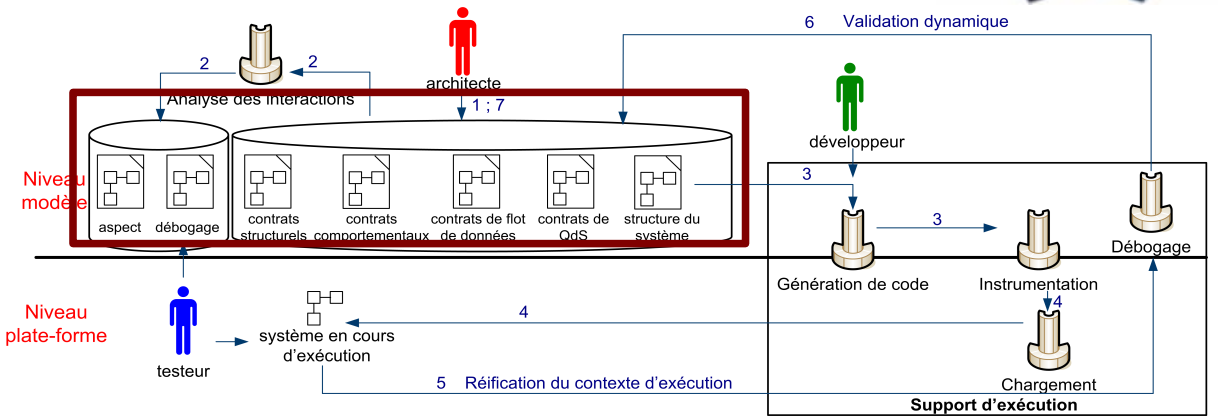
4.1. Conception

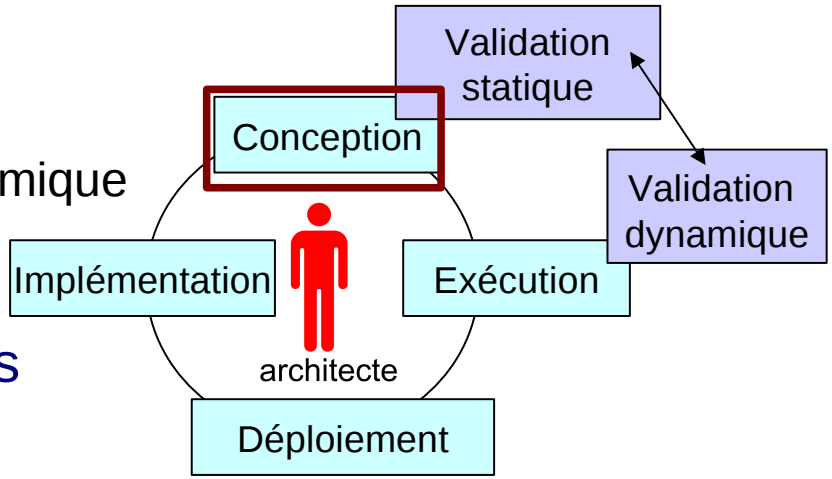
4.1.6. Exemple DMP



Plan

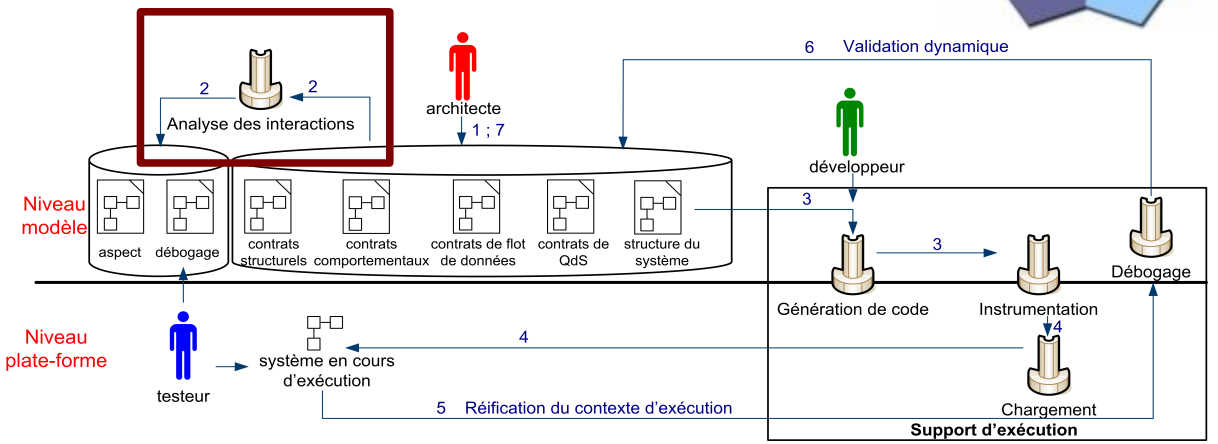
- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives

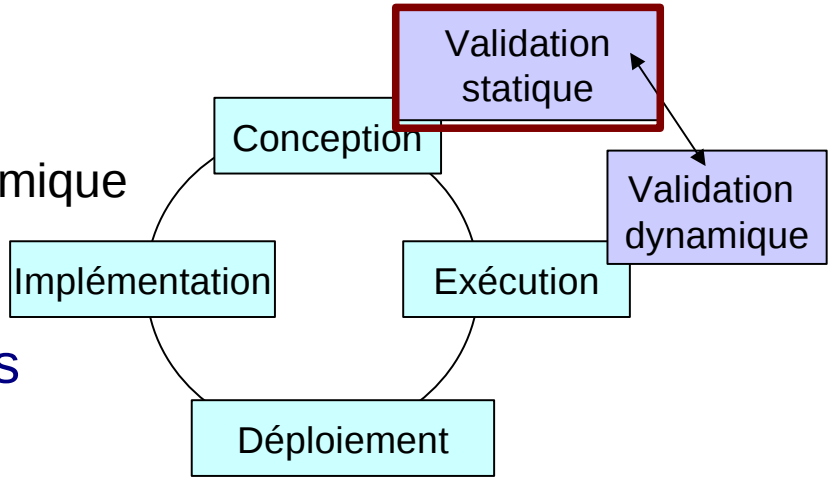




Plan

- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives





4. Cycle de développement de CALICO

4.2. Validation statique

4.2.1. Outil d'analyse des interactions

- Problématique
  - Manque d'intégration d'outils d'analyse statique

4. Cycle de développement de CALICO

4.2. Validation statique

4.2.1. Outil d'analyse des interactions

- Problématique
  - Manque d'intégration d'outils d'analyse statique

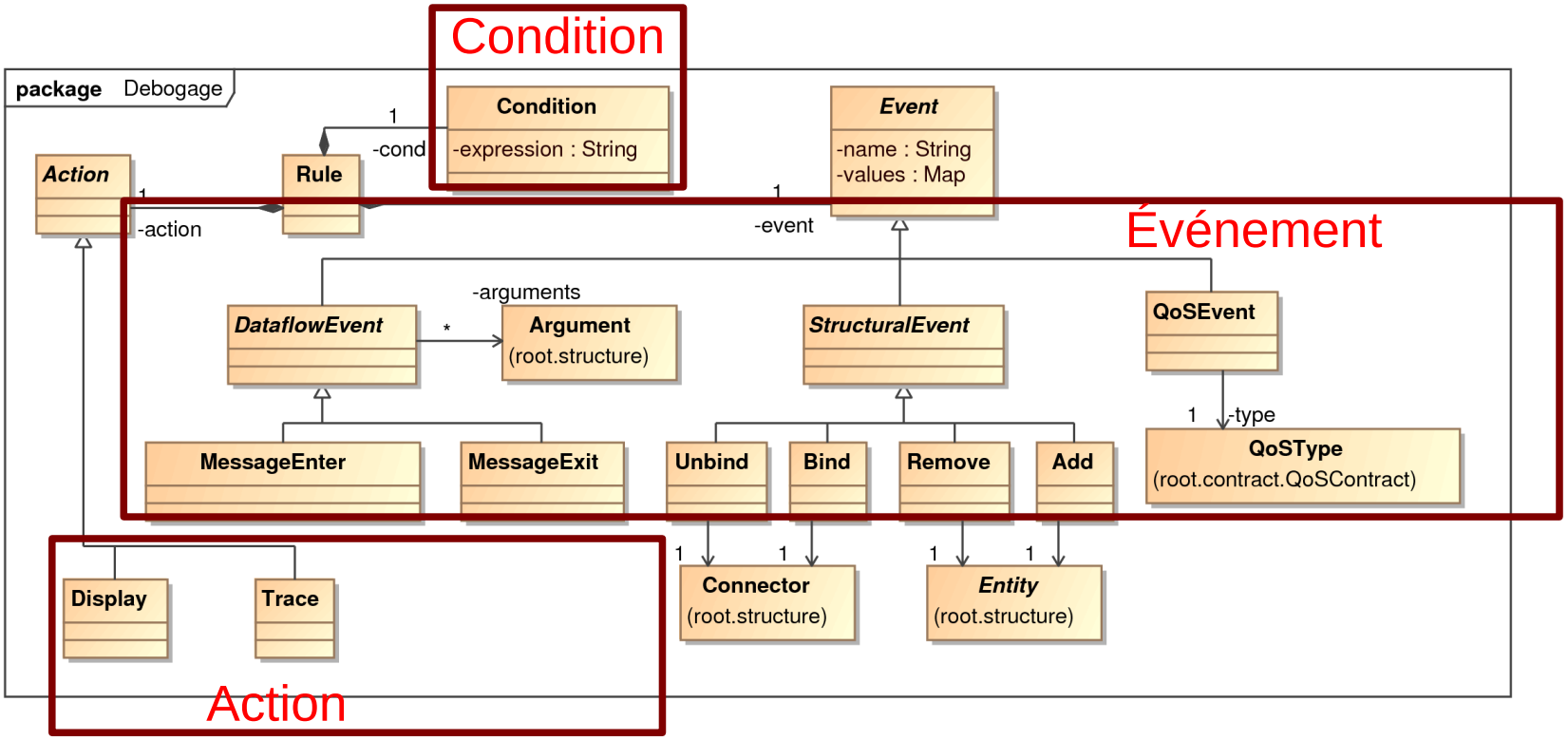
- Outil d'analyse des interactions de CALICO
  - Vérification de la compatibilité des interactions entre composants / services
  - 3 types d'interaction
    - Compatible : ok
    - Incompatible : message d'erreur
    - Partiellement compatible : préparation des analyses dynamiques

4. Cycle de développement de CALICO

4.2. Validation statique

4.2.1. Outil d'analyse des interactions

- Problématique
  - Manque d'intégration d'outils d'analyse statique



4. Cycle de développement de CALICO

4.2. Validation statique

4.2.1. Outil d'analyse des interactions

- Problématique
  - Manque d'intégration d'outils d'analyse statique

- Outil d'analyse des interactions de CALICO
  - Vérification de la compatibilité des interactions entre composants / services
  - 3 types d'interaction
    - Compatible : ok
    - Incompatible : message d'erreur
    - Partiellement compatible : préparation des analyses dynamiques

Avantage

- Cadre fédérateur d'intégration des analyses existantes
  - Paradigme hypothèse/obligation [Lamport93]
  - Factorisation des analyses pour différentes plates-formes

4. Cycle de développement de CALICO

4.2. Validation statique

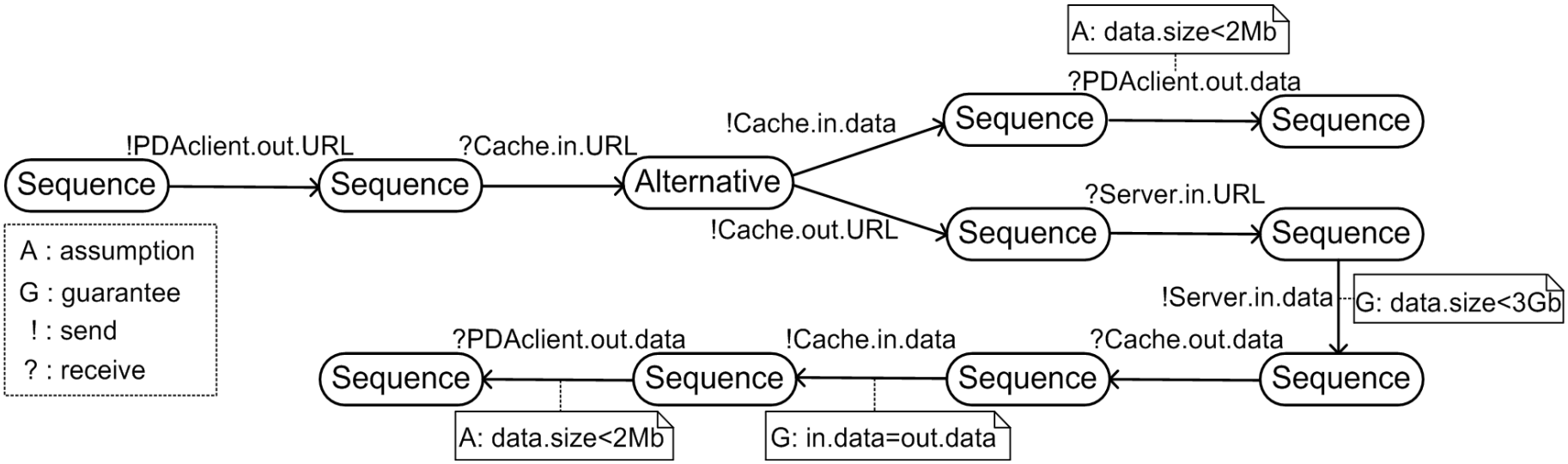
4.2.2. Intégration des analyses existantes

- Analyse structurelle
  - Repose sur OCL [OMG03]
- Analyse du comportement
  - Repose sur les algèbres de processus existants (*e.g.* : CSP [Hoare85])
  - Construit le GFC global en synchronisant les GFCs [Moffat07]
- Analyse de flots de données
  - Repose sur la validation partielle de programme [Kildall73]
  - Propage les garanties en avant
  - Propage les hypothèses en arrière
- Analyse de la QdS
  - Repose sur la prédiction de qualité de services dans les flots de contrôle

4. Cycle de développement de CALICO

4.2. Validation statique

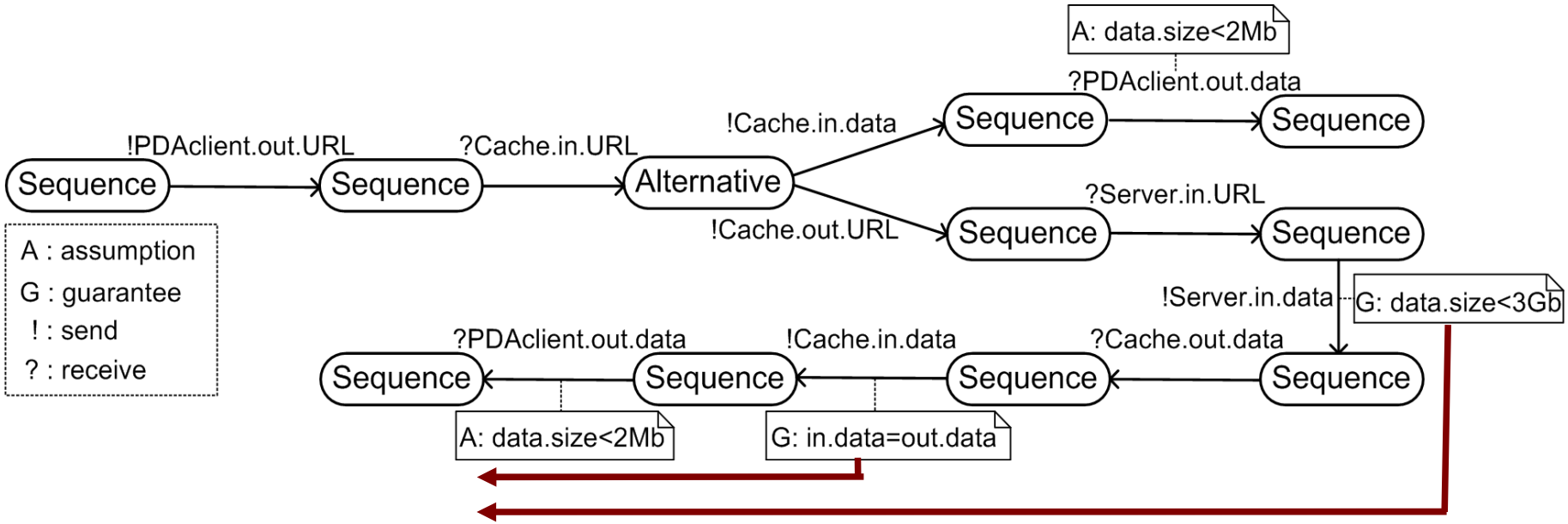
4.2.3. Exemple DMP



4. Cycle de développement de CALICO

4.2. Validation statique

4.2.3. Exemple DMP

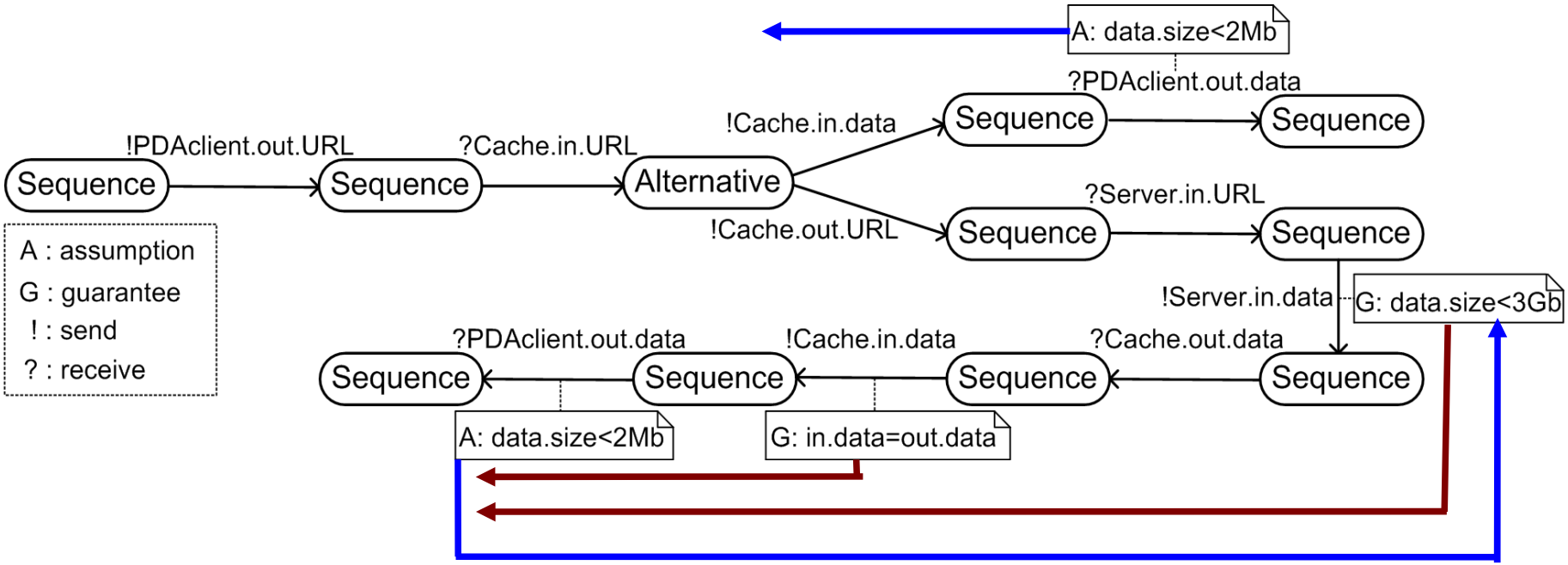


Propagation avant des garanties

4. Cycle de développement de CALICO

4.2. Validation statique

4.2.3. Exemple DMP



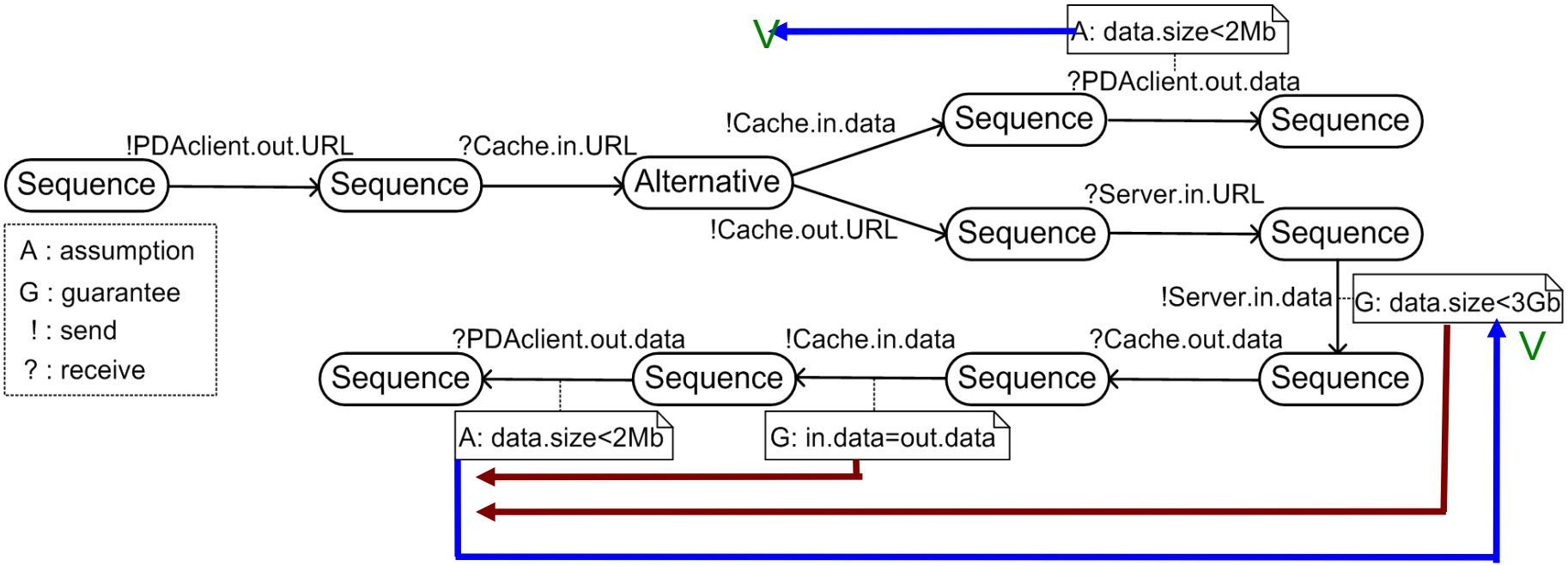
Propagation avant des garanties

Propagation arrière des hypothèses

4. Cycle de développement de CALICO

4.2. Validation statique

4.2.3. Exemple DMP



Propagation avant des garanties

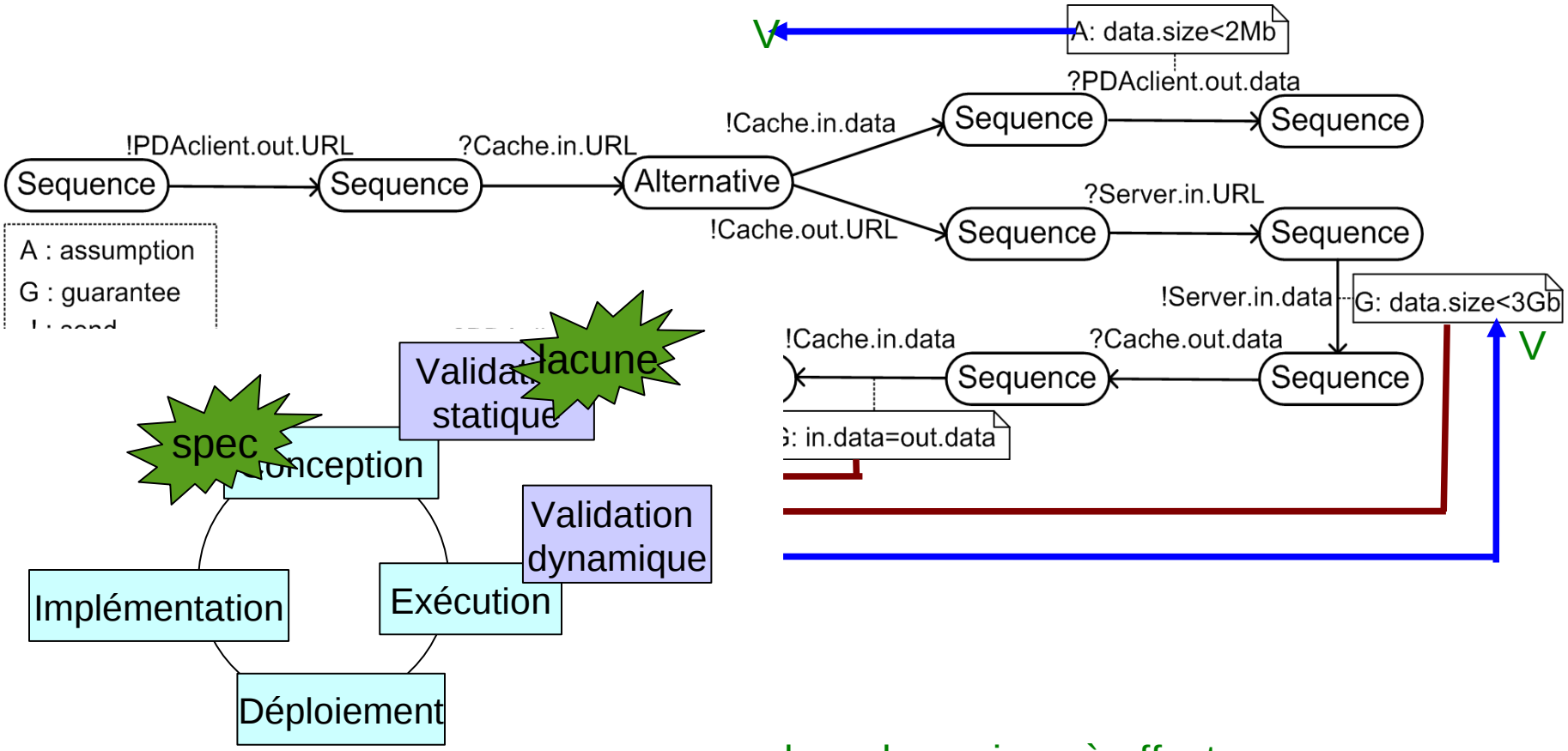
Propagation arrière des hypothèses

Identification de la localisation de l'analyse dynamique à effectuer

4. Cycle de développement de CALICO

4.2. Validation statique

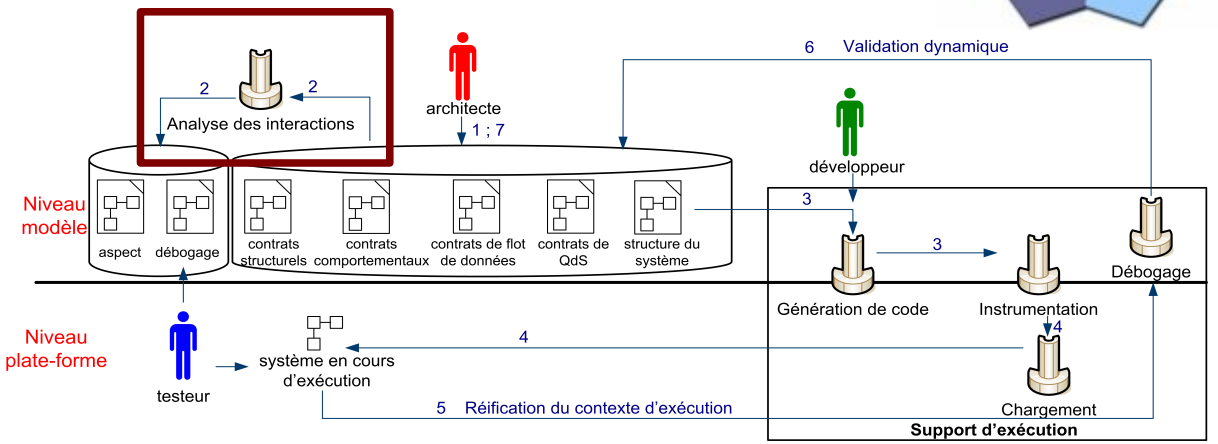
4.2.3. Exemple DMP

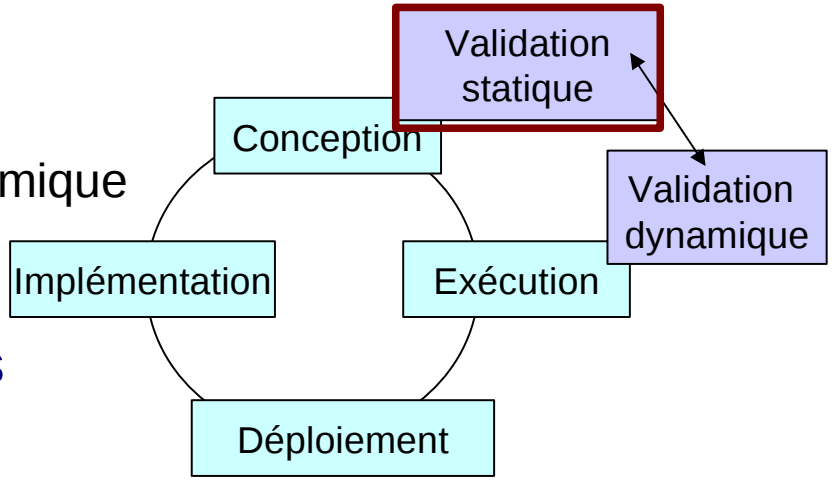


lyse dynamique à effectuer

Plan

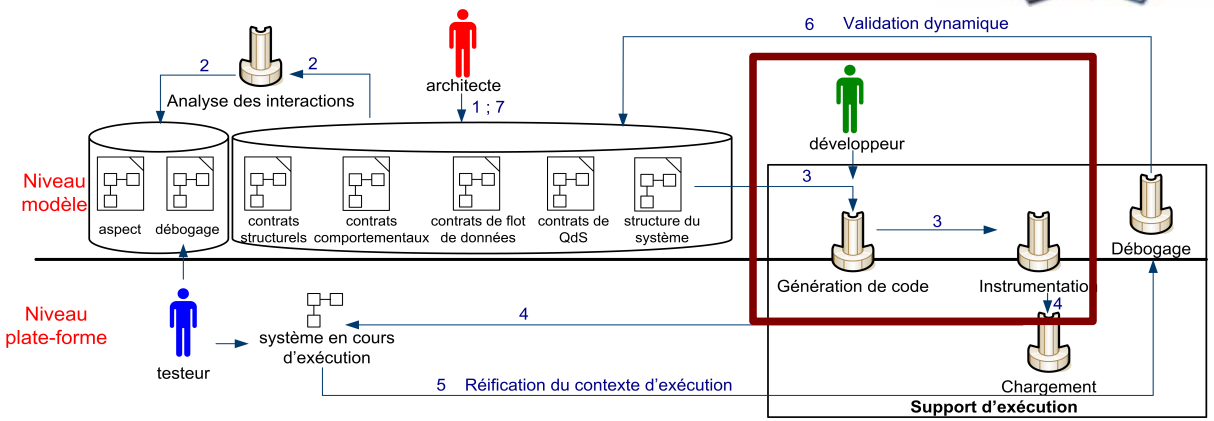
- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives

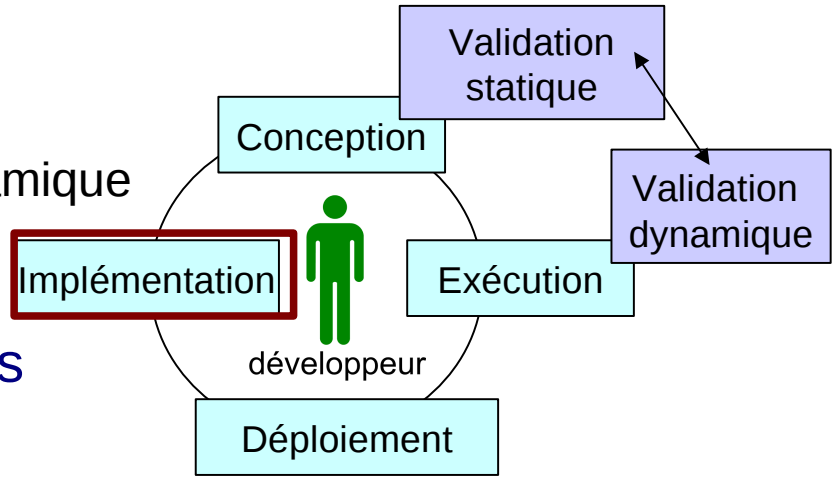


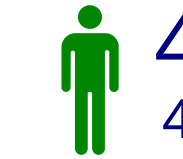


Plan

- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives





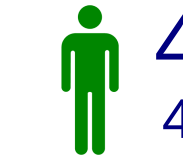


4. Cycle de développement de CALICO

4.3. Implémentation

4.3.1. Outil de génération de code

- Problématique
  - Ecriture du code technique des composants/services
  - Conformance avec la spécification



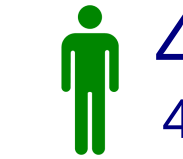
4. Cycle de développement de CALICO

4.3. Implémentation

4.3.1. Outil de génération de code

- Problématique
  - Ecriture du code technique des composants/services
  - Conformance avec la spécification

- Outil de génération de code de CALICO
  - Squelette de code des composants /services (code technique)
  - ADL



4. Cycle de développement de CALICO

4.3. Implémentation

4.3.1. Outil de génération de code

- Problématique
  - Ecriture du code technique des composants/services
  - Conformance avec la spécification

- Outil de génération de code de CALICO
  - Squelette de code des composants /services (code technique)
  - ADL

Avantages

- Limitation d'introduction d'incohérence entre conception et implémentation
- Focalisation du développement sur le code métier

4. Cycle de développement de CALICO

4.3. Implémentation

4.3.2. Exemple DMP

```
@Component(name="Cache", provides=@Interface(...))
public class CacheImpl implements SearchDataItf
{
  @Requires(name="out")
  private ImageServer.SearchDataItf p2;

  public Image get(URL url)
  {
    ...
  }
}
```

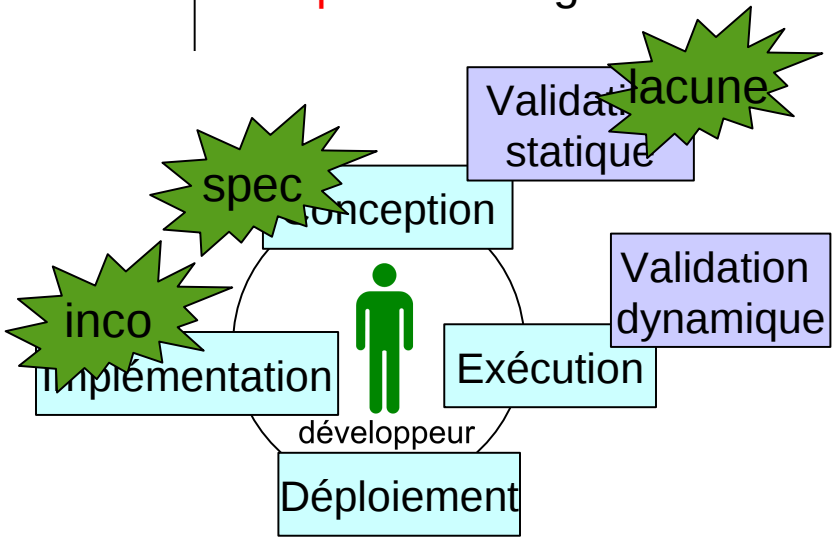
4. Cycle de développement de CALICO

4.3. Implémentation

4.3.2. Exemple DMP

développeur

```
@Component(name="Cache", provides=@Interface(...))
public class CacheImpl implements SearchDataItf
{
  @Requires(name="out")
  private ImageServer.SearchDataItf p2;
```



4. Cycle de développement de CALICO

4.3. Implémentation

4.3.3. Outil d'instrumentation

- Problématique
  - Besoin d'observer les données d'exécution requises par les analyses dynamiques
  - Manque de fonctionnalités d'observation (données, QdS)

4. Cycle de développement de CALICO

4.3. Implémentation

4.3.3. Outil d'instrumentation

- Problématique
  - Besoin d'observer les données d'exécution requises par les analyses dynamiques
  - Manque de fonctionnalités d'observation (données, QdS)

- Outil d'instrumentation
  - Parcourt chaque analyse dynamique spécifiée dans le modèle de débogage
  - Identifie les données d'exécution requises
  - Instrumente l'application pour ajouter/supprimer les sondes
    - Ajoute des aspects au niveau de l'architecture
    - Ajoute des aspects au niveau de l'implémentation

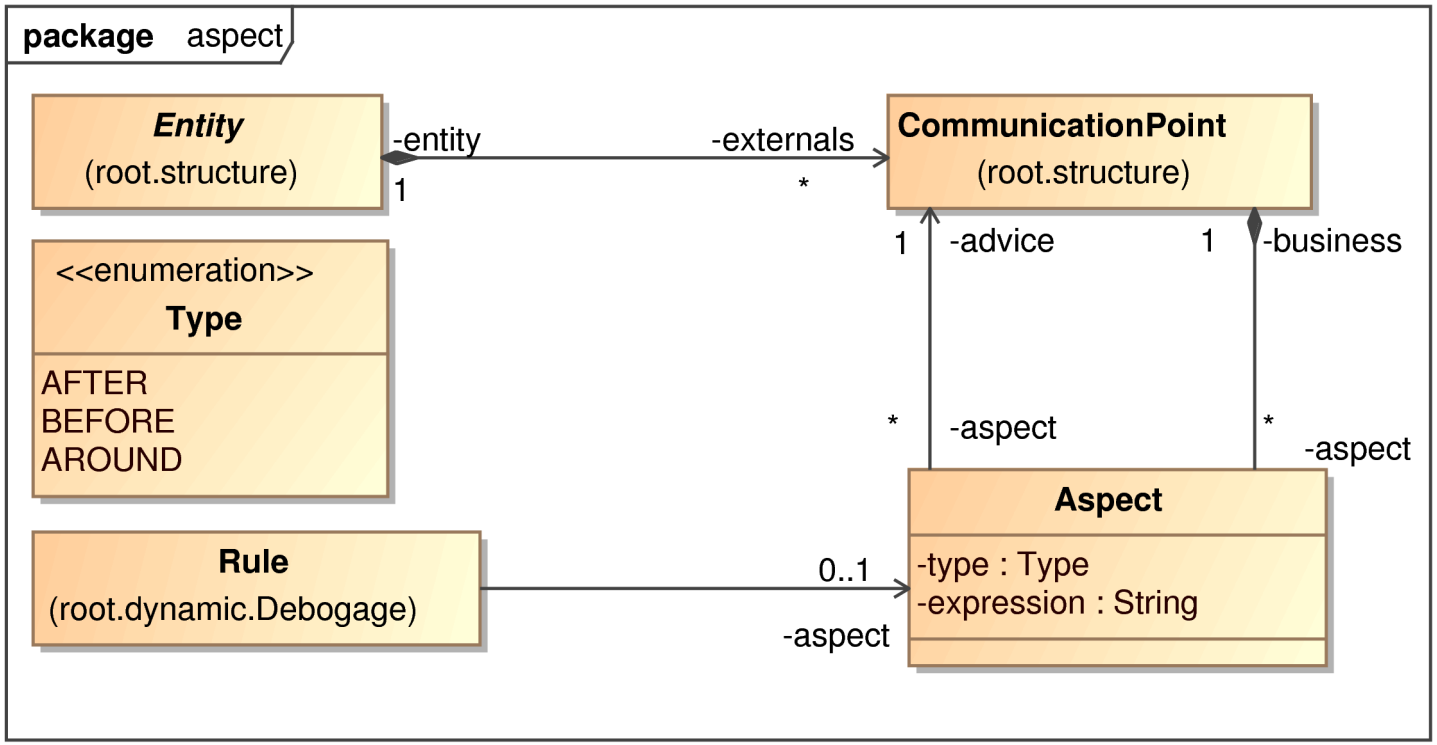
développeur

4. Cycle de développement de CALICO

4.3. Implémentation

4.3.3. Outil d'instrumentation

- Problématique
  - Besoin d'observer les données d'exécution requises par les analyses



Aspect architectural repose sur l'existant (FAC[Pes07], DAOP-ADL[Pinto03])

4. Cycle de développement de CALICO

4.3. Implémentation

4.3.3. Outil d'instrumentation

- Problématique
  - Besoin d'observer les données d'exécution requises par les analyses dynamiques
  - Manque de fonctionnalités d'observation (données, QdS)

- Outil d'instrumentation
  - Parcourt chaque analyse dynamique spécifiée dans le modèle de débogage
  - Identifie les données d'exécution requises
  - Instrumente l'application pour ajouter/supprimer les sondes
    - Ajoute des aspects au niveau de l'architecture
    - Ajoute des aspects au niveau de l'implémentation

Avantages

- Gestion transparente des analyses dynamiques
- Indépendance des plates-formes (composants / services)
- Extensibilité (intégration de canevas existants de sondes)

4. Cycle de développement de CALICO

4.3. Implémentation

4.3.4. Exemple DMP

- Capture de la trace des messages échangés
  - Tissage de 4 aspects

```
@Component(name="Cache", provides=@Interface(...))
public class CacheImpl implements SearchDataItf
{
  @Requires(name="out")
  private ImageServer.SearchDataItf p2;

  public Image get(URL url, List<String> trace)
  {
     trace.add("?Cache.in.get.URL");
     ...
     __res = this.out.getPicture(url,trace);
    trace = __res.getTrace();
    trace.add("?Cache.out.getPicture.data");
    ...
    trace.add("!Cache.in.get.data");
    return __res;
} }
```

développeur

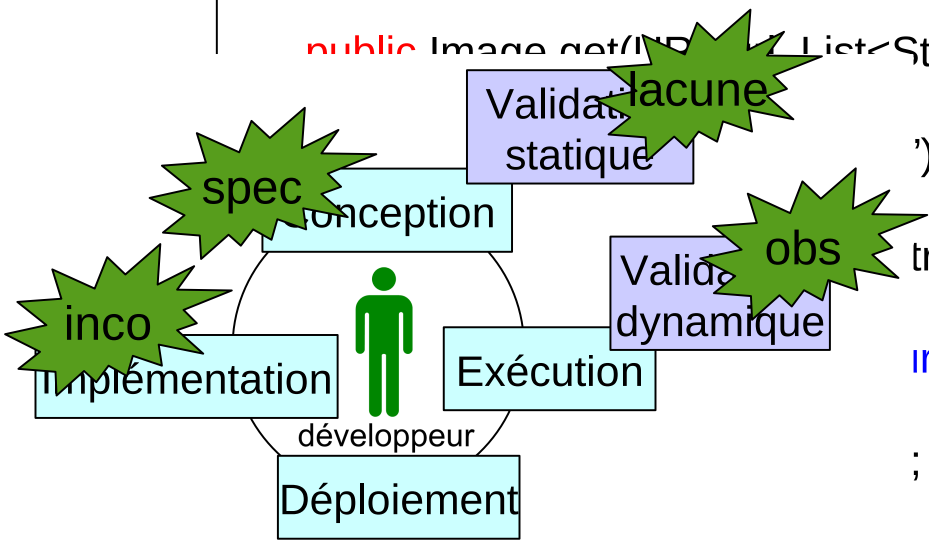
4. Cycle de développement de CALICO

4.3. Implémentation

4.3.4. Exemple DMP

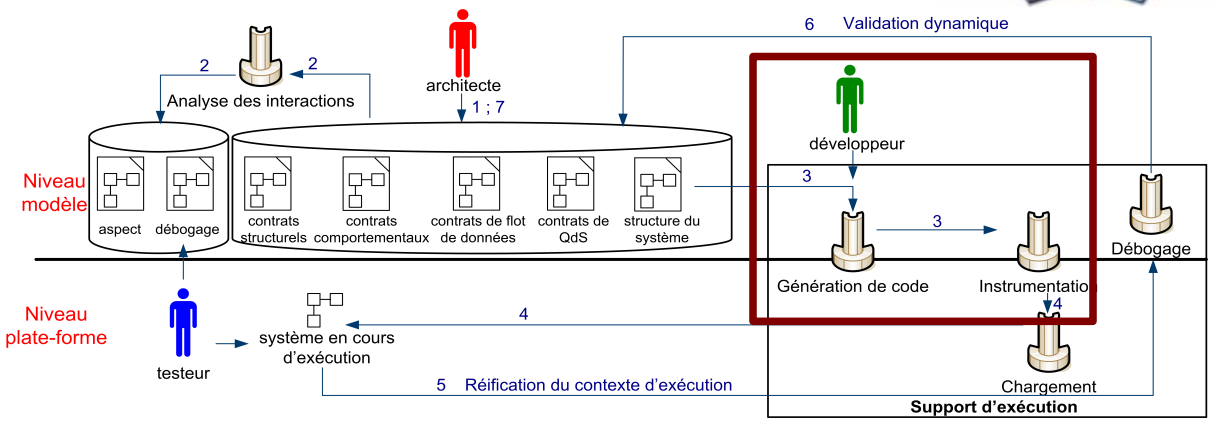
- Capture de la trace des messages échangés
  - Tissage de 4 aspects

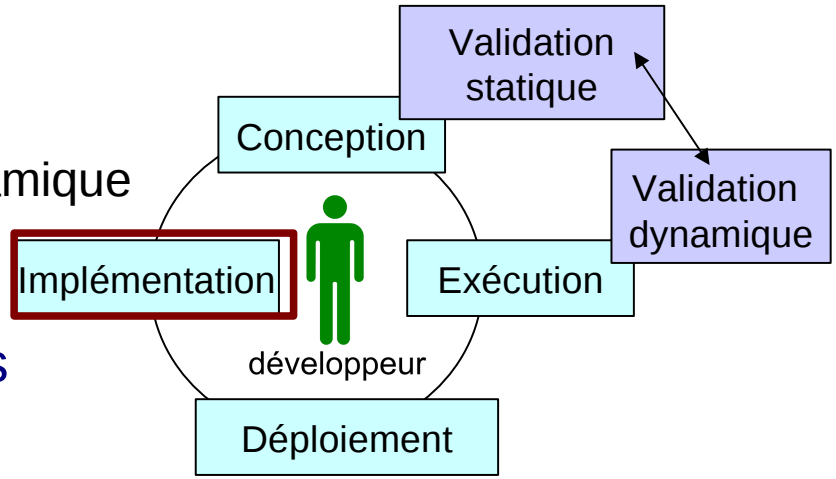
```
@Component(name="Cache", provides=@Interface(...))
public class CacheImpl implements SearchDataItf
{
  @Require(name="out")
  private ImageServer.SearchDataItf p2;

  public Image get(URI ..., List<String> trace)
                                          ");
                                          trace);
                                          ure.data");
                                          ;
```



Plan

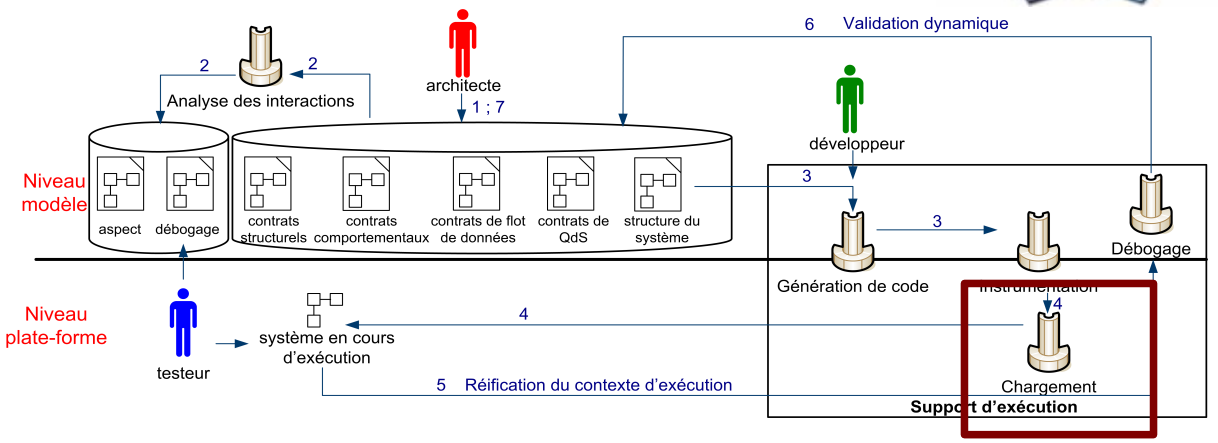
- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives

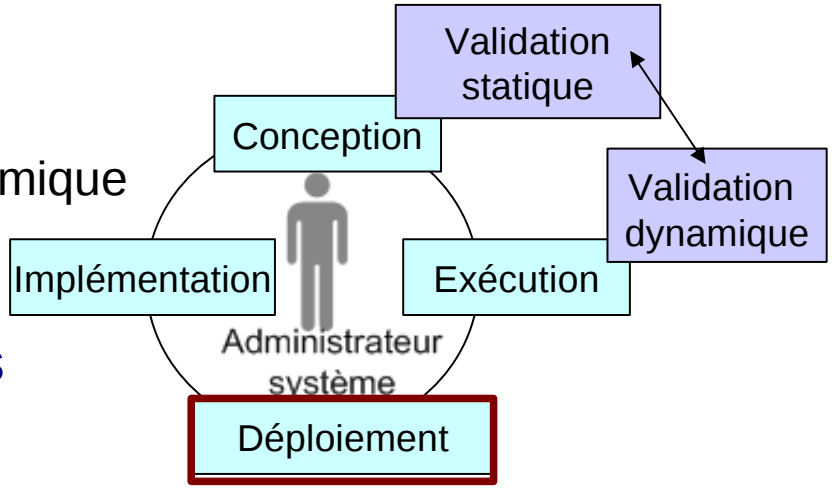




Plan

- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives





4. Cycle de développement de CALICO

4.4. Déploiement

4.4.1. Outil de chargement

- Problématique
  - Propagation des évolutions conceptuelles dans le système en cours d'exécution

4. Cycle de développement de CALICO

4.4. Déploiement

4.4.1. Outil de chargement

- Problématique
  - Propagation des évolutions conceptuelles dans le système en cours d'exécution

- Outil de chargement
  - Effectue un déploiement incrémental

4. Cycle d de CALICO

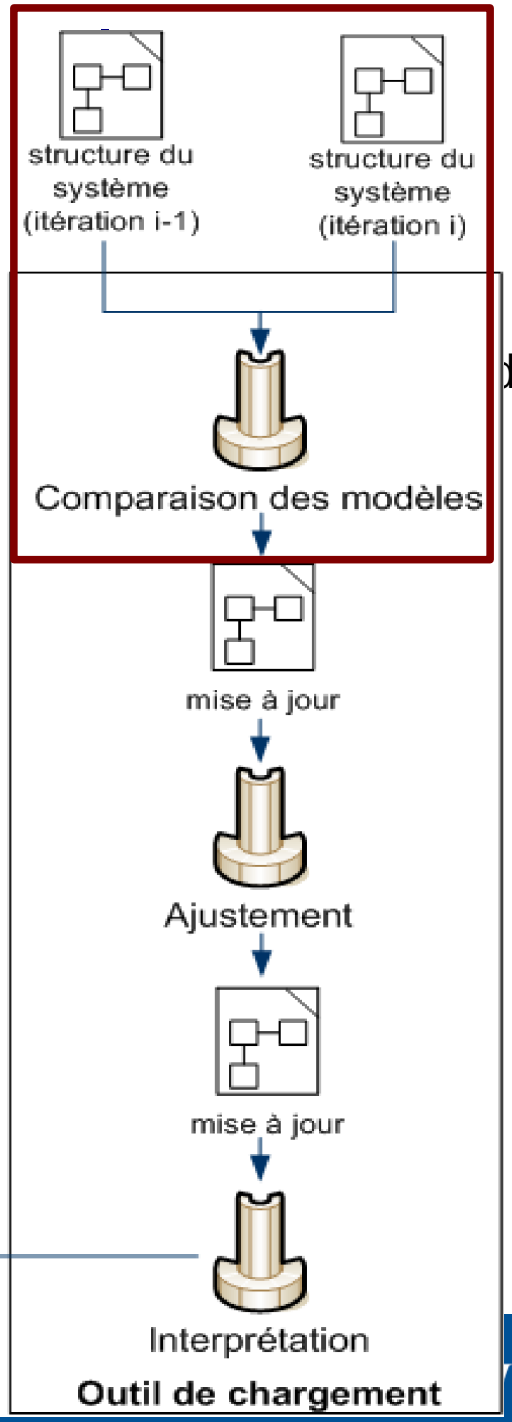
4.4. Déploieme

4.4.1. Outil de

administrateur système

- Problématique
  - Propagatio dans le système en cours d'exécutior
- Outil de charg
  - Effectue ur

Identification des modifications



4. Cycle d de CALICO

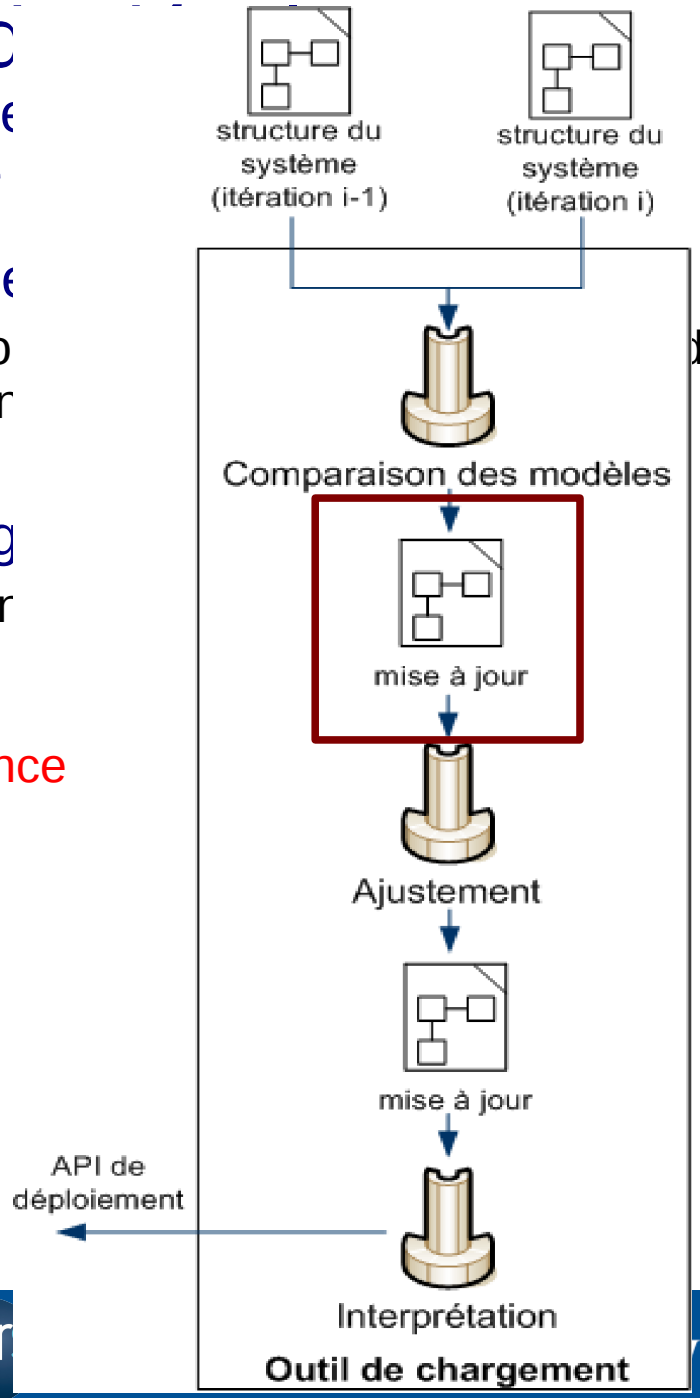
4.4. Déploieme

4.4.1. Outil de

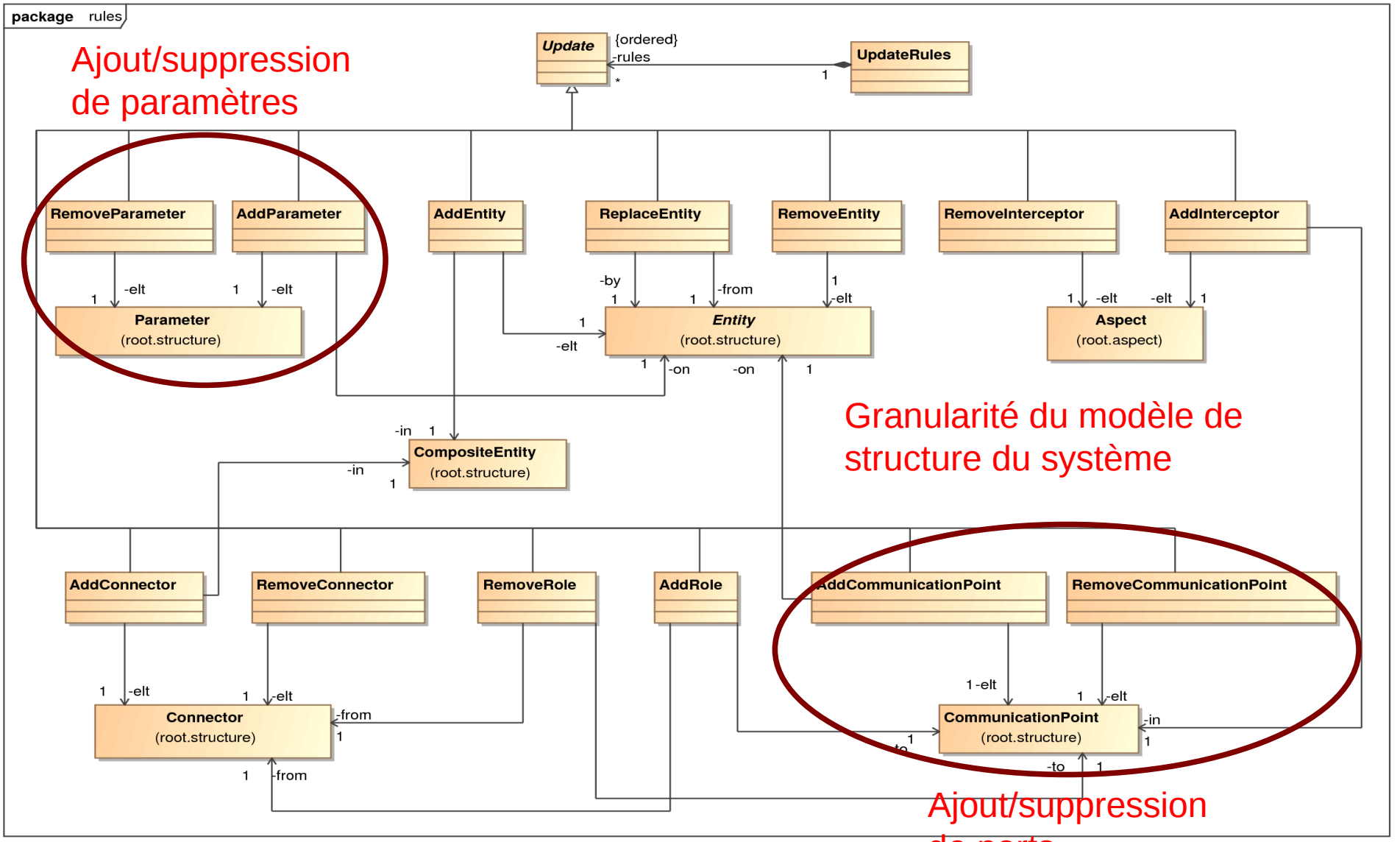
administrateur système

- Problématique
  - Propagatio dans le système en cours d'exécutior
- Outil de charg
  - Effectue ur

Contient la séquence des opérations de construction



4. Cycle d[illegible] de CALICO



Interprétation

Outil de chargement

4. Cycle d de CALICO

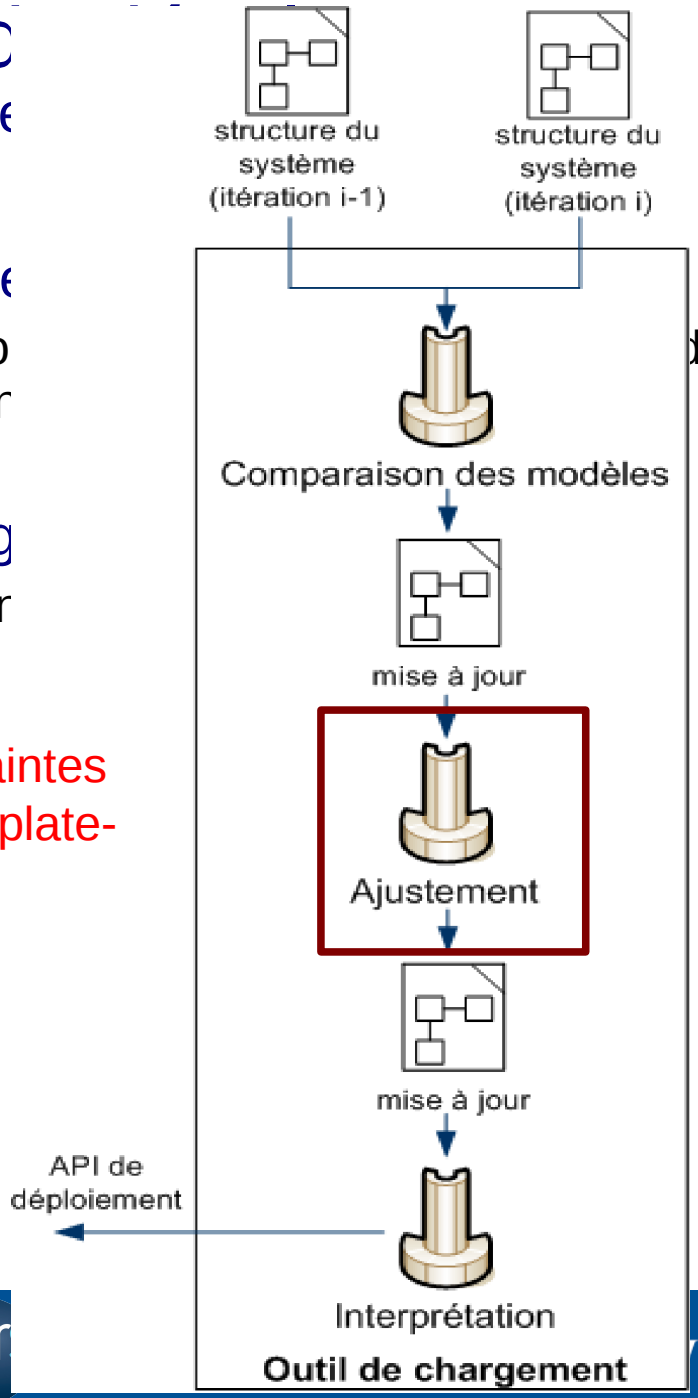
4.4. Déploieme

4.4.1. Outil de

administrateur système

- Problématique
  - Propagatio dans le système en cours d'exécutior
- Outil de charg
  - Effectue ur

Adaptation aux contraintes de déploiement de la plate-forme cible



4. Cycle d… de CALICO

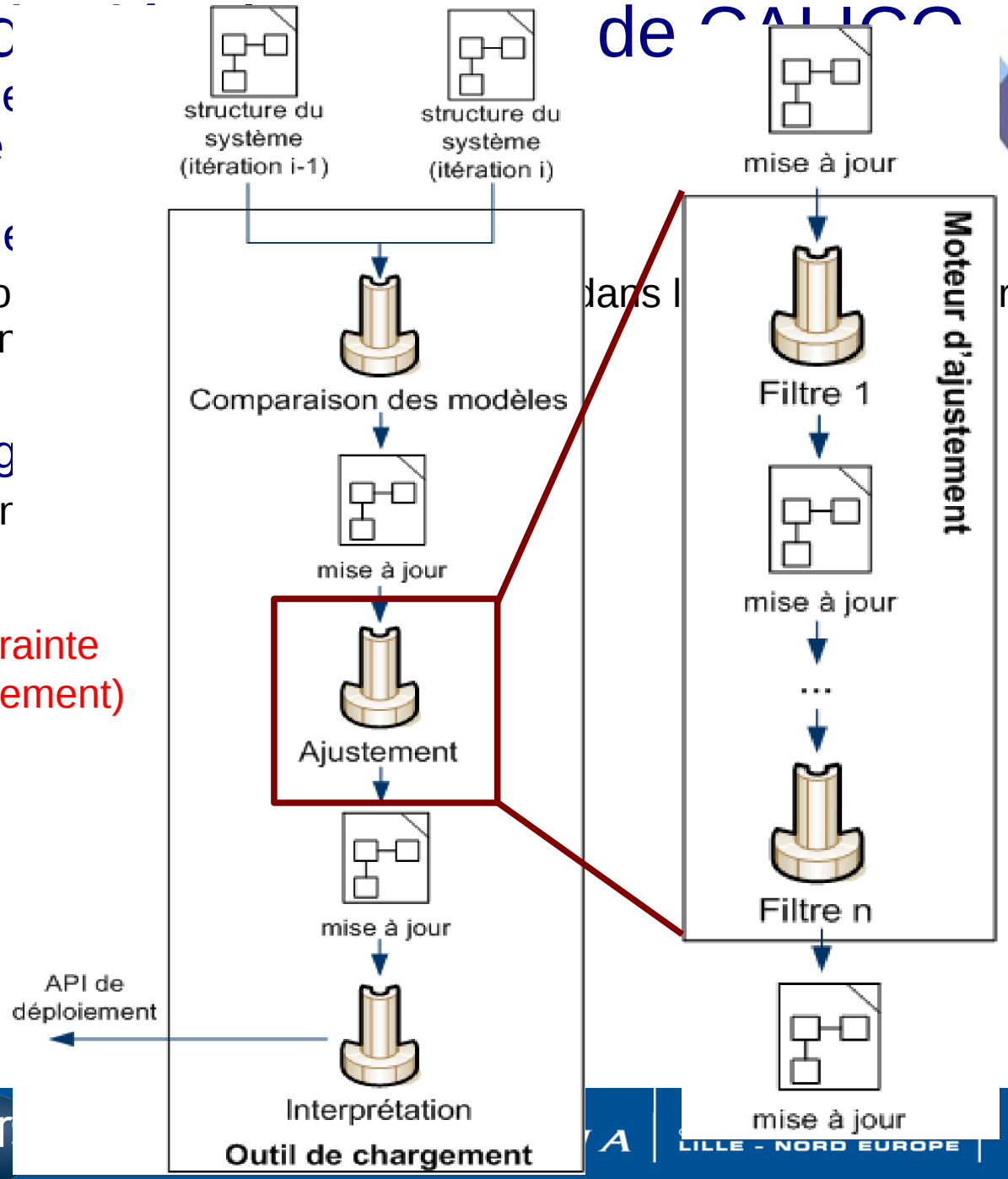
4.4. Déploieme…

4.4.1. Outil de…

administrateur système

- Problématique
  - Propagatio… dans l… rs d'exécutior…
- Outil de charg…
  - Effectue ur…

Filtre = gère une contrainte
(*e.g. :* ordre de déploiement)



4. Cycle d de CALICO

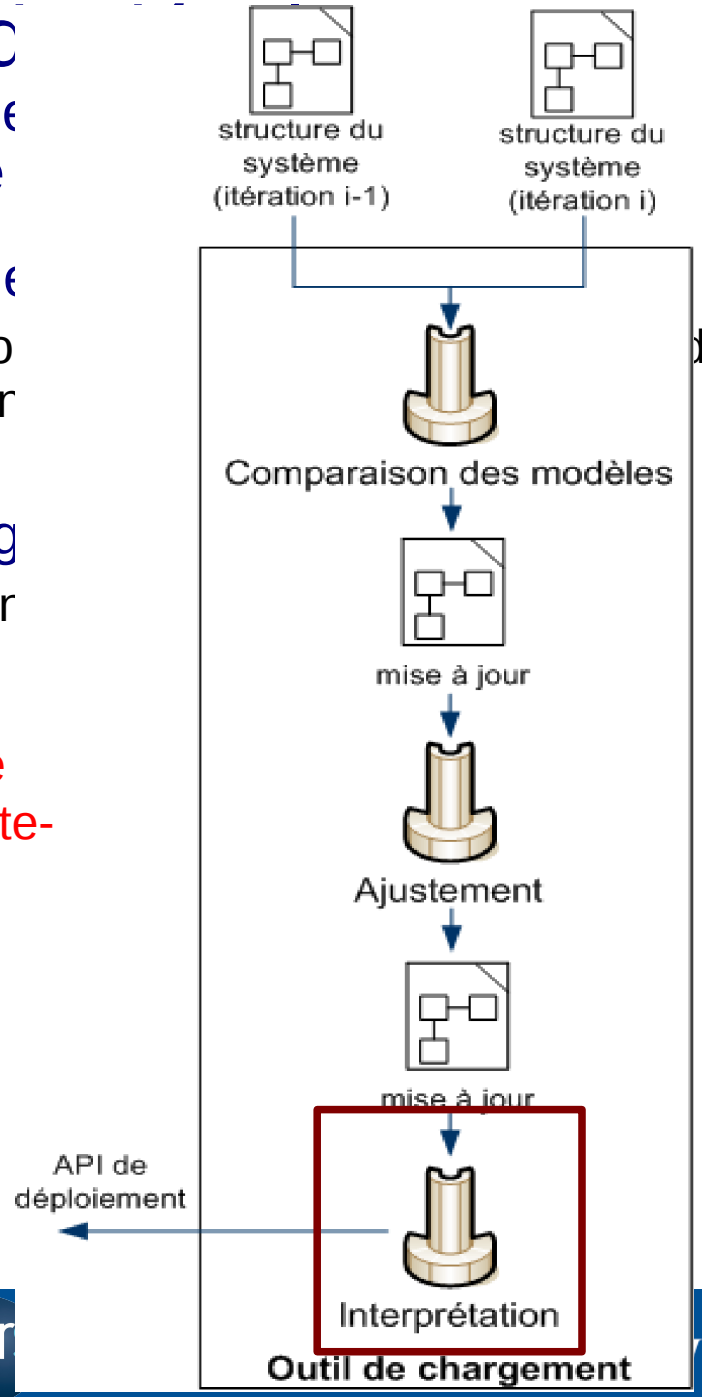
4.4. Déploieme

4.4.1. Outil de

administrateur système

- Problématique
  - Propagatio dans le système en cours d'exécutior
- Outil de charg
  - Effectue ur

Invocation de l'API de déploiement de la plate-forme cible



4. Cycle de développement de CALICO

4.4. Déploiement

4.4.1. Outil de chargement

- Problématique
  - Propagation des évolutions conceptuelles dans le système en cours d'exécution

- Outil de chargement
  - Effectue un déploiement incrémental
  - Compare les modèles entre deux itérations du cycle
  - Produit un modèle de mise à jour qui contient la séquence des opérations de construction du système (ajoute/suppime composants/connecteurs, ...)

Avantages

- Déploiement applicatif complètement automatisé
- Gestion uniforme des ajouts/supressions de fonctionnalités
- Flexibilité (s'adapte à la granularité des plates-formes)

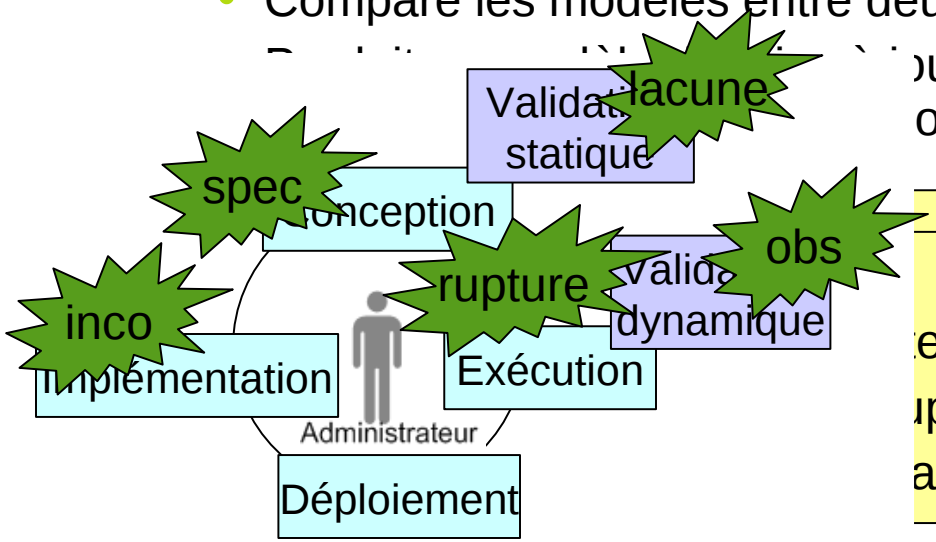
4. Cycle de développement de CALICO

4.4. Déploiement

4.4.1. Outil de chargement

administrateur système

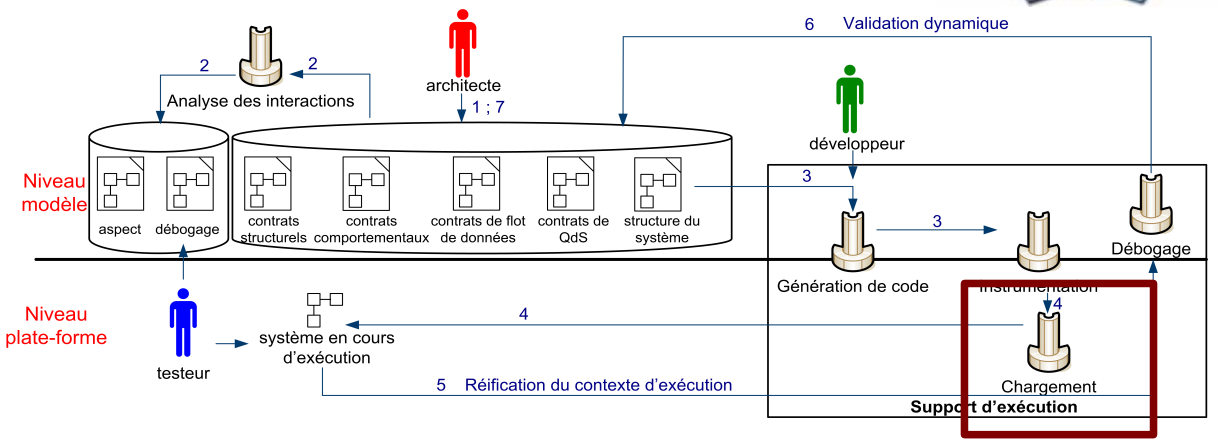
- Problématique
  - Propagation des évolutions conceptuelles dans le système en cours d'exécution
- Outil de chargement
  - Effectue un déploiement incrémental
  - Compare les modèles entre deux itérations du cycle
  - ...our qui contient la séquence des opérations
  - ...oute/suppime composants/connecteurs, ...)

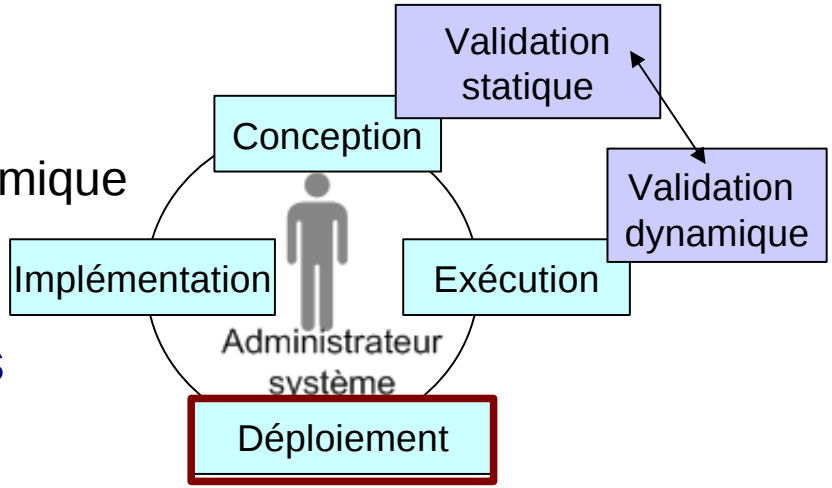


...ement automatisé
...pressions de fonctionnalités
...arité des plates-formes)

Plan

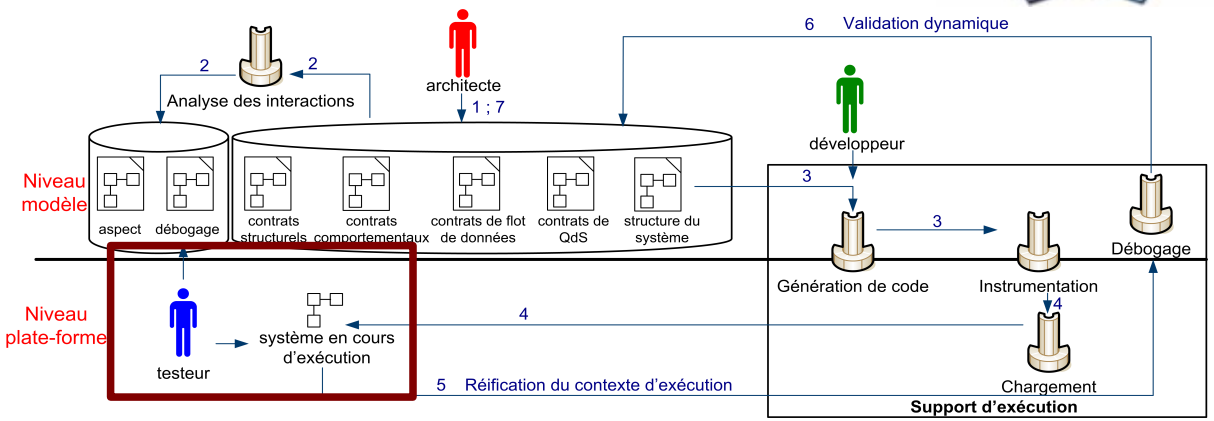
- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives

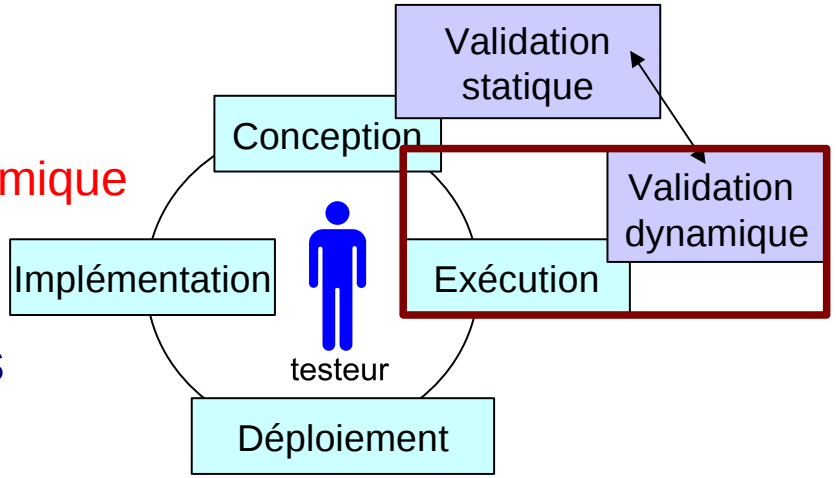


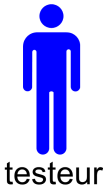


Plan

- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives



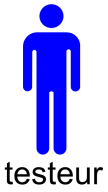




4. Cycle de développement de CALICO

4.5. Exécution / Validation dynamique

- Problématique
  - Exécution des scénarios d'utilisation du logiciel



4. Cycle de développement de CALICO

4.5. Exécution / Validation dynamique

- Problématique
  - Exécution des scénarios d'utilisation du logiciel
- Outil de débogage
  - Reçoit les données d'exécution émises par le logiciel instrumenté
  - Déclenche l'analyse dynamique appropriée

4. Cycle de développement de CALICO

4.5. Exécution / Validation dynamique

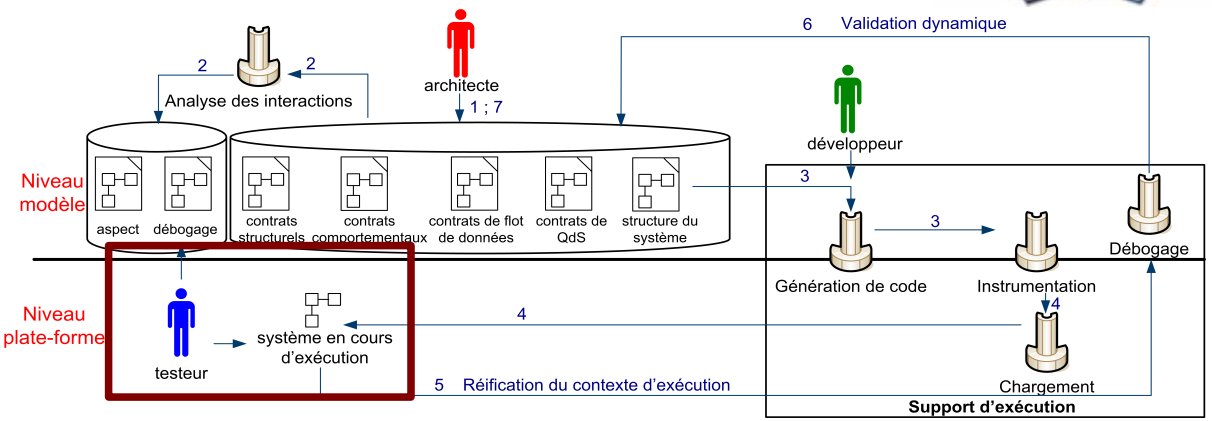
- Problématique
  - Exécution des scénarios d'utilisation du logiciel
- Outil de débogage
  - Reçoit les données d'exécution émises par le logiciel instrumenté
  - Déclenche l'analyse dynamique appropriée

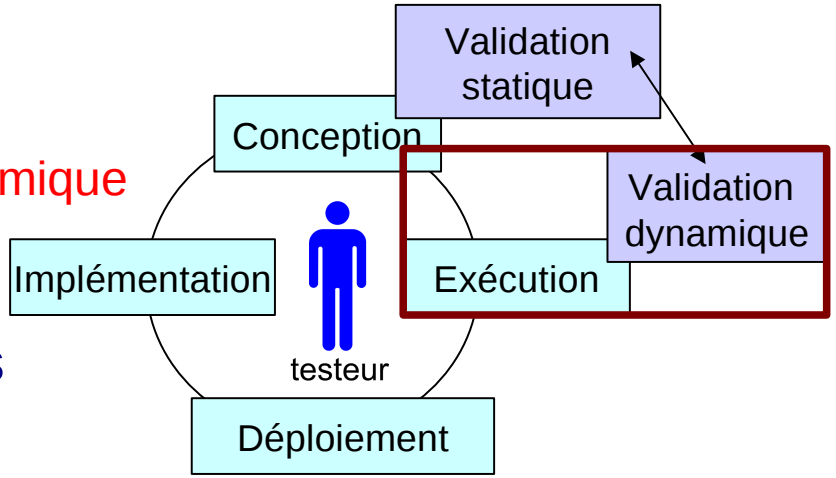
Avantages

- Externalisation des analyses dynamiques
- Indépendance des plates-formes
- Extensibilité

Plan

- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives





4. Cycle de développement de CALICO

4.6. Évolution

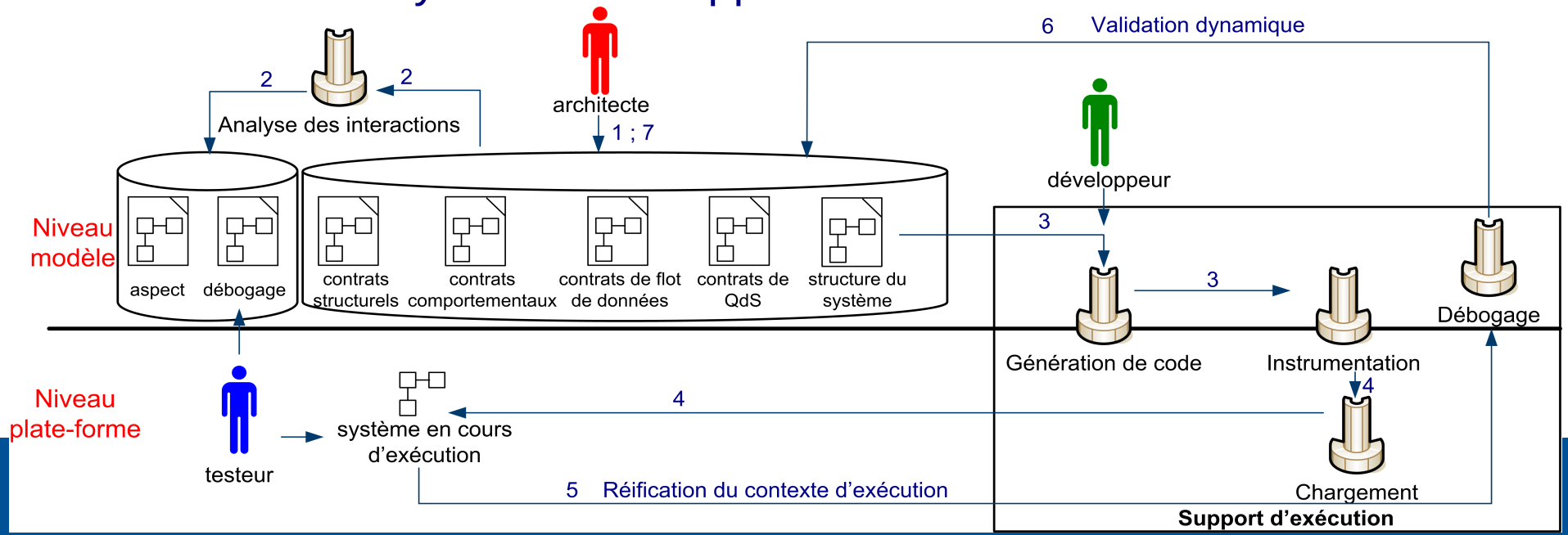
- Problématique
  - Développement agile
  - Prise en charge les demandes d'évolution
    - Nouveaux besoins des utilisateurs
    - Correction des erreurs détectées pendant l'étape de validation dynamique

4. Cycle de développement de CALICO

4.6. Évolution

- Problématique
  - Développement agile
  - Prise en charge les demandes d'évolution
    - Nouveaux besoins des utilisateurs
    - Correction des erreurs détectées pendant l'étape de validation dynamique

- Itération du cycle de développement



4. Cycle de développement de CALICO
4.6. Évolution

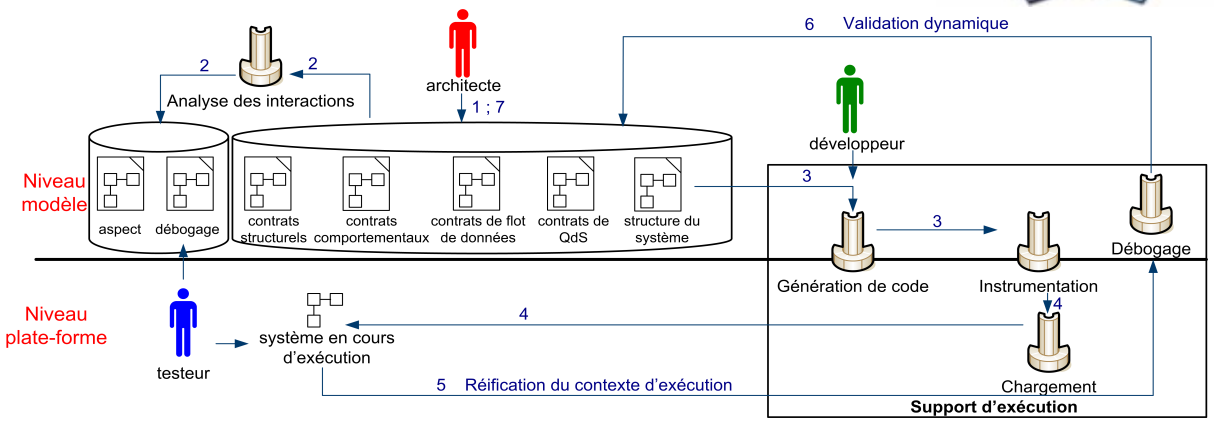
- Problématique
  - Développement agile
  - Prise en charge les demandes d'évolution
    - Nouveaux besoins des utilisateurs
    - Correction des erreurs détectées pendant l'étape de validation dynamique

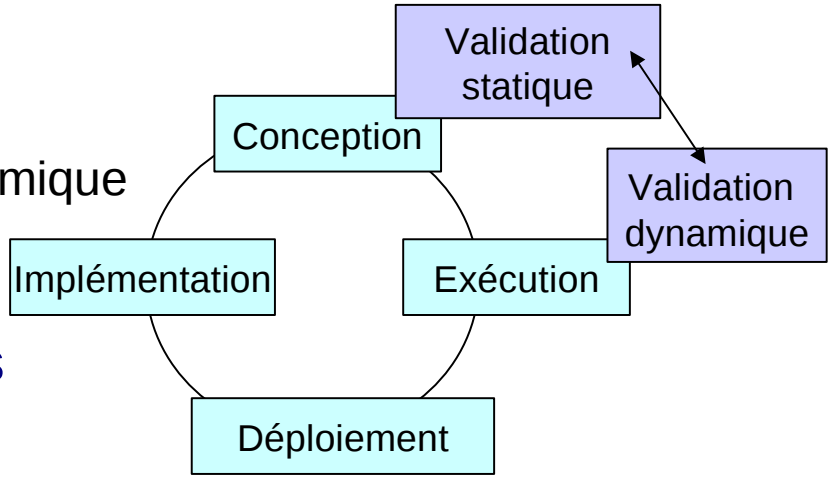
- Itération du cycle de développement
  - Vue conceptuelle qui reflète l'état du logiciel en cours d'exécution
  - Lien causal conservé entre les éléments de conception et d'exécution

Avantages
- Prise en charge uniforme de la conception / évolution
  - Conservation de la vue conceptuelle

Plan

- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives





5. Évaluation
5.1. Fonctionnalités

- Développement agile
  - Implémentation de CALICO (intégration à l'IDE Eclipse)
  - Guide les acteurs de manière transparente
    - Architecte : vue conceptuelle graphique (GMF)
    - Développeur : intégration de l'outillage Eclipse
    - Administrateur : déploiement applicatif automatique
    - Testeur : remontée des erreurs dans la vue Eclipse

5. Évaluation
5.1. Fonctionnalités

- Développement agile
  - Implémentation de CALICO (intégration à l'IDE Eclipse)
  - Guide les acteurs de manière transparente
    - Architecte : vue conceptuelle graphique (GMF)
    - Développeur : intégration de l'outillage Eclipse
    - Administrateur : déploiement applicatif automatique
    - Testeur : remontée des erreurs dans la vue Eclipse

- Fiabilisation des évolutions
  - Intégration d'outils d'analyse (e.g. : OCL, fractal BPC)
  - Intégration d'outil d'observation (e.g. : canevas WildCAT)

5. Évaluation
5.1. Fonctionnalités

- Développement agile
  - Implémentation de CALICO (intégration à l'IDE Eclipse)
  - Guide les acteurs de manière transparente
    - Architecte : vue conceptuelle graphique (GMF)
    - Développeur : intégration de l'outillage Eclipse
    - Administrateur : déploiement applicatif automatique
    - Testeur : remontée des erreurs dans la vue Eclipse

- Fiabilisation des évolutions
  - Intégration d'outils d'analyse (e.g. : OCL, fractal BPC)
  - Intégration d'outil d'observation (e.g. : canevas WildCAT)

- Multi plate-formes
  - Composant (Fractal, OpenCCM, OpenCOM), Service (GlassFish), Mixte (FraSCAti)

5. Évaluation
5.2. Passage à l'échelle

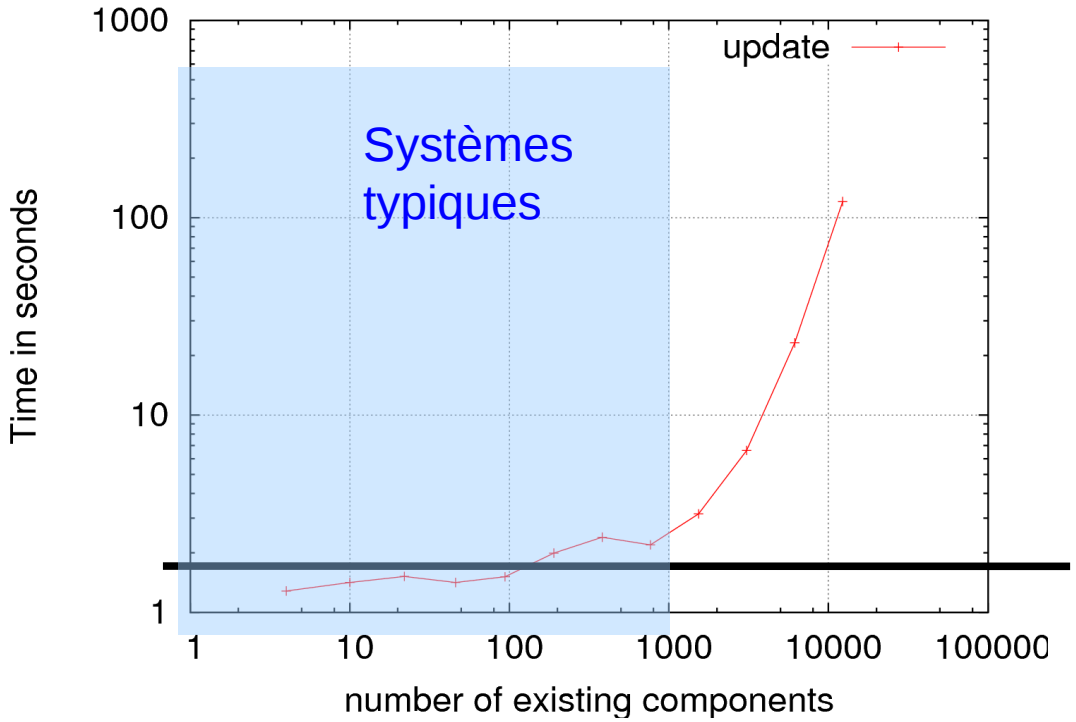
- Environnement des tests
  - Portable Core 2 Duo at 1,33Ghz
  - Java 1.6.0_11
  - Plate-forme à composants Fractal

- Test des limites de CALICO
  - Architecture hiérarchique très grande (1 - 100 000 composants)

- Majorité des applications à base de composants
  - < 1 000 composants
  - Limite des plates-formes à composants
    - SCA Apache Tuscany : < 6 000 composants

5. Évaluation
5.3. Déploiement incrémental

Coût des modèles / comparaison des modèles :

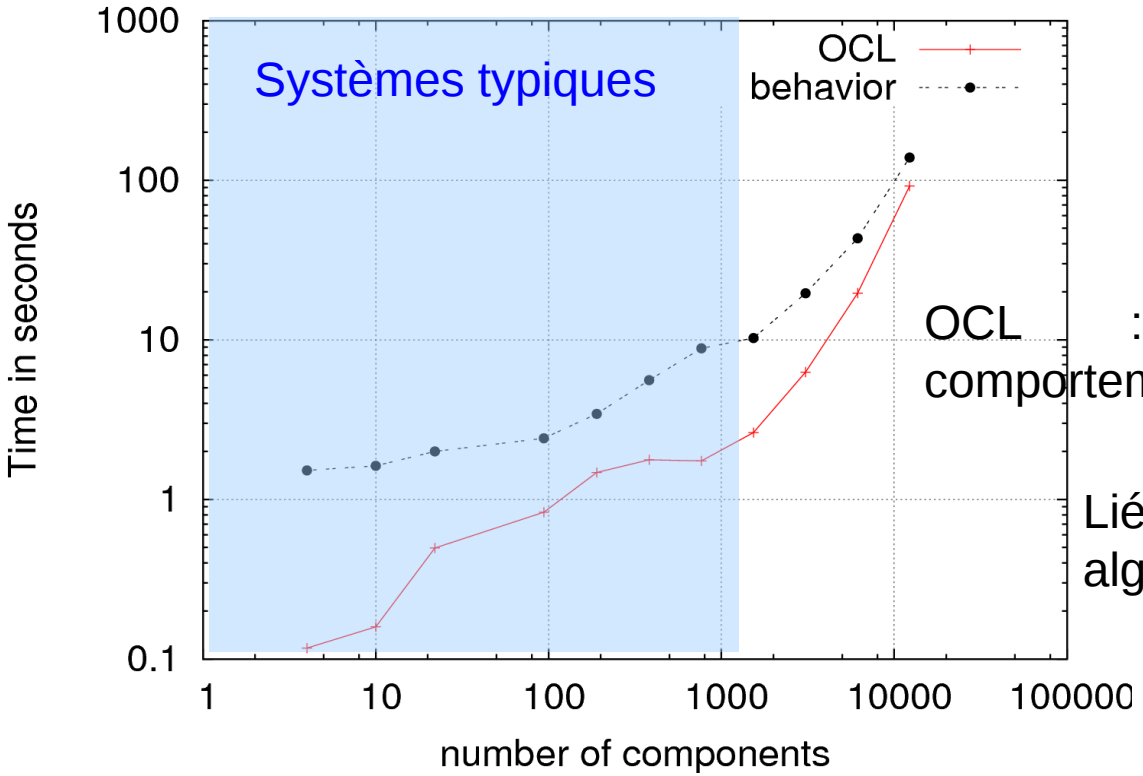


Avec Fractal ADL

CALICO : ~ 2s pour déployer un composant dans un assemblage
Fractal Native tool : ~ 2s pour déployer un composant

Même niveau de performance que l'outil natif

5. Évaluation
5.4. Analyses

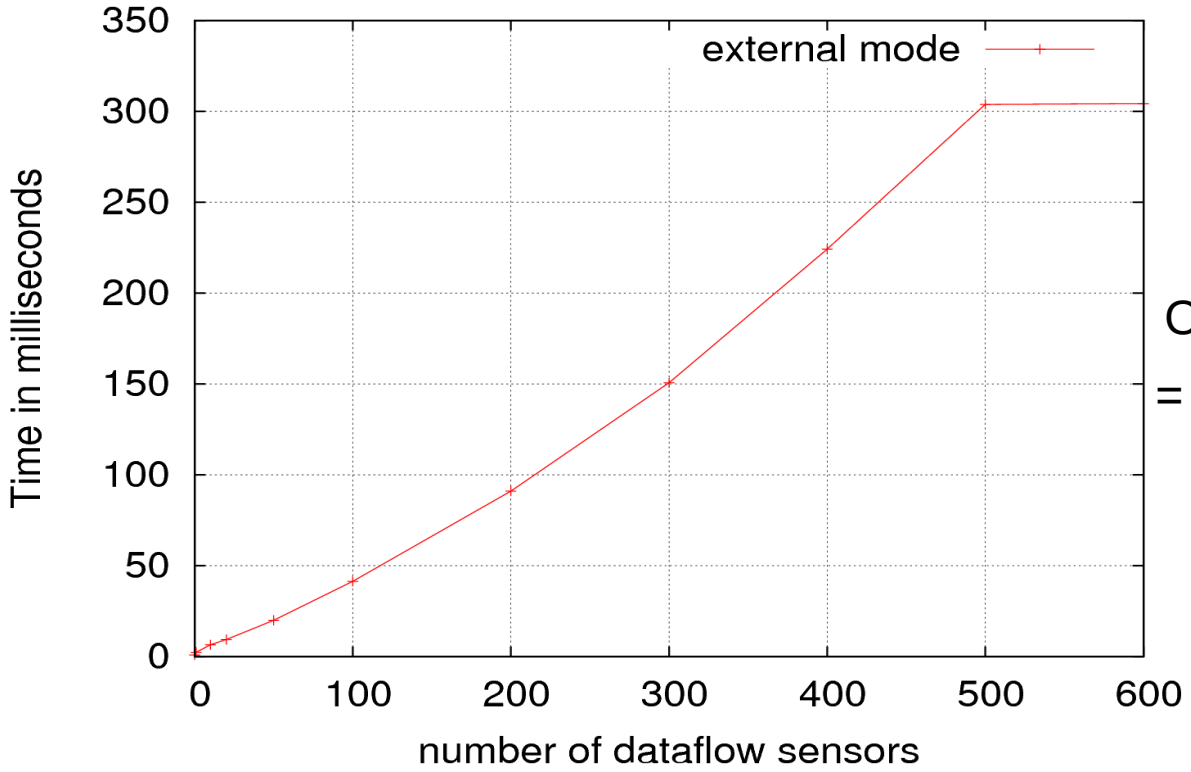


OCL : ~2s pour 1000 composants
comportement : ~10s pour 1000 composants

Liée à la complexité des algorithmes d'analyse

Architectes/testeurs peuvent activer/désactiver chaque outil d'analyse à n'importe quel moment

5. Évaluation
5.5. Observation des données



Observation 1 événement : ~1ms

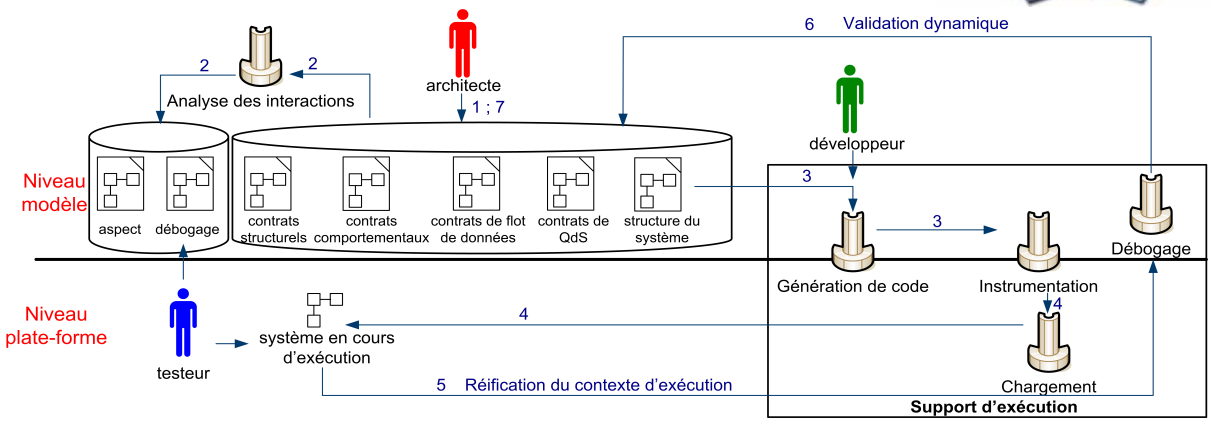
= Coût d'un appel distant

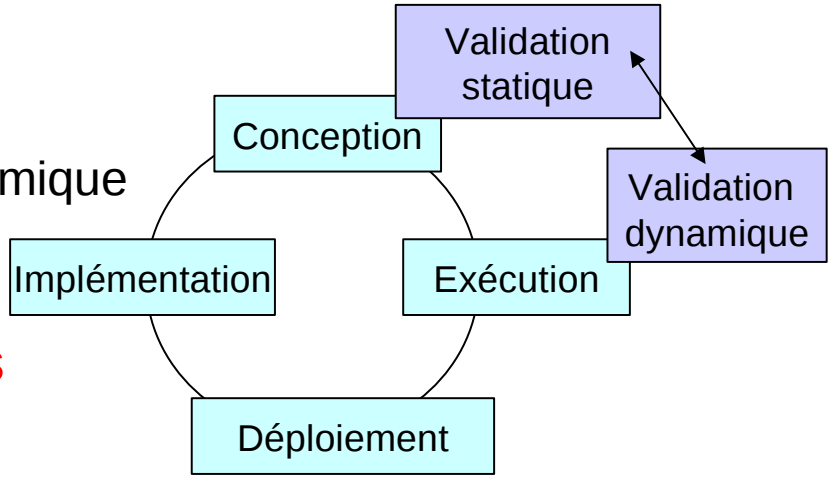
Avantages

Les performances de CALICO le rendent utilisable pour faire évoluer de manière fiable un logiciel

Plan

- Contexte
- Exemple
- Ma problématique
- CALICO
  - Conception
  - Validation statique
  - Implémentation
  - Déploiement
  - Exécution/Validation dynamique
- Evaluation
- Conclusion et perspectives





6. Conclusion et perspectives
6.1. Rappel du problème

- Agile
  - Découplage entre chaque étape du cycle
    - Introduction d'incohérences
    - Rupture dans le cycle de développement (pas itératif / incrémental)
  - Vue conceptuelle perdue pendant l'exécution du logiciel

- Fiabilisation des évolutions
  - Conception : manque de support de spécification
  - Validation statique : manque d'intégration d'outil d'analyse
  - Validation dynamique : mise en oeuvre manuelle

- Multi plates-formes
  - Support fortement couplé avec la plate-forme cible

6. Conclusion et perspectives
6.2. Bilan

- Agile
  - Evolution des logiciels à composants et orientés services
  - Cycle de développement itératif et incrémental en 6 étapes
  - Vue conceptuelle synchronisée avec le logiciel en cours d'exécution

- Evolution fiable
  - Large palette de spécifications (structure, comportementent, données, QdS)
  - Cadre fédérateur pour intégrer des outils d'analyse statique/dynamique
  - Prise en compte des interactions partiellement compatibles

- Multi plates-formes
  - Indépendant des plates-formes
    - Fractal, OpenCOM, OpenCCM, WebService, FraSCAti
  - Extensible (plates-formes, sondes, analyses)

http://calico.gforge.inria.fr

6. Conclusion et perspectives
6.3. Limitations

- Conception
  - Services déjà déployés
  - Annuaire
- Implémentation
  - Limitation à la conformance structurelle
  - Modification du code technique
- Déploiement
  - Non distribué
  - Non transactionnel

6. Conclusion et perspectives
6.4. Perspectives

- Extension aux applications autonomes
  - Permettre la validation des règles d'adaptation
  - Gérer le matériel contraint
  - Gérer la distribution

- Extension au domaine d'application
  - Ajout d'un niveau de modélisation métier
    - Permettre l'intégration de métamodèles métier existants
    - Permettre l'intégration d'outils d'analyse du métier
  - Définir un mécanisme de co évolution entre le modèle métier et les modèles de CALICO (de structure et de propriétés applicatives)

Conférence Internationales

A Model-Based Framework to Design and Debug Safe Component-Based Autonomic Systems. Guillaume Waignier, Anne-Francoise Le Meur and Laurence Duchien. In *Proceedings of the 5th International Conference on the Quality of Software-Architectures* (QoSA 2009), Pennsylvania, USA, jun 2009.

Architectural Specification and Static Analyses of Contractual Application Properties. Guillaume Waignier, Anne-Françoise Le Meur and Laurence Duchien. *In Proceedings of the 4th International Conference on the Quality of Software-Architectures* (QoSA 2008), Karlsruhe (TH), Germany, oct 2008.

A Model-Based Framework for Statically and Dynamically Checking Component Interactions. Guillaume Waignier, Prawee Sriplakich, Anne-Françoise Le Meur and Laurence Duchien. *In Proceedings of the ACM/IEEE 11th International Conference on Model-Driven Engineering Languages and Systems* (MODELS 2008), Toulouse, France, oct 2008.

Workshop Internationaux

A Framework for Bridging the Gap Between Design and Runtime Debugging of Component-Based Applications. Guillaume Waignier, Prawee Sriplakich, Anne-Françoise Le Meur and Laurence Duchien. *In Proceedings of the 3rd International Workshop on Models@runtime* 2008, Toulouse, France, oct 2008.

Enabling Dynamic Co-Evolution of Models and Runtime Applications. Prawee Sriplakich, Guillaume Waignier and Anne-Françoise Le Meur. *In Proceedings of the 1st IEEE International Workshop on Model-Driven Development of Autonomic Systems* (MDDAS 2008), pages 1116 - 1121, Turku, Finland, jul 2008.

Démonstration

A Framework for Agile Development of Component-Based Applications. Guillaume Waignier, Estéban Duguepéroux, Anne-Francoise Le Meur and Laurence Duchien. In *the 8th BElgian-NEtherlands software eVOLution seminar* (BENEVOL 2009), Louvain, Belgium, dec 2009.

Poster

CALICO : a Component Assembly Interaction Control Framework. Guillaume Waignier, Anne-Francoise Le Meur and Laurence Duchien. In *Poster au journées nationales du GDR GPL*, Toulouse, France, jan 2009.

Merci de votre attention

Questions?