Gestion de composants partiellement disponibles

- Romain FONTUGNE, magistère (2006)

Plan

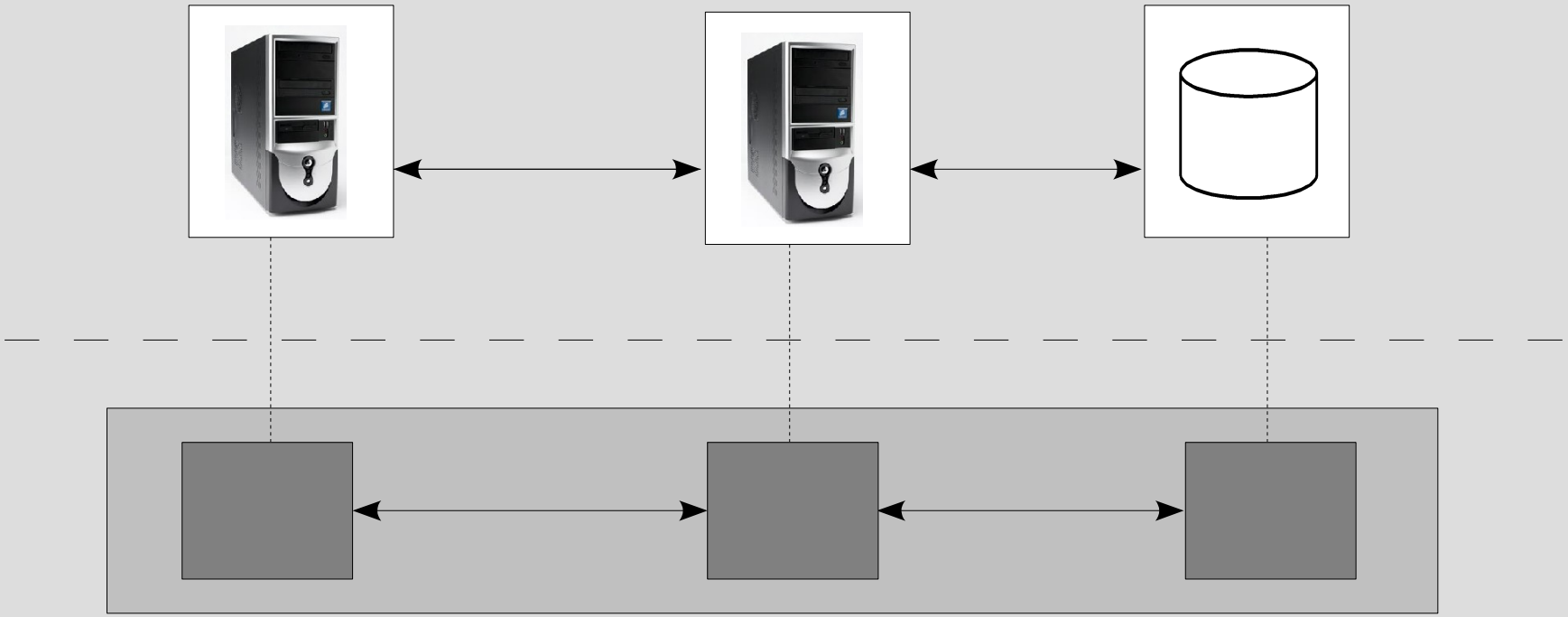
- Contexte de travail
- Le modèle Fractal
- Julia
- Problématique
- Etat de l'art
- Identification des problèmes
- Solution proposée

Contexte de travail

- Application réparties sur réseaux :
  - Hétérogènes
  - Nature dynamique
  - Grande taille
- Administration autonome d'applications complexes

Contexte de travail

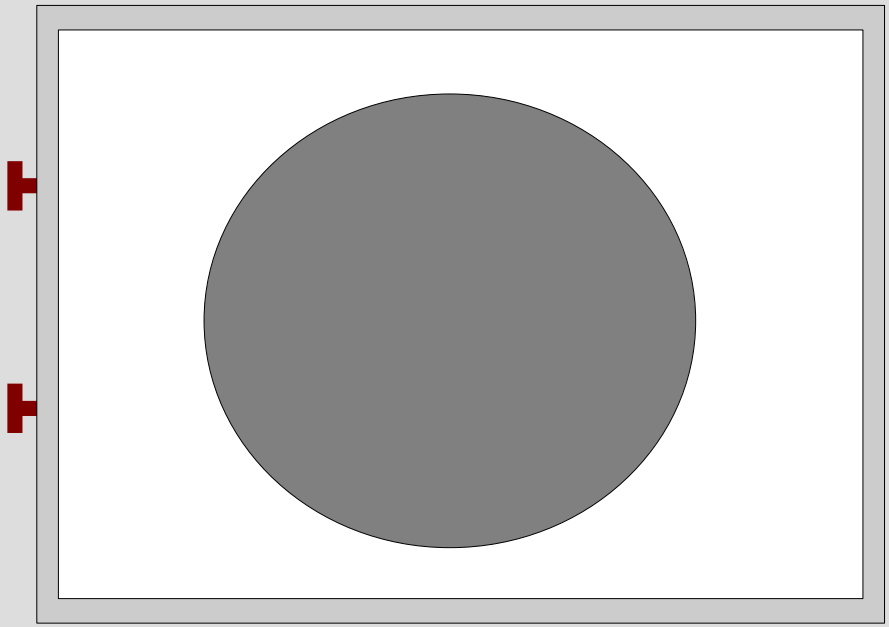
- Application répartie



- Composants correspondant

Objet / Composant

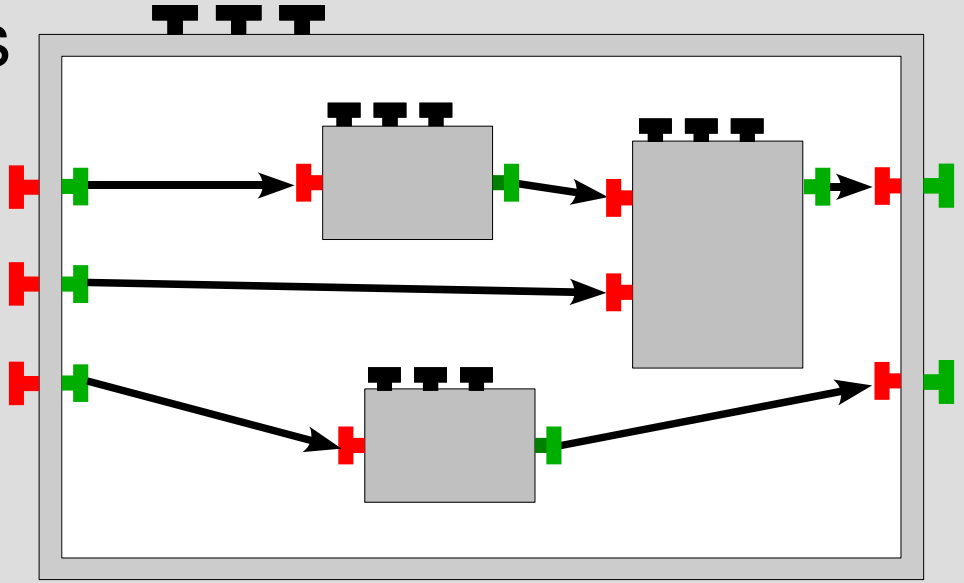
- Objet :
  - Interfaces fonctionnelles



Inconvénient : masquage des dépendances

Objet / Composant

- Composant :
  - Interfaces fonctionnelles
    - Service requis
    - Service fournit



  - Interfaces de contrôles
    - Gestion du contenu, des liens, ...
  - Architecture explicite

Fractal

- Spécification de modèle de composant :
  - Composant récursif
  - Composant réflexif
- Flexible et extensible
- Implémentation de Fractal : Julia
  - Composants répartis

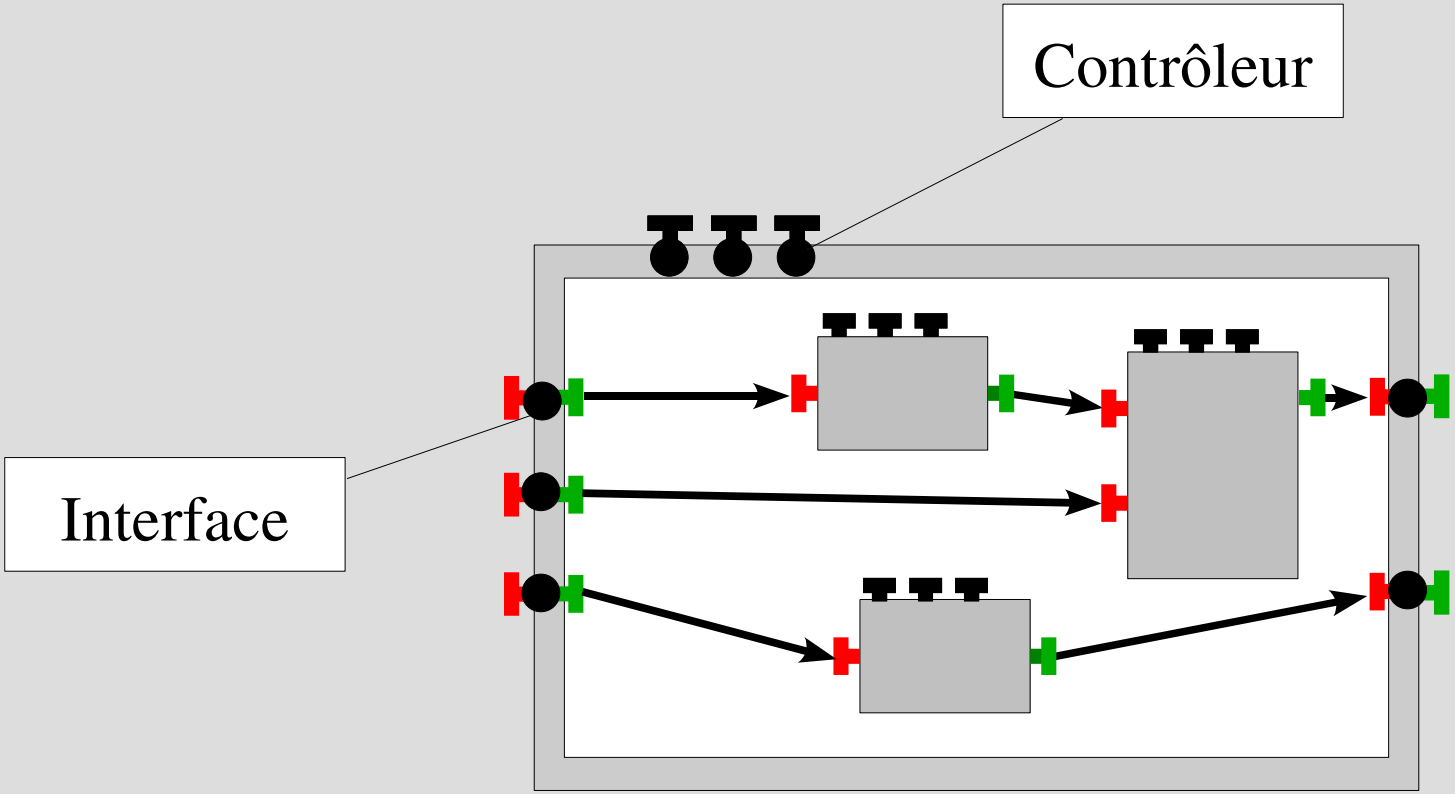
Contrôleurs Fractal

- **BindingController** : bindFc, unbindFc
- **ContentController** : addFcSubComponent, getFcInternalInterface, getFcSubComponent, removeFcComponent
- **LifeCycleController** : startFc, stopFc
- **AttributeController, NameController, SuperController**

Julia

- Implémentation Java :

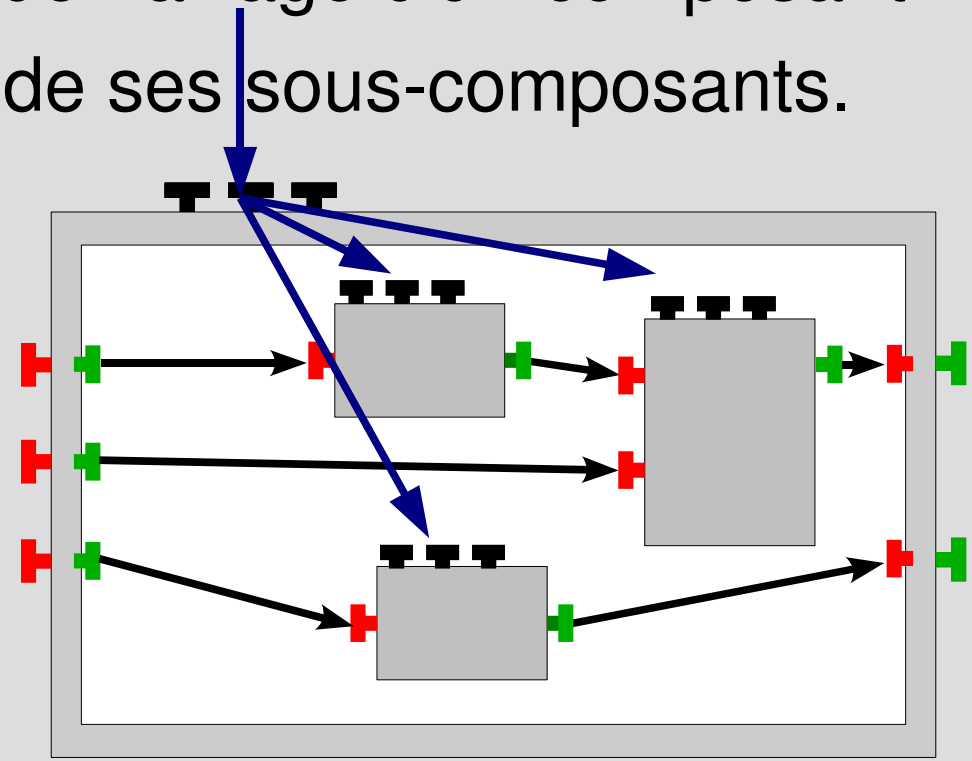
  Un composant = Plusieurs objets



Julia

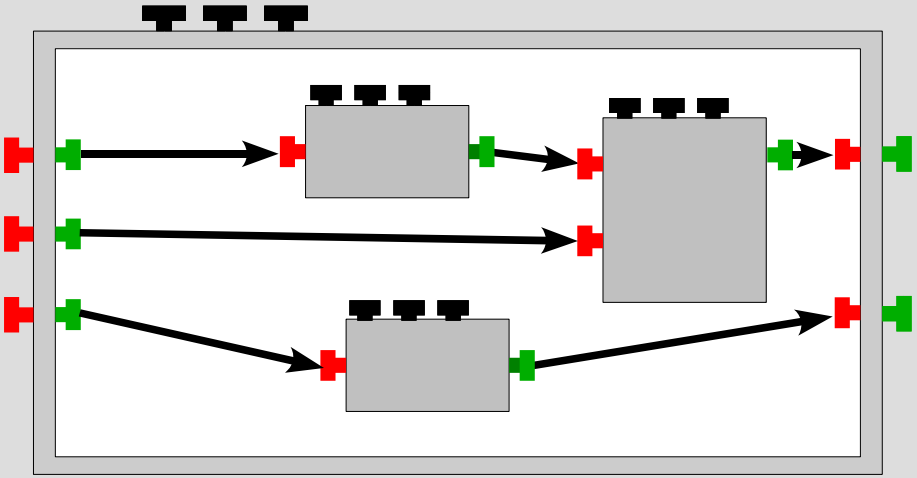
- Lien entre un contrôleur et d'autres composants :

  Exemple : Le démarrage d'un composant implique le démarrage de ses sous-composants.



Problématique générale

- Objectif : Administration d'une application répartie complexe
- Principe : Abstraction de l'application avec un modèle à composant
  - Maintenir la disponibilité de l'application en cas de panne



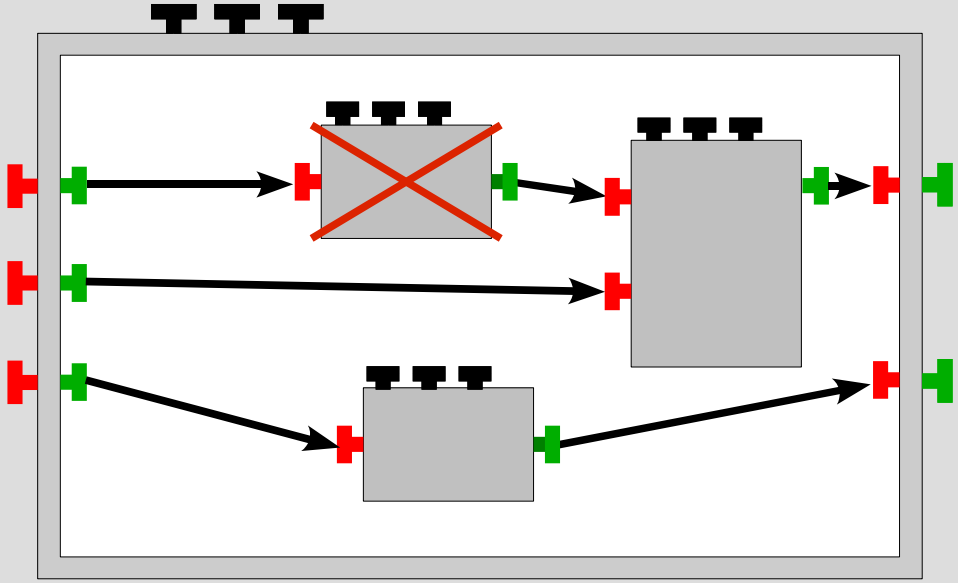
Problématique générale

- Gestion de pannes :
  - Arrêt du composant englobant la panne
  - On retire le composant défaillant
  - On alloue des ressources pour y installer un nouveau composant
  - On ajoute ce composant à l'application
  - On redémarre le composant englobant

Ne fonctionne pas si un sous-composant est défaillant !

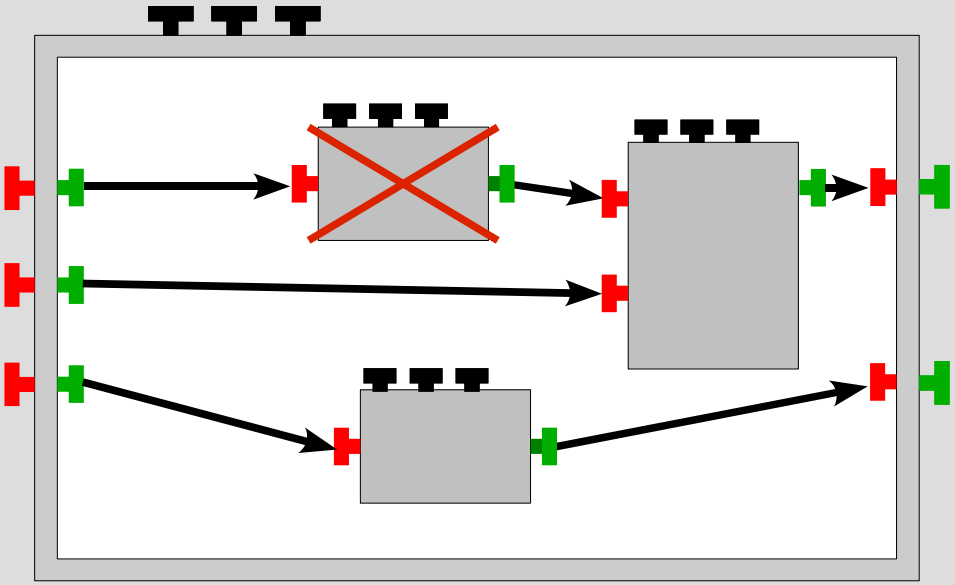
Problématique du stage

- Objectif du stage :
  - Ajouter à Julia l'infrastructure nécessaire pour manipuler une structure à composant partiellement défaillante



Etat de l'art

- Université Bretagne Sud (Valoria)
- Fractal
- Interface fonctionnelle



Plan de travail

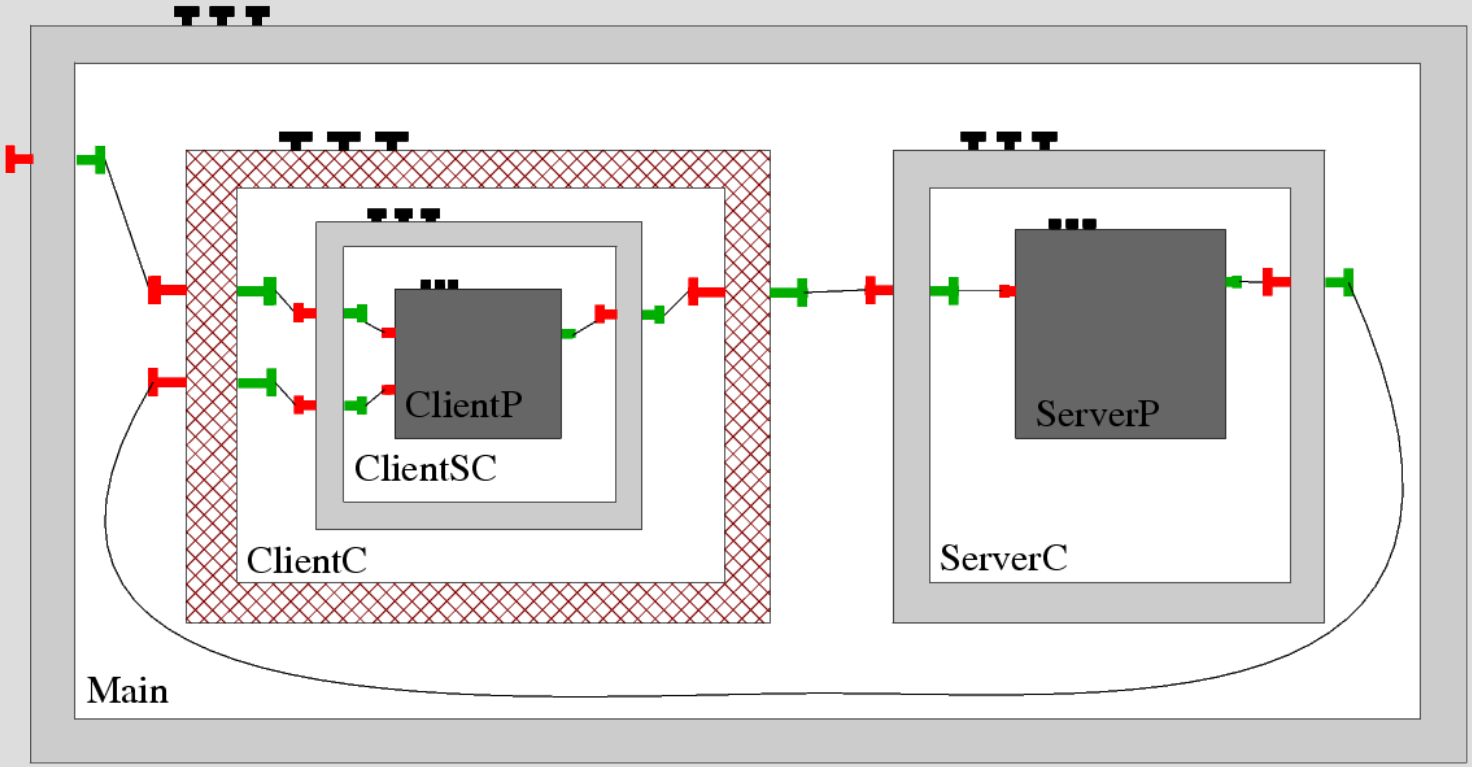
- Prise en main de Julia
- Identifier les interfaces ne fonctionnant pas dans un composant partiellement disponible
- Proposer un nouvelle sémantique de contrôle
- Caractériser l'origine des problèmes rencontrés
- Proposer une solution et l'implanter

Nouvelle sémantique

- Garder un historique des opérations
- Mode dégradé :
  - Effectuer l'opération en ignorant les composants indisponibles
  - Annuler l'opération de contrôle

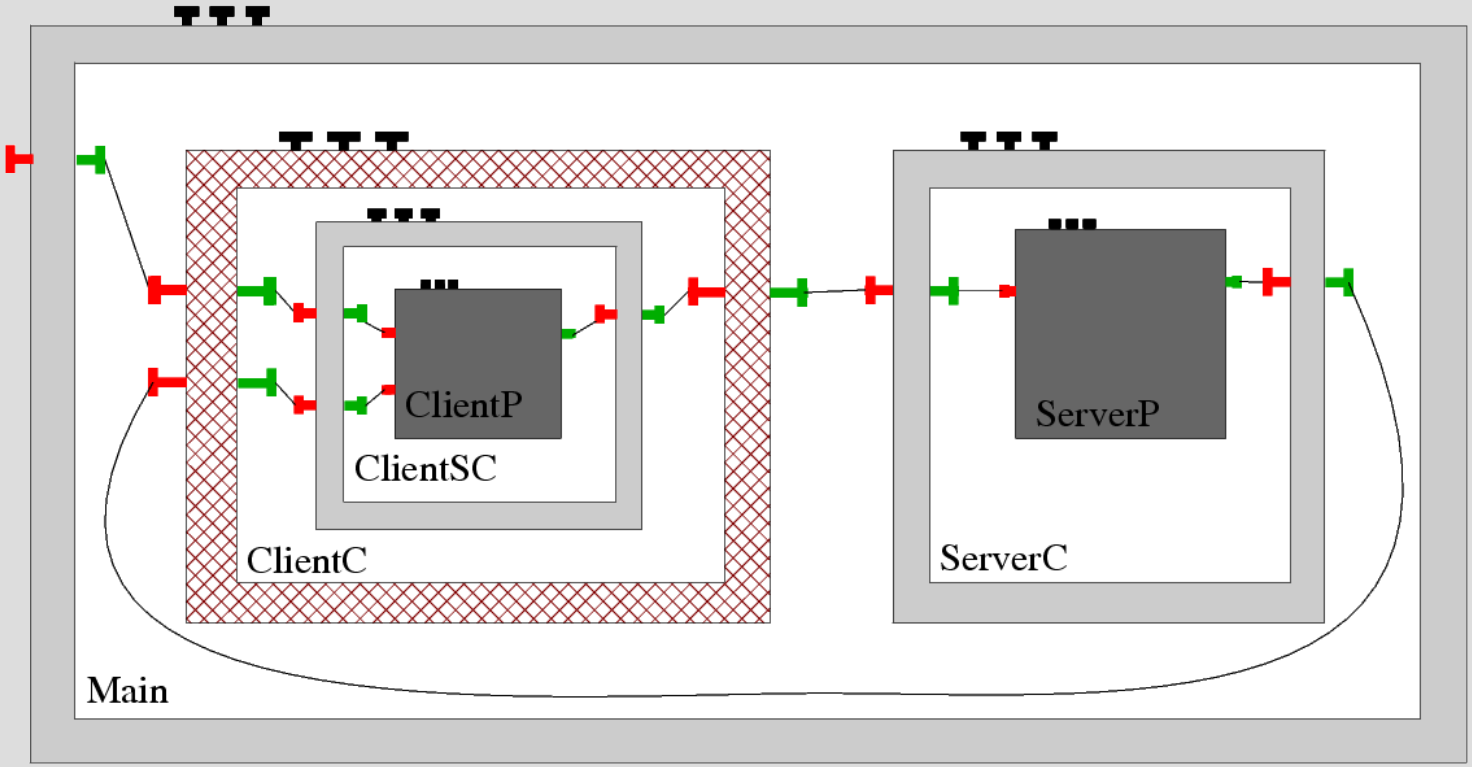
Quelles interfaces posent problème?

- Test sur une application simple :



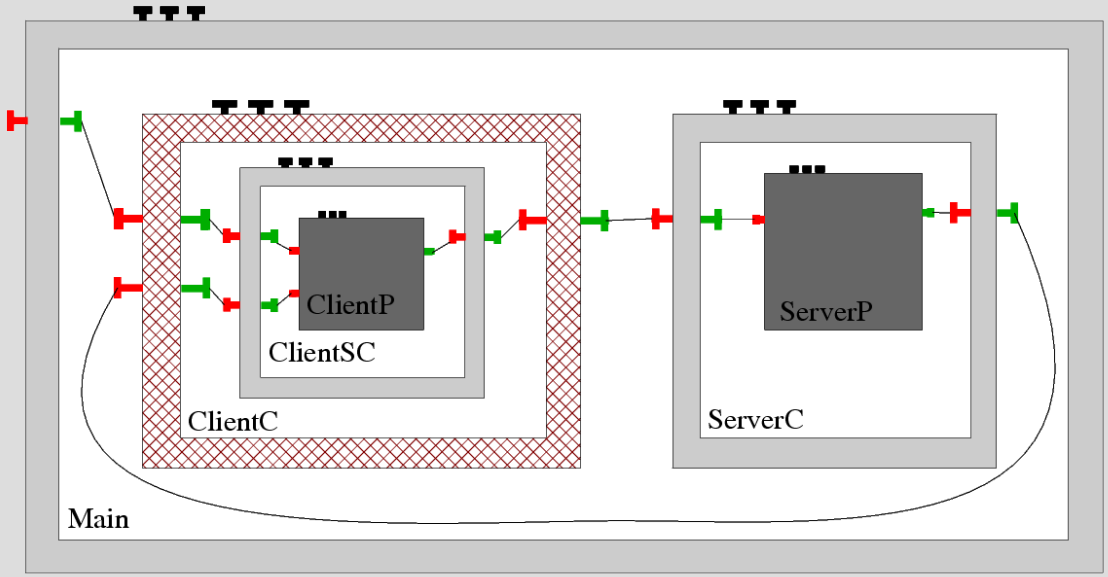
Quelles interfaces posent problème?

- BindingController, ContentController, LifeCycleController.



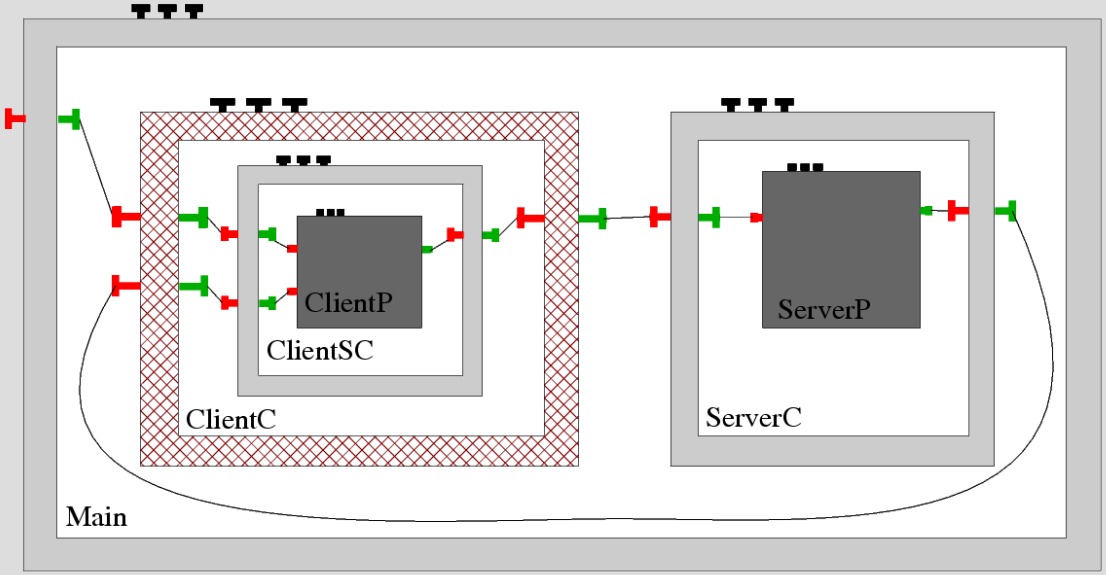
Pourquoi le BindingController pose problème?

- Création d'un lien vers un composant défaillant : impossible de vérifier la compatibilité des deux interfaces à lier



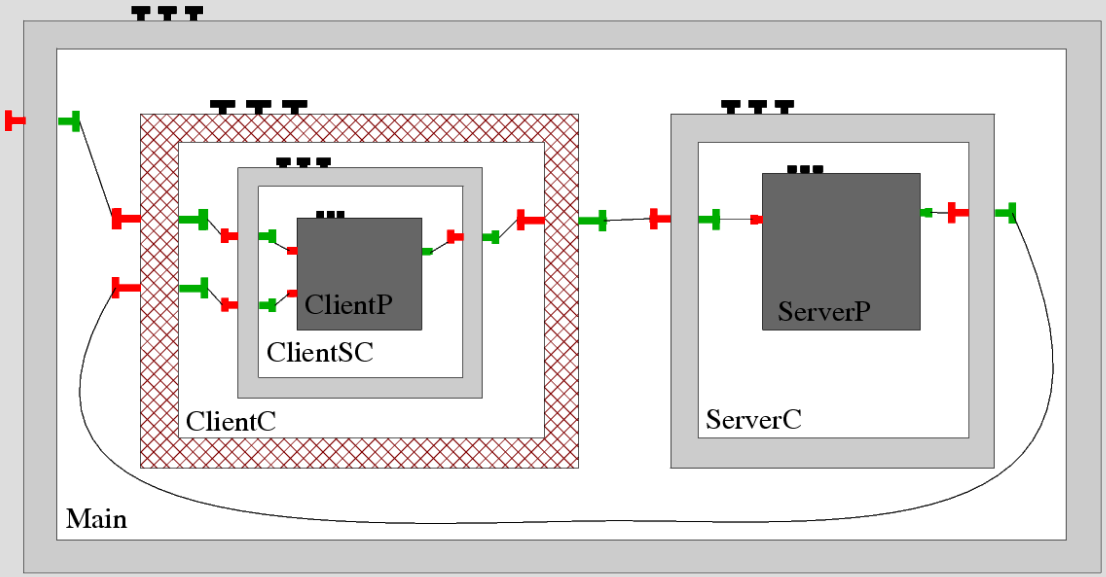
Pourquoi le ContentController pose problème?

- Retirer un sous-composant :
  - Difficulté à trouver le sous-composant à retirer (comparer deux composants)
  - Difficulté à vérifier les liens du sous-composant



Pourquoi le LifeCycleController pose problème?

- Stopper un composant : (idem pour démarrer)
  - Impossible de stopper les sous-composants
  - Impossible de stopper les composants affectés



Quelles interfaces posent problème?

- Différents problèmes :
  - Problème d'implantation (removeFcSubComponent)
  - Problème de sémantique
    - Opération impossible (startFc, bindFc)
    - Opération pas adaptée (stopFc)

Solution proposée

- Principe :
  - L'opération peut être effectuée en ignorant les composants défaillants
  - Sinon l'opération est annulée

Solution proposée

- On ignore les composants défaillants :
  - ContentController (removeFcSubComponent)
  - LifeCycleController (stopFc)
- On annule l'opération de contrôle et on signale que l'opération est annulée :
  - BindingController (bindFc)
  - LifeCycleController (startFc)

Implantation

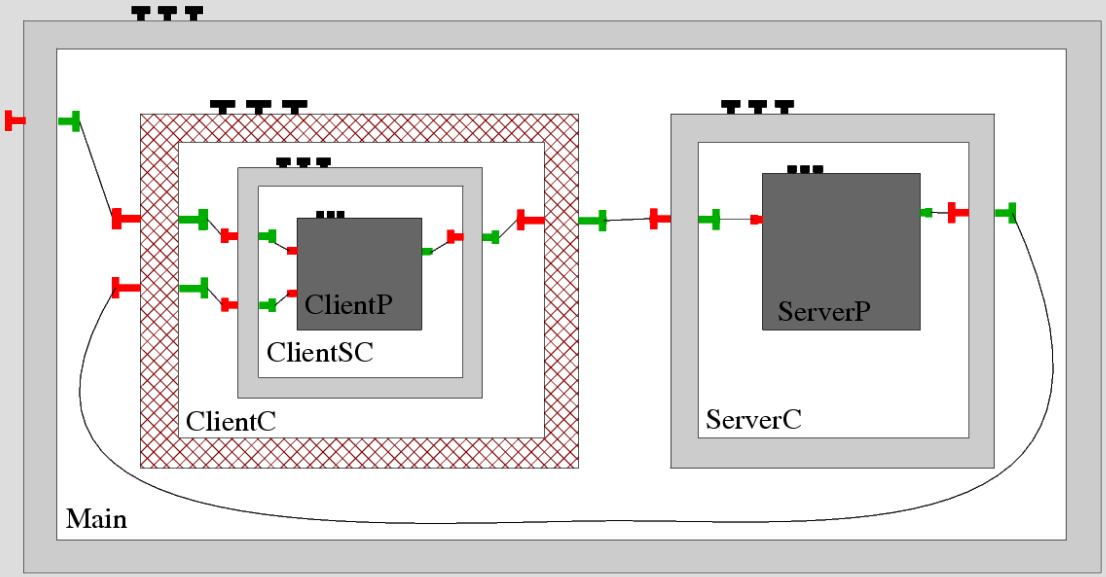
- On a besoin de pouvoir comparer deux composants même si l'un d'eux est indisponible. Et connaître l'état des composants.
  - Ajout d'un contrôleur :
    - Vérification de la disponibilité de composants
    - Comparaison de composants
      - Composants distants ou non
      - Composants disponible ou non

Implantation

- Modification des contrôleurs existants
  - Ignorer les composants défaillants
  - Annuler l'opération de contrôle
  - Modification du code pour les problèmes d'implantation

Exemples

- Démarrer le composant Main => annuler
- Retirer le composant indisponible => effectuer
- Stopper le composant Main => effectuer



Perspectives

- Intégration dans un gestionnaire de pannes
- Considérer des composants indisponibles pendant un temps fini

Quelques liens...

- INRIA : http://www.inrialpes.fr
- SARDES : http://sardes.inrialpes.fr
- JADE : http://sardes.inrialpes.fr/research/jade/
- ObjectWeb : http://www.objectweb.org
- Fractal : http://fractal.objectweb.org
- Distribution d'un composant hiérarchique dans un environnement partiellement connecté
  Didier Hoareau - Yves Mahéo (Valoria)
  http://www-valoria.univ-ubs.fr/CASA/Publications/

Démonstration