ConFract

Une approche pour contractualiser des composants hierarchiques

V. Poirriez

LAMIH UMR UVHC-CNRS 8530

Avril 2008 réunion TACOS

Sommaire

Liminaire

Les sources de cette présentation sont

- les différents articles de P. Collet et al.
- un dialogue avec Alain Ozanne lors d'un séminaire au LAMIH

Équipes

- Laboratoire I3S CNRS - Sophia Antipolis
- France Télécom R & D

Personnes

- Philippe Collet : **Enseignant-Chercheur I3S**
- Nicolas Rivierre : **Chercheur FTel**
- Alain Ozanne : **thèse 2007**
- Hervé Chang : **thèse 2007**

Sous Licence

Nécessite l'obtention d'une licence auprès de France Telécom

Historique basé sur les publications

- Collet, Ozanne : "Spécification d'intégration du modèle de contrats dans la plateforme Fractal", France Télécom, Rapport de Contrat de Recherche Externe n° 422721832, septembre 2003.
- Vers la négociation de contrats dans les composants logiciels hiérarchiques. **RR-2004-33**
- Un système de contractualisation pour Fractal : Intégration et retours sur expérience. **RR-2005-01**
- Thèse A. Ozanne : **Interact : un modèle général de contrat pour la garantie des assemblages de composants et services** 11/2007`http://a.ozanne.free.fr/Interact%20v1.73.pdf`
- Thèse Hervé Chang :**Négociation de contrats dans les systèmes à composants logiciels hiérarchiques** Décembre 2007.

Publications d'Alain Ozanne I

- Collet, Malenfant, Ozanne, Rivierre : "Composite Contract Enforcement in Hierarchical Component Systems". In ETAPS 2007, 6th International Symposium on Software Composition (SC 2007). LNCS, Springer Verlag.
- Collet, Ozanne, Rivierre : "Towards a Versatile Contract Model to Organize Behavioral Specifications". In 33rd International Conference on Current Trends in Theory and Practice of Computer Science SOFSEM 2007, Harrachov (Czech Republic).
- Chang, Collet, Ozanne, Rivierre : "From components to autonomic elements using negotiable contracts". In 3rd International Conference on Autonomic and Trusted Computing (ATC'06).

Publications d'Alain Ozanne II

- Chang, Collet, Ozanne, Rivierre : "Some autonomic features of hierarchical components with negotiable contracts", ICAC 2006.
- Collet, Ozanne, Rivierre : "Enforcing Different Contracts in Component-based Hierarchical Systems", ETAPS 2006 (Software Composition Symposium).
- Collet, Ozanne, Rivierre : "On Contracting Behavioral Properties in Component-Based Systems", poster, ACM Symposium on Applied Computing 2006 (Software Engineering Track).
- Collet, Ozanne : "Un système de contractualisation pour Fractal : intégration et retour sur expérience", Journées Composants 2005.

Points de départ pour ConFract

- **Proposition**
  - appliquer l'approche de programmation par contrats au développement logiciel à base de composants hiérarchiques
- **Contrat** :
  - Spécification et vérification de propriétés, tout en associant à chaque entité **des responsabilités bien définies**
- **Objectifs**
  - définir un modèle de contractualisation pour Fractal,
  - d'abord simplifié :*langage d'assertions, contraintes fonctionnelles*
  - extensible

Le cahier des charges défini par P. Collet

- spécifications incrémentales, fonctionnelles ou non
- supporter à terme plusieurs formalismes
- spécifications ponctuelles ou compositionnelles
- distinguer les spécifications des contrats
  - *Les contrats suivent les reconfigurations dynamiques*
- permettre des traitements variés en cas de violation des contrats

Ensemble de participants

- **le garant** qui décrit la propriété logique,

  chaque disposition a un garant unique.

- **le bénéficiaire** qui veut se servir de la propriété vérifiée,

  Chaque disposition d'un contrat profite à un ensemble de bénéficiaires

- **le contributeur** qui représente toute entité indirectement responsable d'une partie de la spécification et qui peut agir pour valider la spécification.

  chaque disposition d'un contrat utilise des contributeurs.

Classification des contrats

- Contrat de bibliothèque = contrat *"objet"* (au niveau des classes)
- **Contrat d'interface** contrat d'une connexion point à point entre une interface requise et une interface fournie. Semblables aux contrats dans les objets.
- **Contrat de composition** contrat défini sur la membrane d'un composant Fractal
  - Contrat de composition externe : plusieurs interfaces externes
    exprime les règles d'utilisation et de comportement externe du composant.
  - Contrat de composition interne : interfaces internes + sous-composants
    exprimer les règles d'assemblage et de comportement interne d'implantation.

Sémantique des contrats

- **Contrat d'interface**
  - Évaluation classique des pre et post conditions, ainsi que des invariants.
- **Contrat de composition**
  - Conjonction de chaque type de spécification
  - Le composite (l'assembleur) est responsable de tout problème survenant à son niveau.

Un langage d'expression de contrats : CCL-J

- **CCL-J**= *Component Constraint Language for Java*. Inspiré d'OCL
- **pre**, **post** et **inv** classiques
- **rely** les propriétés sensées être vraies durant l'appel de la méthode
- **guarantee** état garanti pendant l'exécution de la méthode.
- la portée est associée aux interfaces (construction *context*), mais aussi aux composants (construction *on*)

Les contrats en résumé dans l'implantation actuelle

- composé de dispositions, qui contiennent chacune une spécification(une assertion en *CCL-J*), ainsi que le contexte temporel et spatial nécessaire à sa vérification.
- Les contrats sont alors vérifiés aux événements appropriés (vérification des préconditions à l'entrée de méthodes, . . .)
- et, par défaut, des exceptions sont déclenchées en cas de violation.

Travaux de Chang I

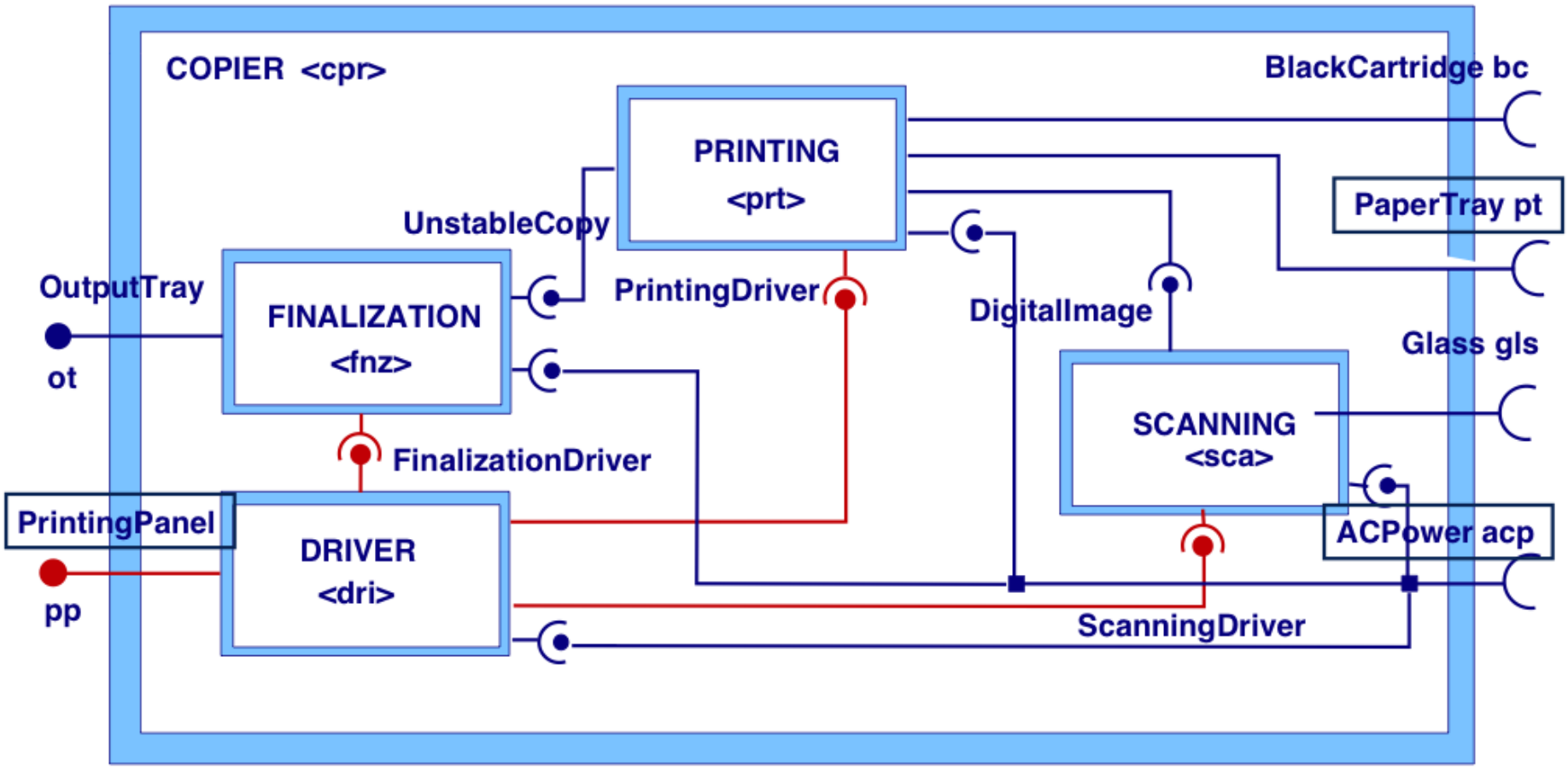
- Modèle de négociation de contrats prenant en compte la dynamicité des applications.
  - Politique par relâchement : relaxation de contraintes ou reconfiguration de paramètres pour les bénéficiaires.
  - Fonctionnement dégradés acceptables.
- Architecture de contrôle $\Rightarrow$ systèmes capables de *se* surveiller et d'ajuster leur fonctionnement.
- Politique *par effort :*
  - Demandant aux composants *garants* de se reconfigurer afin de garantir à nouveau les propriétés dont ils sont garants.

travaux de Chang II

`http://www.i3s.unice.fr/~chang/publications.php`

- Chang H., Collet P., « Fine-grained Contract Negotiation for Hierarchical Software Compo- nents », EUROMICRO-SEAA'2005, IEEE Computer, Portugal, 2005
- Hervé Chang and Philippe Collet. Eléments d'architecture pour la négociation de contrats extrafonctionnels. In Proceedings of 1ère Conférence francophone sur les Architectures Logicielles (CAL'06), pp. 151-167, France, September 2006, L'Objet Hermès.
- Hervé Chang and Philippe Collet. Compositional Patterns of Non-Functional Properties for Contract Negotiation. Journal of Software (JSW), Academy Publisher, 2(2), pp. 52-63, Août 2007.

Architecture interne d'un photocopieur



Interfaces

public est implicite

```
interface PaperTray {
   double trayWidth();       // inches
   double trayHeight();      // inches
   int capacity();          // max. nb. of sheets in paper tray
   Stack<Sheet> tray();     // the full paper tray
   void loadTray(Stack<Sheet> new);  // push new sheets onto the tray
   boolean isPaperJam();
}

interface PrintingPanel {
   float pageLength();
   boolean isPowerOn();     //  is copier on ?
   boolean isCopying();     // is copier currently copying ?
   void setPowerOn();       // connect ACPower
   Time copyDuration();     // effective duration of current copy (sec)
   void copy(int n );       // launch n copies of the original
}

public interface AcPower {
  boolean isConnected();  // is AC power connected?
  void connect();
  void disConnect();
}
```

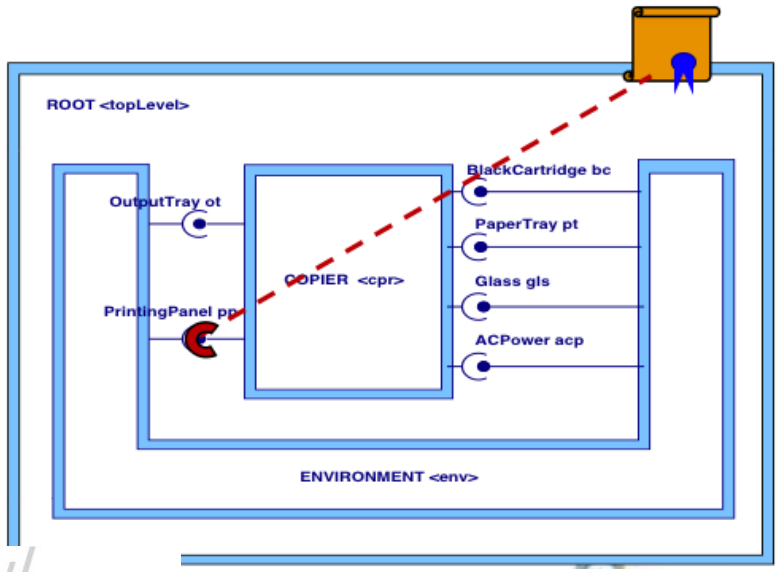
spécifications de propriétés

```
context void PrintingPanel.copy(int n)
  pre: 0 <= n <= 500;
  post: ! isCopying();
```

```
context PaperTray inv:
  maxNbSheets: tray.count <= capacity
  foreach i in 1 .. tray.count :
     tray.item(i).width <= width &&
     tray.item(i).height <= height
```

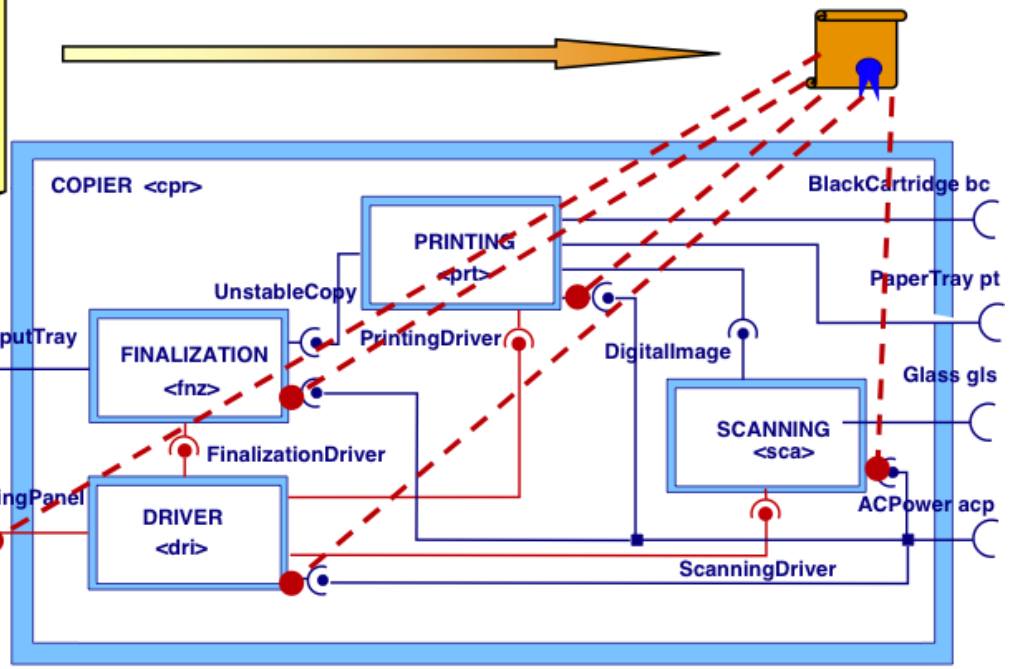
Contrat d'Interfaces

```
Interface contrat on <topLevel> {
Binding server = PrintingPanel <cpr>.pp,
        client = Main <env>.main
  Participants <cpr>, <env>
  Provisions
    context void copy(int n)
      pre: 0 <= n <= 500; // guarantor is <env>
      post: ! isCopying(); // guarantor is <cpr>
...
```



Spécification d'un composant

```
on <COPIER>
  context void pp.*(*) {
    pre: <*>.acp.isConnected();
```

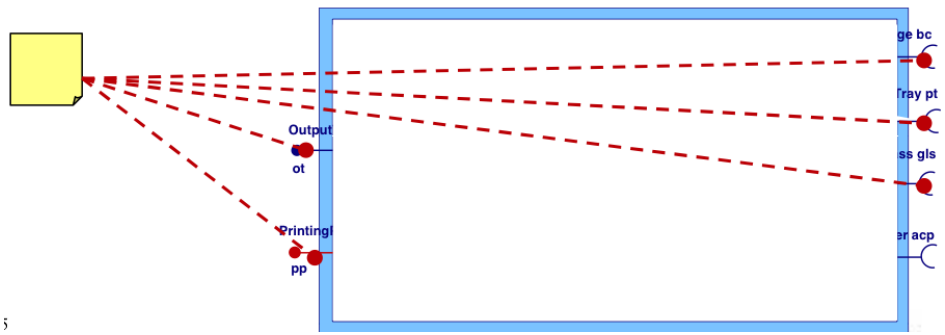


Contrat interne résultant

```
Composition contrat
              on <cpr>
 Participants <cpr>,<fnz>,
        <prt>,<sca>,<drv>;
  Provisions on server interface PrintingPanel pp {
    void copy(int n)
      pre: // guarantor is <cpr>, beneficiaries are {<cpr>}
       <prt>.acp.isConnected();
       <sca>.acp.isConnected();
       . . .
```

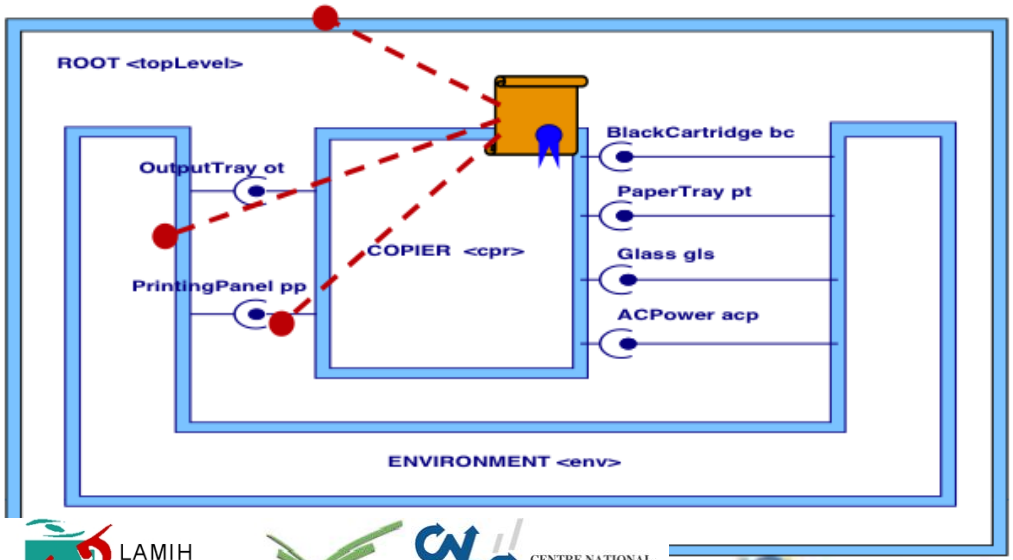
Spécification d'un composant (suite)

```
on <COPIER>
  context void pp.copy(int n) {
    pre:
      InkLevelOk:  bc.blackCartridge.level >= bc.minLevel
      TrayOk: pt.tray.count >= n && ! pt.ispaperJam()   // magasin en bon état et assez rempli
    post:
      //  Caractéristiques de la copie : égalité des images à la résolution près
      copy: forEach i in 1..n : ot.tray[i].isEqual(gls.original.image, reqPrecision)
      bc.blackCartridge.level <= bc.blackCartridge.level@pre
      pt.tray.count == pt.tray.count@pre - n // consommation de papier
```

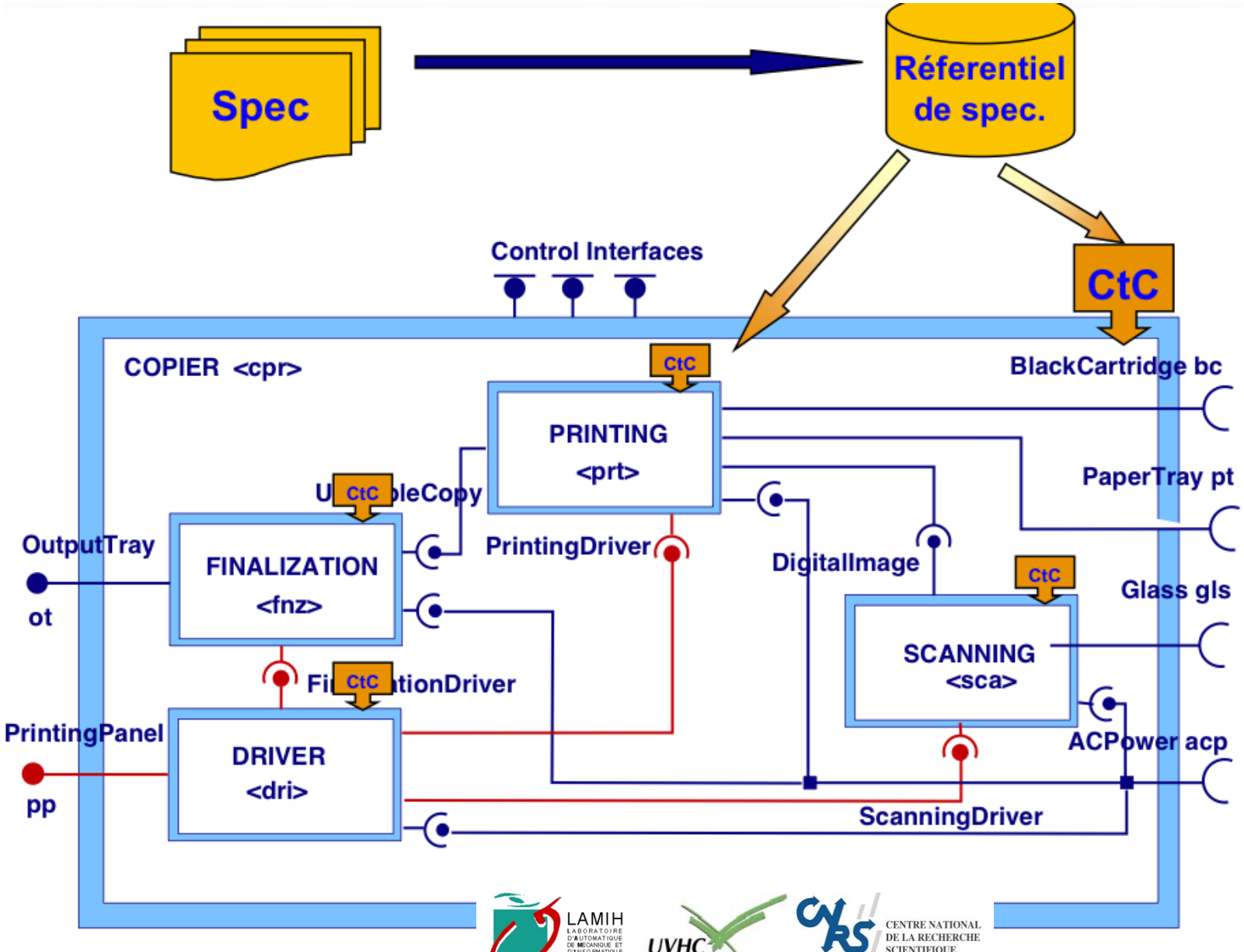


Composition externe

```
Composition contract on <cpr> { // external view
 Participants <toplevel>, <cpr>, <env>
 Provisions on server interface PrintingPanel pp {
   void copy(int n)
     pre: // guarantor is <topLevel>, beneficiaries are {<cpr>}
       bc.blackCartridge.level >= bc.minLevel
     ...
     post: // guarantor is <cpr>, beneficiaries are {<env>, <topLevel>}
        pt.tray.count == pt.tray.count@pre - n
```



Intégration dans fractal



Le Contract Controller

- **Principe**
  - Un contrôleur de contrat par composant
  - Gère les interfaces externes d'un composant primitif
  - Pour un composite gère :
    - les contrats d'interfaces de tous les sous-composants
    - son contrat de composition (interne et externe)
- **Rôles**
  - Construction de contrats
    - Interface
    - Composition : au fur et à mesure des insertions de composants.
  - Évaluation des contrats. Sont automatiquement mis à jour lors des reconfigurations dynamiques des composants

Mécanisme de validation ?

- Validation dynamique exclusivement (dans la plateforme diffusée) par le mécanisme d'assertions.
- Pour l'instant pas d'outil formel associé. Il y a eu un article montrant qu'il est possible d'utiliser TLA pour valider : ETAPSC06.
- Couplage avec une approche test en collaboration avec Daniel Deveaux du Valoria.

  Apinya Tangkawanit. Test intégré de composant basé sur les contrats. Rapport de stage Université de Bretagne Sud - Master Recherche Informatique, Juin 2006

Mécanisme de validation ?

- Validation dynamique exclusivement (dans la plateforme diffusée) par le mécanisme d'assertions.
- Pour l'instant pas d'outil formel associé. Il y a eu un article montrant qu'il est possible d'utiliser TLA pour valider : ETAPSC06.
- Couplage avec une approche test en collaboration avec Daniel Deveaux du Valoria.

  Apinya Tangkawanit. Test intégré de composant basé sur les contrats. Rapport de stage Université de Bretagne Sud - Master Recherche Informatique, Juin 2006

Mécanisme de validation ?

- Validation dynamique exclusivement (dans la plateforme diffusée) par le mécanisme d'assertions.
- Pour l'instant pas d'outil formel associé. Il y a eu un article montrant qu'il est possible d'utiliser TLA pour valider : ETAPSC06.
- Couplage avec une approche test en collaboration avec Daniel Deveaux du Valoria.

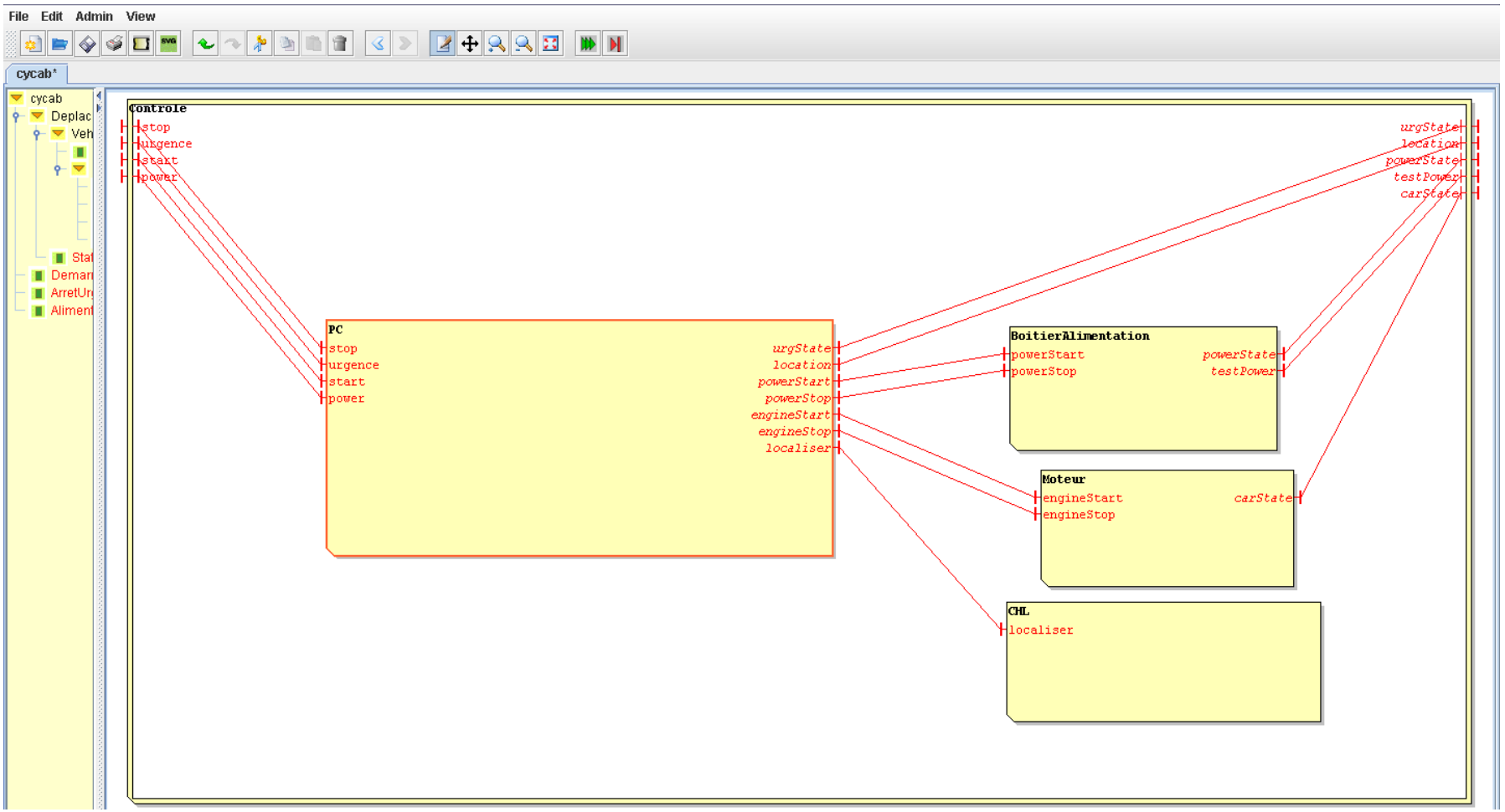
  Apinya Tangkawanit. Test intégré de composant basé sur les contrats. Rapport de stage Université de Bretagne Sud - Master Recherche Informatique, Juin 2006

Mise en œuvre de confract

Comprendre ConFract

- Pour l'instant, les exemples de la distribution fonctionnent
- Essais en cours pour créer une application "à nous"
- À partir du modèle Fractal du Cycab par Julien
- Une première idée d'intégration du composant de localisation.

Schéma du cycab avec CHL



Et maintenant

Plusieurs pistes possibles

- Intégrer un formalisme (B ?, Automates temporisés ?) à ConFract
- Quelles idées mises en œuvre dans ConFract seraient utilisables pour contractualiser des composants SCA ? *Travail de Marc Guarnieri*
- La thèse d'Alain Ozanne : **Interact** tente de s'abstraire de Fractal et propose un *cadre* pour la contractualisation de composants hiérarchiques et/ou de Web Services. Utilise les Behaviour Procoles pour formaliser les interactions.
- à la lecture de la thèse de A.Ozanne, les travaux de Milanovic : Automatic Web Service Composition, et Contract-based Web Service Composition. Utilise **B**.

Et maintenant

Plusieurs pistes possibles

- Intégrer un formalisme (B ?, Automates temporisés ?) à ConFract
- Quelles idées mises en œuvre dans ConFract seraient utilisables pour contractualiser des composants SCA ? *Travail de Marc Guarnieri*
- La thèse d'Alain Ozanne : **Interact** tente de s'abstraire de Fractal et propose un *cadre* pour la contractualisation de composants hiérarchiques et/ou de Web Services. Utilise les Behaviour Procoles pour formaliser les interactions.
- à la lecture de la thèse de A.Ozanne, les travaux de Milanovic : Automatic Web Service Composition, et Contract-based Web Service Composition. Utilise **B**.