Promotion dans les modèles à composants et services (petit tour d'horizon)

Pascal ANDRE

COLOSS / LINA – UMR CNRS 6241
{Firstname.Lastname}@univ-nantes.fr

Exposés COLOSS

Plan de l'exposé

Plan

Introduction / contexte

Contexte : Composition et promotion dans Kmelia

- observation de variables (comme un client ordinaire)
- promotion de services (appropriation/délégation)
- contrat multi-levels - cas des composites (cf exposé MM)
- aspects modèles (concepts, articulation) et langage
- aspects vérification (propriétés spécifiques)
- aspects méthodologiques (conception ascendante/descendante)

Problématique complexe : paradigmes concernés :

- agrégation (inclusion, observation, visibilité)
- promotion (délégation, partage)
- encapsulation (composition, composite)
- adaptation (transformation)
- contract (renforcement,affaiblissement)
- connecteurs (délégation, *hierarchical binding*, interception), composite connectors

Pourquoi tout ça ? : background des reviewers

Introduction / verrous

Verrous :

- que promouvoir ?
  - structure
  - comportement
  - fonctionalités, contrats
- comment promouvoir ?
  - pont de nommage
  - transformation
  - visibilité, encapsulation
- pourquoi promouvoir ?
  - composition encapsulée (composite)
  - méthodologique
- Pattern Composites
  - contrôleur global
  - connecteurs spécifique

Autre source : Zaremski[Zaremski and Wing, 1997] : component matching lattice

Plan

Classification

Etat de l'art (un début) :

- modèles : UML2, Sofa2, Fractal/Julia, BIP, ArchJava, Java/A, ... les 25 modèles de Crnkovic (GRD2009) et [Crnkovic and Larsson, 2000]
- classifier les approches
- étudier les mécanismes
- exemples

Une référence : UML 2.0 (diagramme de classes/composants) : relation de composition

- inclusion (non visible à l'extérieur) - agrégation forte
- utilisation : clientèle classique, accès par notation pointée
  - envoi de message
  - accès aux attributs publics
- promotion : uniquement pour les composants
  - *delegation connector*
  - *aggregation connector*

Classification

unité exhibée par les composants

composant pas vu

port connecteur de délégation, agrégation

- UML-2, Autosar, Kobra, Java/A, Palladio ; liaison aux interfaces
- BIP (canaux), Wright, Darwin,
- ArchJava

interface connecteur de délégation, agrégation

- Fractal/proActive : *interface binding*
  pont de nommage, connecteur de délégation, shared, filtrage dans ConFract ProActive : import/export binding
- Sofa
-

service binding

- SCA : service promotion [Ding et al., 2008a]
- Darwin : binding
- Unicon : connecteur simple
- ...

Classification 2

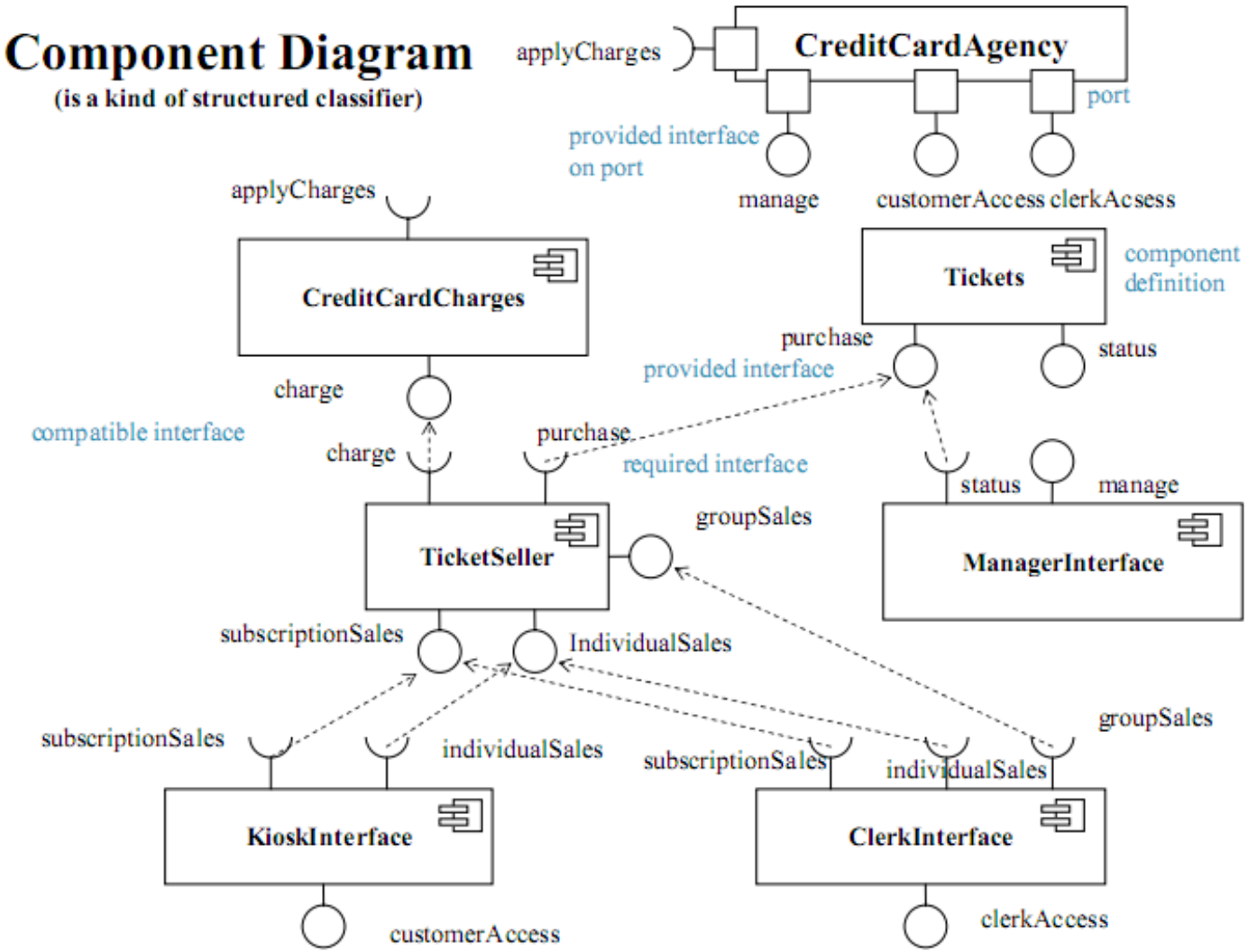
Contrôle par contrats

contracts translationnel

- Hidden dependencies [Enselme et al., 2004]
- Full contracts [Fenech et al., 2009]
- ConFract for composites [Collet et al., 2007a]
- Safety systems [Dong et al., 2007]
- Kmelia
- ...

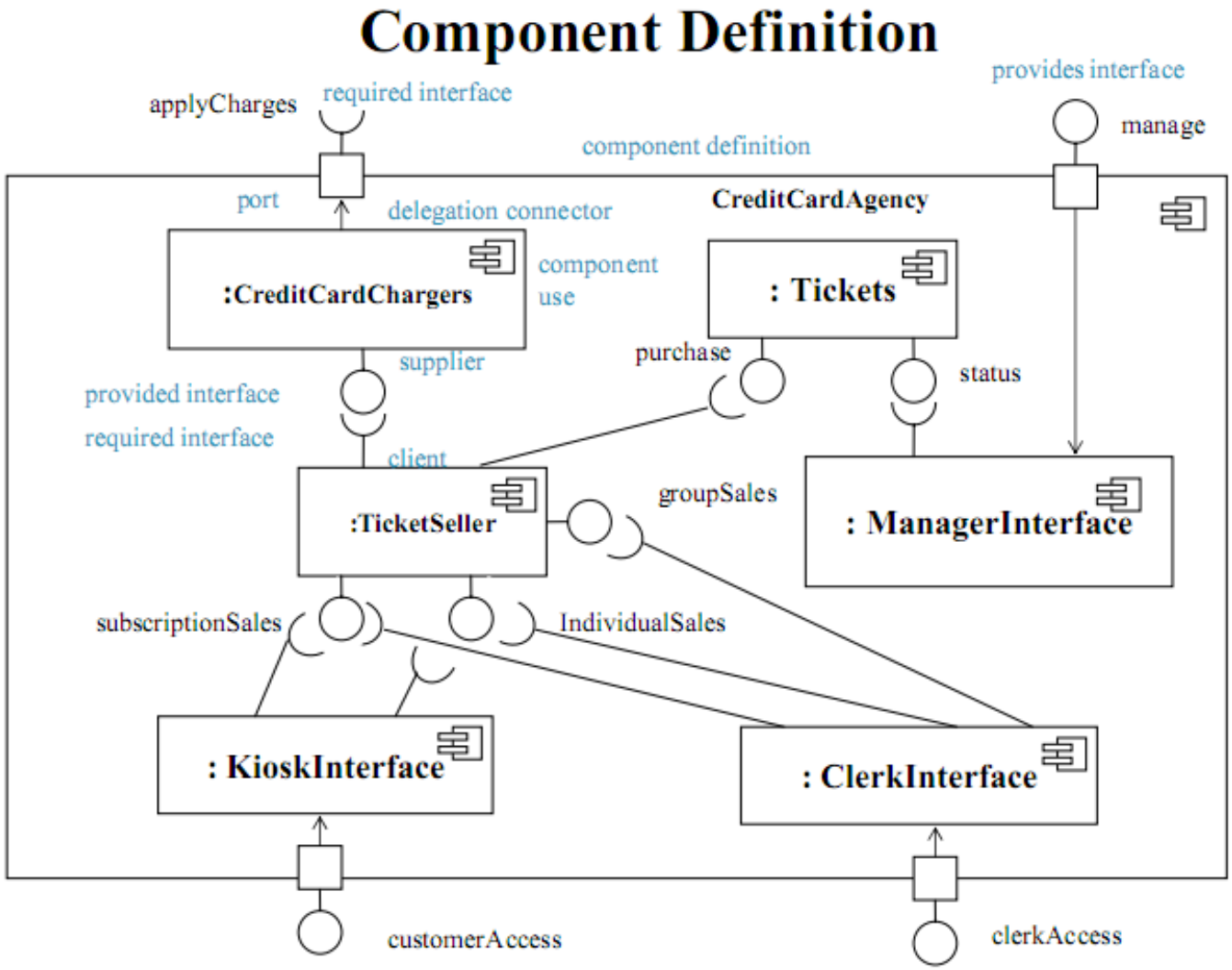
Plan

UML 2.0 Components 1/2



http://www.soi.wide.ad.jp/class/20060036/materials_for_student/14/14-UML2.pdf

UML 2.0 Components 2/2



http://www.soi.wide.ad.jp/class/20060036/materials_for_student/14/14-UML2.pdf

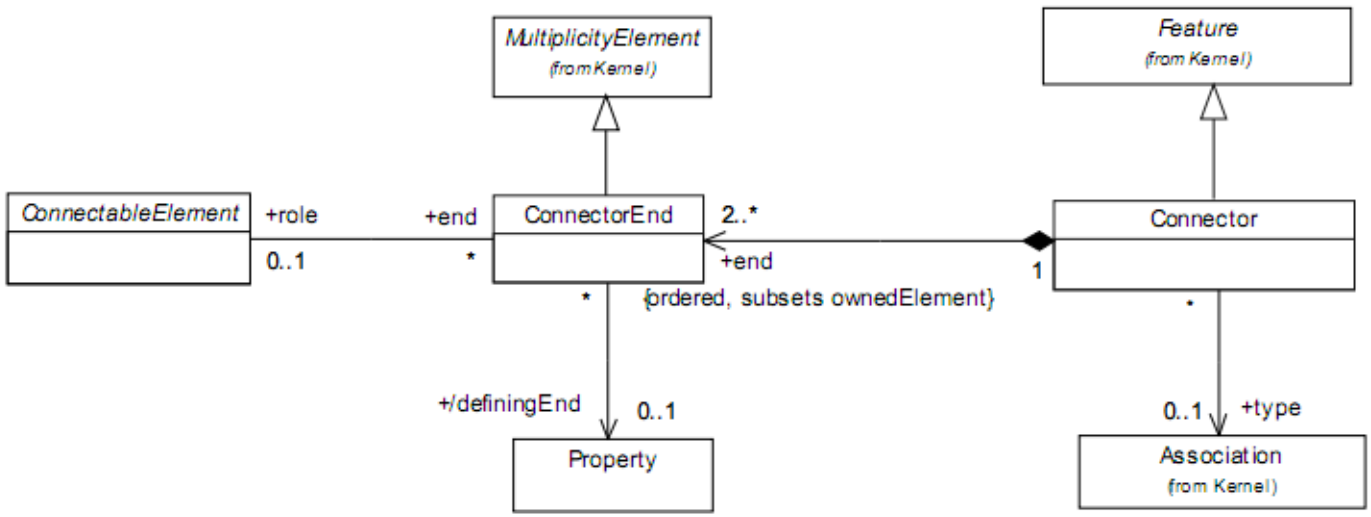
UML Connectors 1/3

The connector concept is extended in the Components package to include interface based constraints and notation.

- A delegation connector is a connector that links the external contract of a component (as specified by its ports) to the internal realization of that behaviour by the component's parts. It represents the forwarding of signals (operation requests and events) : a signal that arrives at a port that has a delegation connector to a part or to another port will be passed on to that target for handling.
- An assembly connector is a connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port.

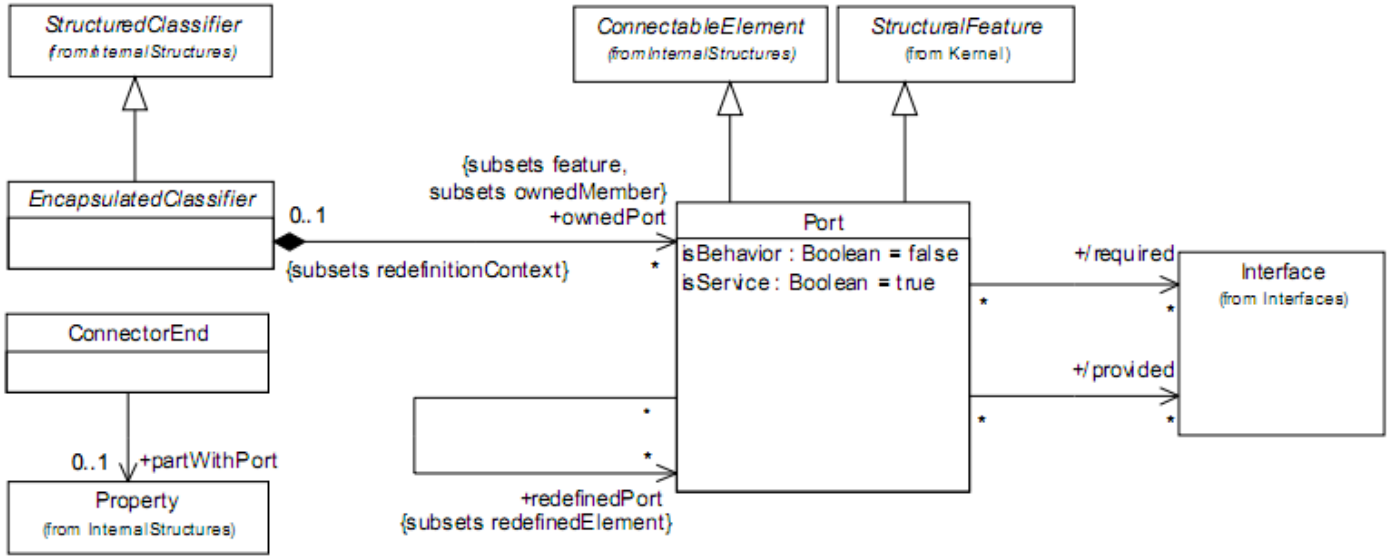
lien promo / lien assemblage
(source : TopCase)
Ca reste un Modèle de référence

UML Connectors 2/3



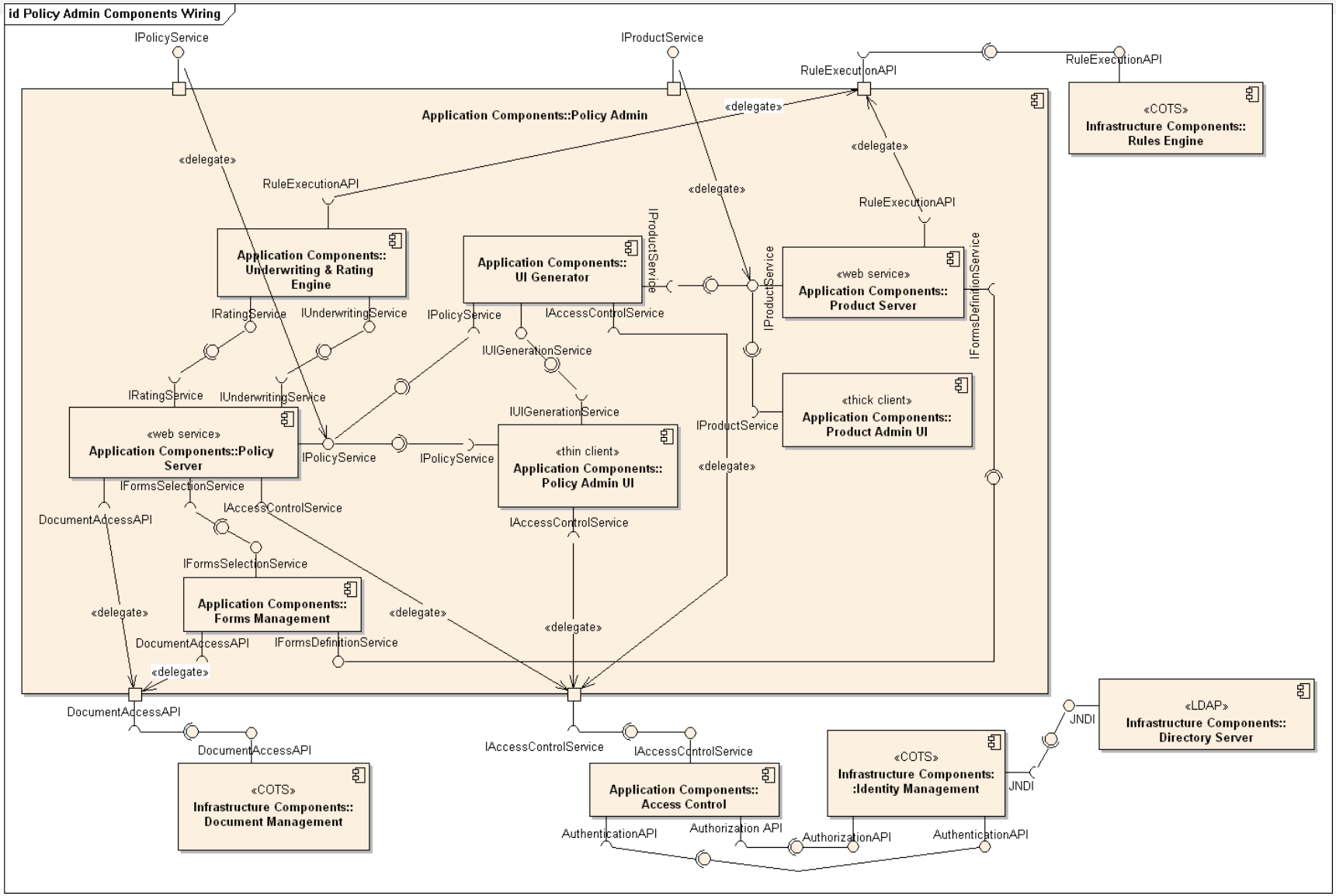
UML Superstructure 2.0 Draft Adopted Specification

UML Connectors 2/3



UML Superstructure 2.0 Draft Adopted Specification

Exemple UML

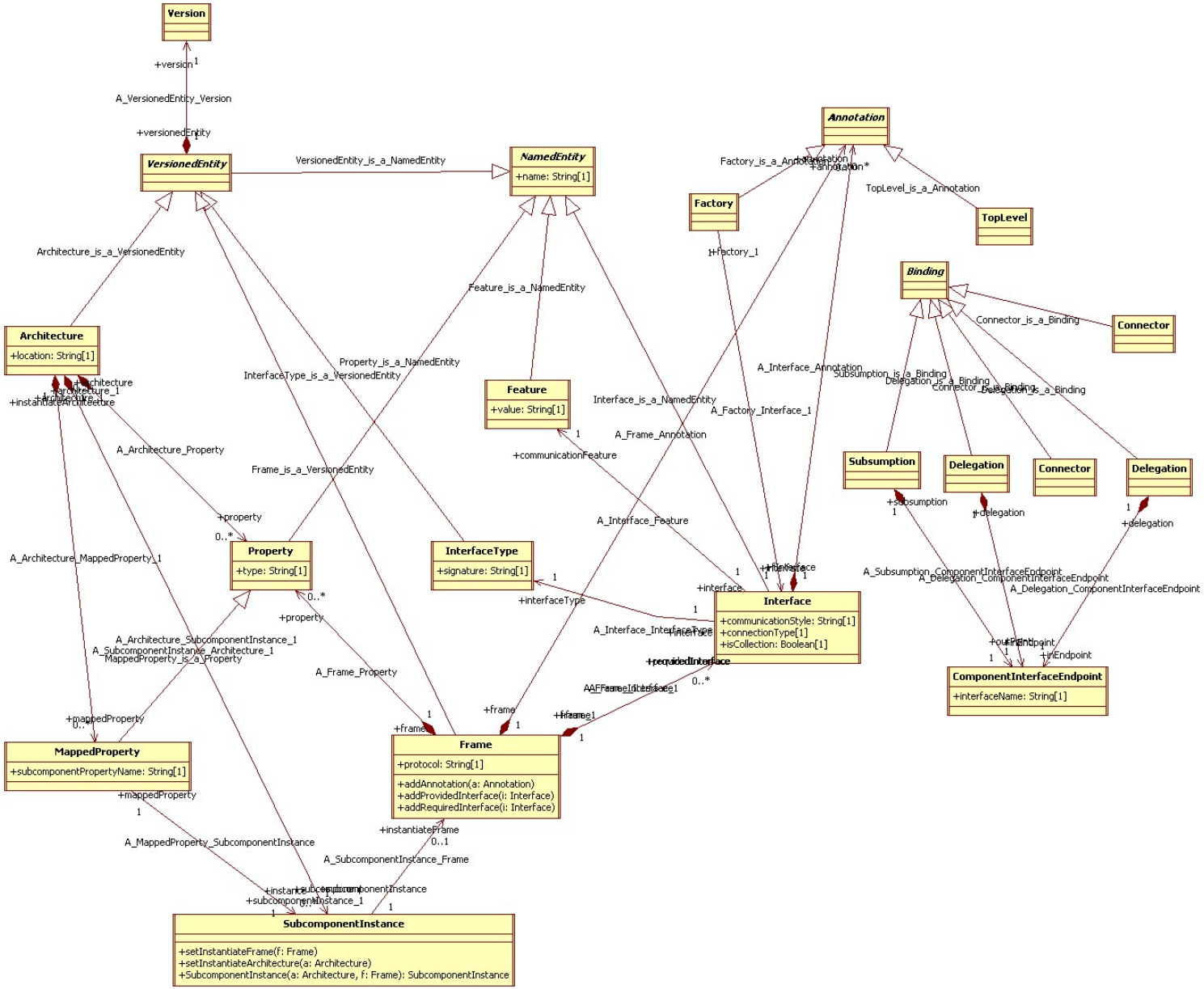


http://upload.wikimedia.org/wikipedia/commons/b/b8/Policy_Admin_Component_Diagram.PNG

Sofa 2 Metamodel 1/2

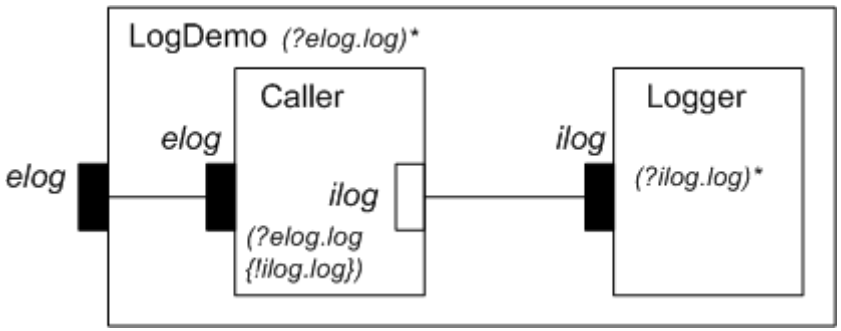
- composition
  - architecture implements frame
  - récursif : plus de 2 niveaux ? ?
- connector
  - *bind* `p-r`
  - *delegate* `p-p`
  - *subsume* `r-r`
- traitement
  - design : vérification static/dyn pas contrat
  - deployment : génération automatique
- pas vu de transformation mais *control by aspects*

Sofa Metamodel 2/2



Sofa Example

[Bures et al., 2006, Bureš et al., 2008] my `sofa2console` fails
Short example of SOFA architecture : the `LogDemo` example [André et al., 2007].




```
module logdemo {
  interface LogInterface {
    void log(in string number);
  };
  frame Caller {
    requires:
      LogInterface Log;
  };
  frame Logger {
    provides:
      LogInterface Log;
  };
};
```

There are two primitive components - `Caller` (or `Caller` in his example) and `Logger`. `Logger` provides the interface `LogInterface`, `Caller` requires this interface, in the component `logdemo` these two components are instantiated and their provision and requirement are bound together.

Sofa Validation

Validation	Type	Number	Message
A class can not have neither subelements nor ports.	Error	239	A class can't have any elements.
	Error	240	A class can't have any ports.
Delegate Connector			
A delegate connector must be attached to instance specification or component or property.	Error	156	A supplier end of a delegate connector must be an instance of a component or a property.
	Error	157	A client end of a delegate connector must be a component, that owns a connector.
A delegate connector must connect component to its subcomponent.	Error	158	A supplier end of a delegate connector must be a subcomponent, of a component that owns a connector.
A subsume connector must be attached to instance specification or component or property.	Error	159	A client end of a subsume connector must be an instance of a component or a property.
	Error	160	A supplier end of a subsume connector must be a component, that owns a connector.
A subsume connector must connect subcomponent to its owner.	Error	161	A client end of a subsume connector must be a subcomponent, of a component that owns a connector.
A delegate/subsume connector must interconnect component and its subcomponent with the same provisions/requirements.	Error	162	A Delegate/Subsume connector must be between a component and a subcomponent.
	Error	163	Components connected with a Delegate/Subsume connector must have the same interface(s).
	Error	164	Components on a client end of a Delegate/Subsume connector must have the same interface as a port connected on a supplier end.
	Error	165	Components on a supplier end of a Delegate/Subsume connector must have the same interface as a port connected on a client end.
	Error	166	Ports on client and supplier end of a Delegate/Subsume connector must have the same interface.
Frame			
A frame must have a name.	Error	121	A Frame must have a name.
A frame may be contained only by a package.	Error	122	Only a Package can be a parent of a Frame.
Invalid connector attached to a frame. Both client and supplier end must be specified.	Error	123	Invalid connector.
	Error	127	Invalid connector.
A frame inheritance is allowed only between frames.	Error	124	A base of a Frame must be another Frame.
	Error	128	Only a Frame can inherit from a Frame.
A frame can not realize other elements. Only an architecture may realize frame.	Error	125	A Frame can't be on a client end of a realization.
	Error	129	A realization of a Frame must be an Architecture.
Only *generalization* and *realization* dependency may be attached to a frame.	Error	126	A connector can't be connected to a Frame on a client side.
	Error	130	A connector can't be connected to a Frame on a supplier side.
Methods of frames are not allowed.	Error	131	A Frame can't have any methods.
A frame can not have an internal structure.	Error	132	A Frame can't have any subcomponents.
	Error	133	A Frame can't have any bindings.
InstanceSpecification			
An instance must have a name.	Error	174	An InstanceSpecification must have a name.

UML 2.0 Components — Appendix B. SOFA Validations - 3

source:Polakmaster

ArchJava

[Abi-Antoun et al., 2007, Aldrich et al., 2002a, Aldrich et al., 2002b, Aldrich, 2008]
a small, backwards-compatible extension to Java that integrates software architecture smoothly with Java implementation code

- ArchJava seamlessly unifies architectural structure and implementation in one language, allowing flexible implementation techniques, ensuring **traceability** between architecture and code, and supporting the **co-evolution** of architecture and implementation.
- ArchJava guarantees **communication integrity** between an architecture and its implementation, even in the presence of advanced architectural features like **run time component creation** and **connection**.
- \+ **Communication Integrity** : *Each component in the implementation may only communicate directly with the components to which it is connected in the architecture.*
- Composites
- Dynamic architectures
- pas de contrats

ArchJava model 1

Component A component is a special kind of object that communicates with other components in a structured way. Components are instances of component classes.

Ports A component can only communicate with other components at its level in the architecture through explicitly declared ports-regular method calls between components are not allowed. A port represents a logical communication channel between a component and one or more components that it is connected to.

```
public component class Parser {
public port in {
provides void setInfo(Token symbol, SymTabEntry e);
requires Token nextToken() throws ScanException;
}
public port out {
provides SymTabEntry getInfo(Token t);
requires void compile(AST ast);
}
public void parse() {
Token tok = in.nextToken();
AST ast = parseFile(tok);
out.compile(ast);
}
AST parseFile(Token lookahead) { ... }
void setInfo(Token t, SymTabEntry e) {...}
SymTabEntry getInfo(Token t) { ... }
...
}
```

ArchJava - Composition/Promotion

Composition *Composite* components are made up of a number of subcomponents connected together. A subcomponent1 is a component instance nested within another component.

Connection The symmetric `connect` primitive connects two or more ports together, binding each required method to a provided method with the same name and signature.

Délégation *Provided methods can be implemented by forwarding invocations to subcomponents or to the required methods of another port.*

```
public component class Compiler {
private final Scanner scanner = ...;
private final Parser parser = ...;
private final CodeGen codegen = ...;

connect scanner.out, parser.in;

connect parser.out, codegen.in;

public static void main(String args[]) {
new Compiler().compile(args);
}

public void compile(String args[]) {
// for each file in args do :
... parser.parse ();...
}
}
```

Fractal/Julia

[Bruneton et al., 2006]

Modèle délimitations de zones de code

- Composants, Interfaces, Membrane, contrôleur, Liaisons
- Instanciation, Hiérarchisation, Partage, Typage

Composant délimitations de zones de code

- membrane $\Longrightarrow$ Interfaces fonctionnelles + Liaisons
- contenu $\Longrightarrow$ sous-composants

Interface délimitations de zones de code

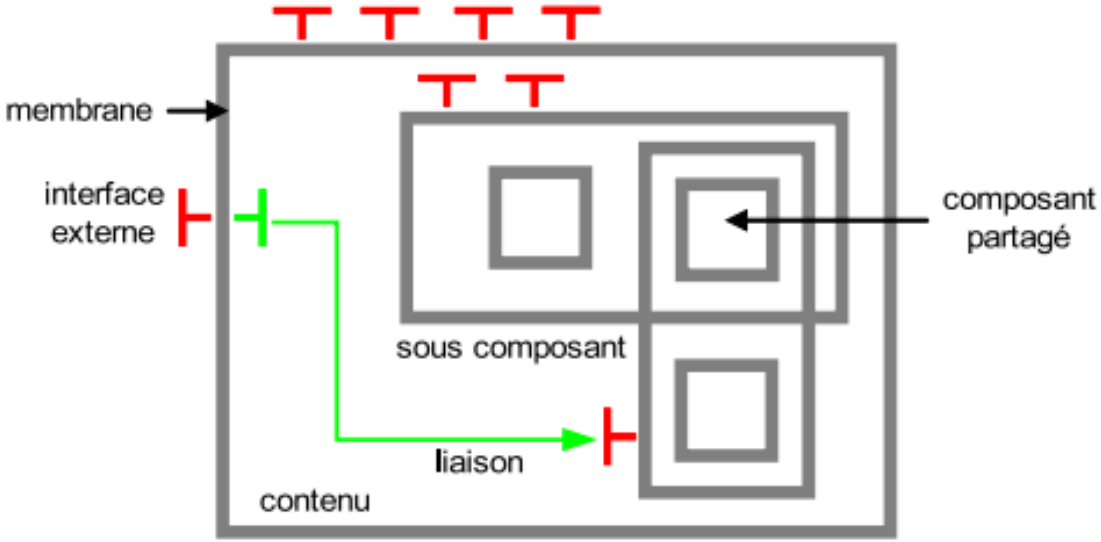
- externe $\Longrightarrow$ accessible
- interne $\Longrightarrow$ par les sous-composants

Architectures

- imbrication
- liaison - *binding controllers* $\Longrightarrow$ aspects méthodologiques
  - primitive (client/serveur)
  - composite (chemin) : ensemble de composants de liaisons *import/export bindings*

Exemple : [Bulej et al., 2008]

Fractal Composite



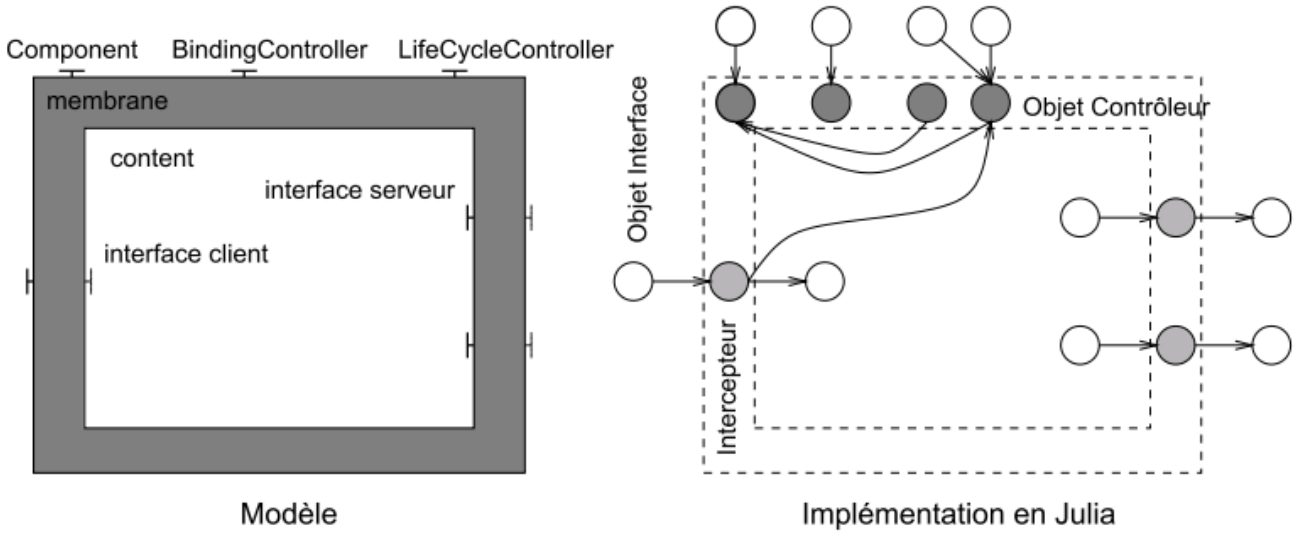
Contrôleurs (sémantique dans l'implantation)

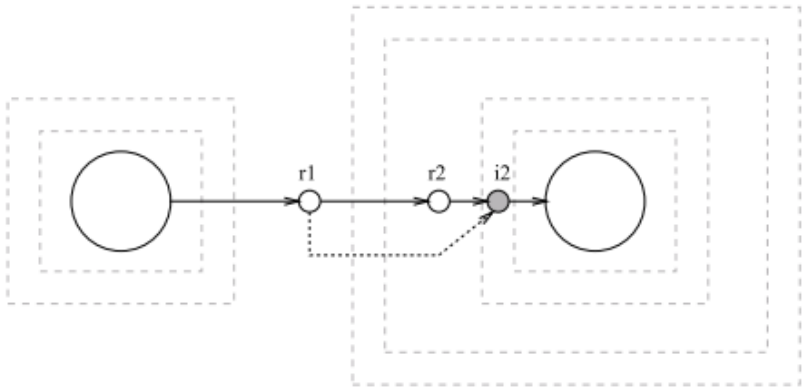
- attributs
- liaison
- contenu
- cycle de vie

`source:Intergiciel`

Fractal Controlers (Julia)

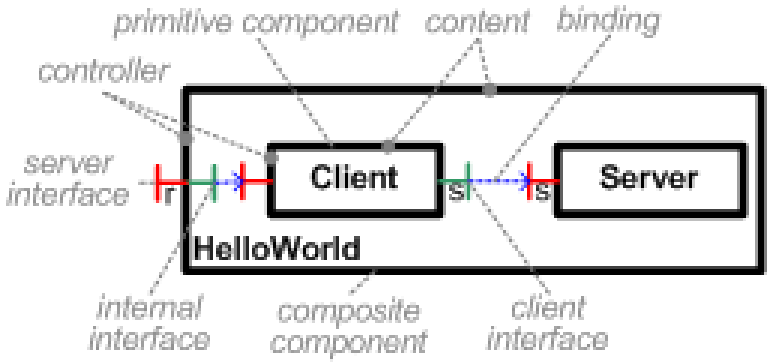
Implantation du contrôle et optimisation





source:Intergiciel

Fractal Example



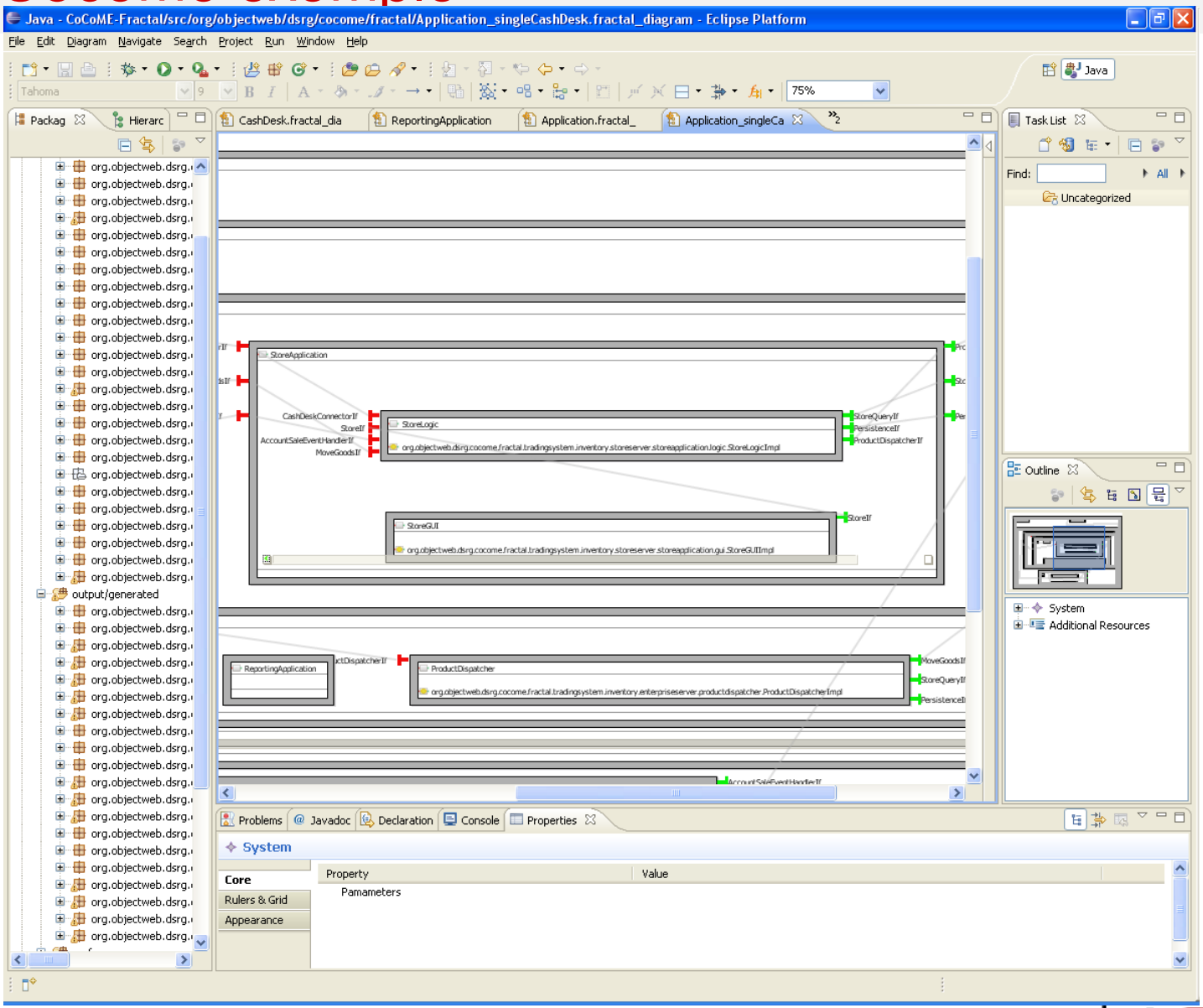

```
<definition name="HelloWorld">
    <interface name="r" role="server" signature="java.lang.Runnable"/>

  <component name="client" definition="Client"/>
    <component name="serverA" definition="ServerA('Hi!')"/>

  <binding client="this.r" server="client.r"/>
    <binding client="client.default" server="serverA.service"/>
</ definition >
```

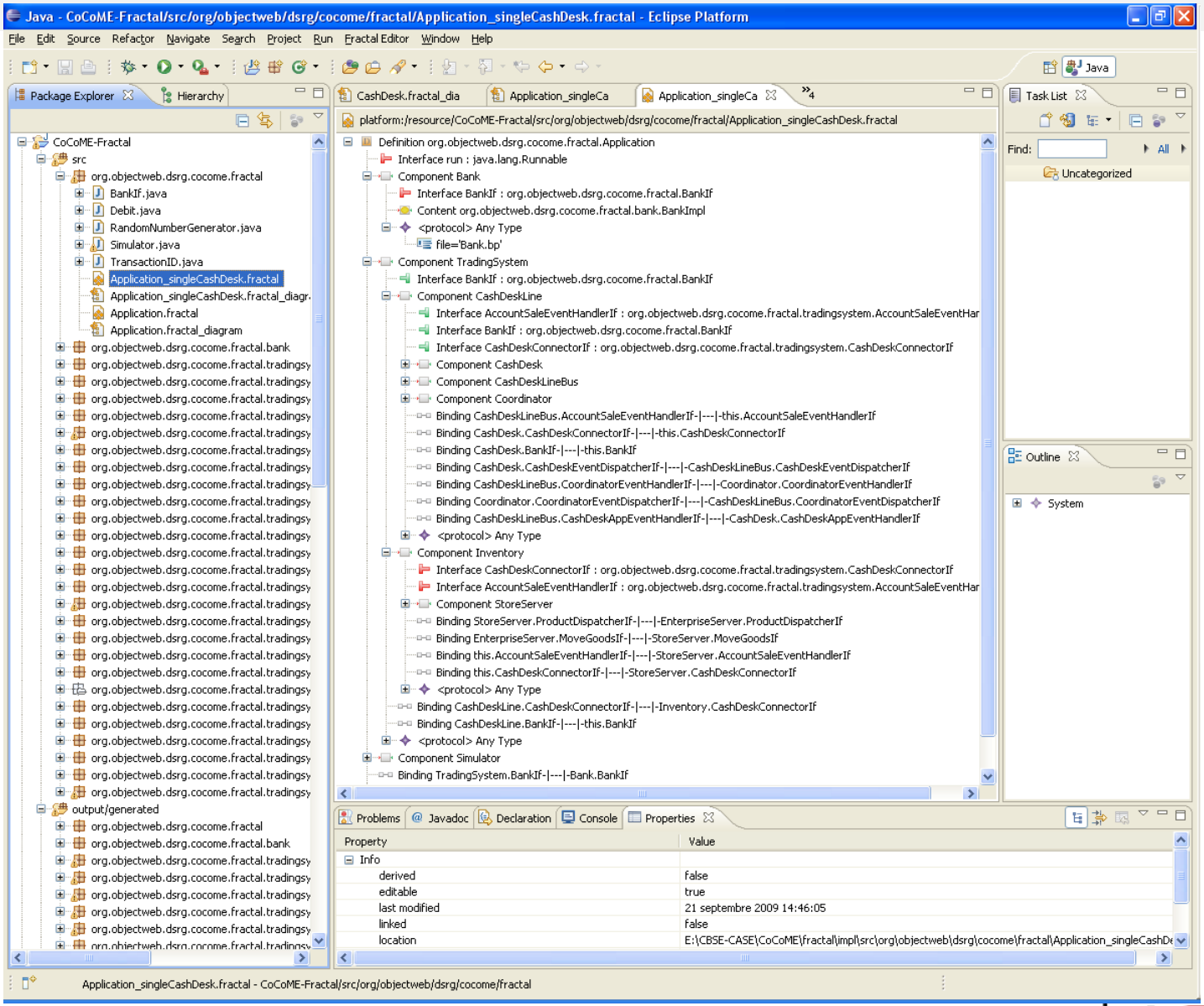
Voir aussi mon exposé sur la génération de code source : `http://fractal.ow2.org/`

Fractal/Cocome exemple



CoCoME architecture / Fractal ADL / Fractal RMI

CoCoME/Fractal



CoCoME génération

Fractal/Vercors

Vercors [Barros et al., 2007]

- GCM : promotion mais pas contrats
- ports = canaux
- CADP - pas de contrats, filtrage possible - pas de redéfinition

GCM/Proactive [Baduel et al., 2006]

- promotion mais pas contrats
- communications à N : Gathercast
- ...

ConFract [Collet et al., 2005]
a suivre

Plan

Contrats et promotion

- ConFract for composites [Collet et al., 2007a]
- Contracts in BIP [Graf and Quinton, 2007]
- Hidden dependencies [Enselme et al., 2004]
- Full contracts [Fenech et al., 2009]
- Safety systems [Dong et al., 2007]
- Kmelia
- ...

Fractal/ConFract

Fractal contract [Collet et al., 2005, Ozanne, 2007, Collet et al., 2007a]

Classification des contrats

de bibliothèque contrat "objet" (au niveau des classes)

d'interface contrat d'une connexion point à point entre une interface requise et une interface fournie.

l'assertion peut comporter des appels aux autres méthodes de l'interface possédant la méthode sur laquelle elle porte

de composition contrat défini sur la membrane d'un composant Fractal

externe : règles d'utilisation et de comportement externe du composant sur plusieurs interfaces externes.

l'assertion peut comporter des appels aux méthodes des interfaces externes du composant porteur de l'interface exposant la méthode qu'elle contraint

interne : règles d'assemblage et de comportement interne d'implantation (interfaces internes + sous-composants).

l'assertion porte sur une interface d'un composant composite et elle peut comporter des appels aux méthodes des interfaces : a) internes du composites, b) externes des sous composants.

Source : Poirriez, Ozanne, Chang

Fractal/ConFract

Fractal contract [Collet et al., 2005, Ozanne, 2007, Collet et al., 2007b]
Composite contract [Collet et al., 2007a]

Langage CCL-J = *Component Constraint Language for Java.* - Inspiré d'OCL

Sémantique des contrats

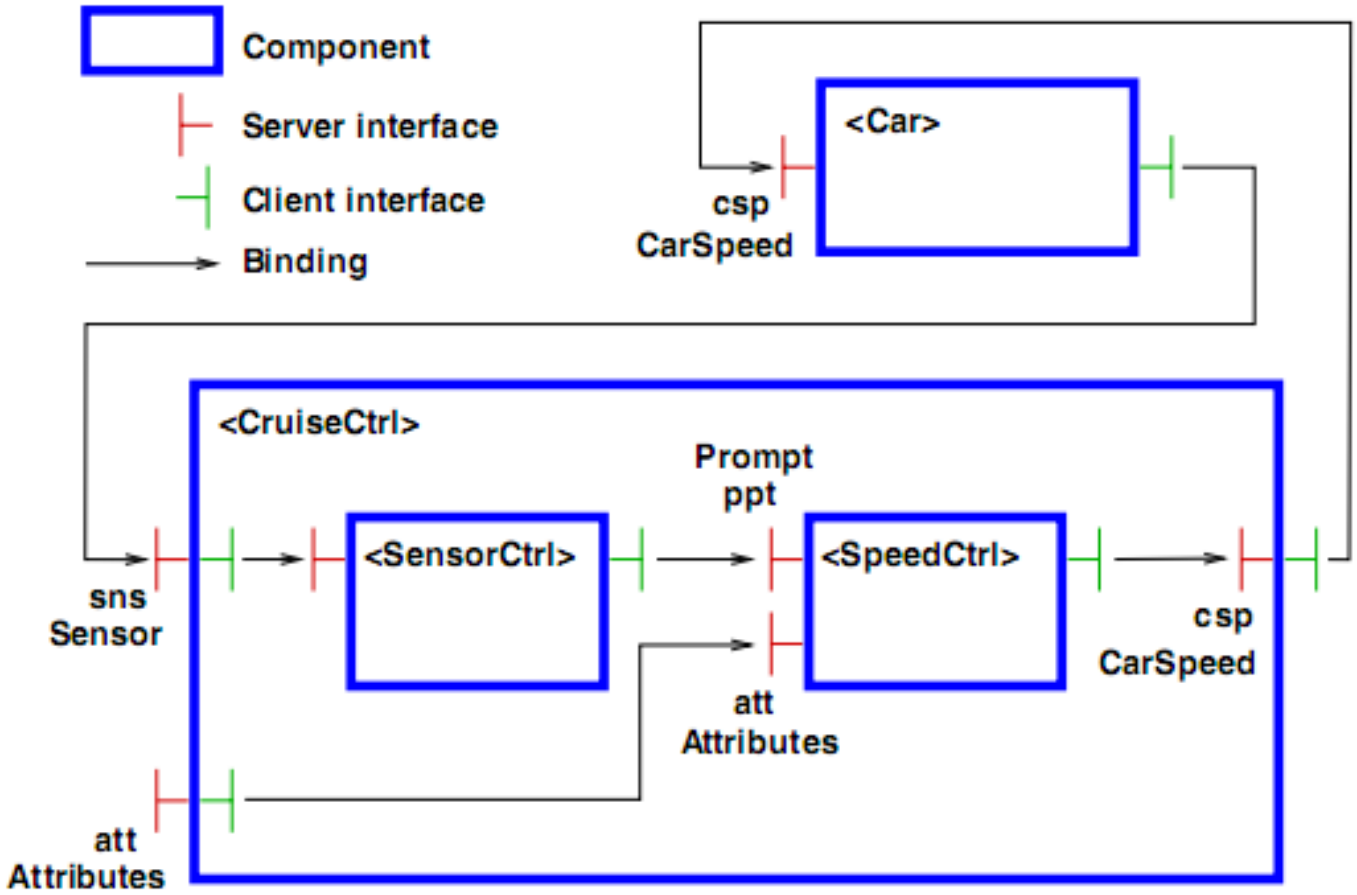
Contrat d'interface : Evaluation classique des pre et post conditions, ainsi que des invariants.

Contrat de composition : Conjonction de chaque type de spécification Le composite (l'assembleur) est responsable de tout problème survenant à son niveau.

Négociation (Chang)

Exemples : slides V. Poirriez, etc.

ConFract exemple

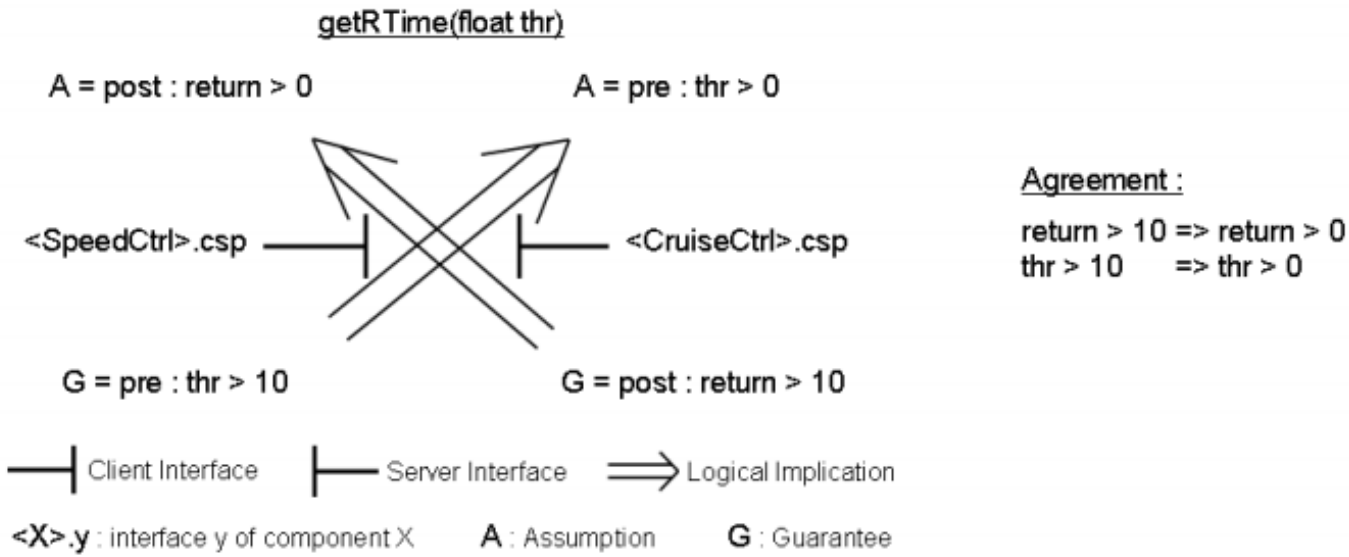


[Collet et al., 2007b]

ConFract exemple

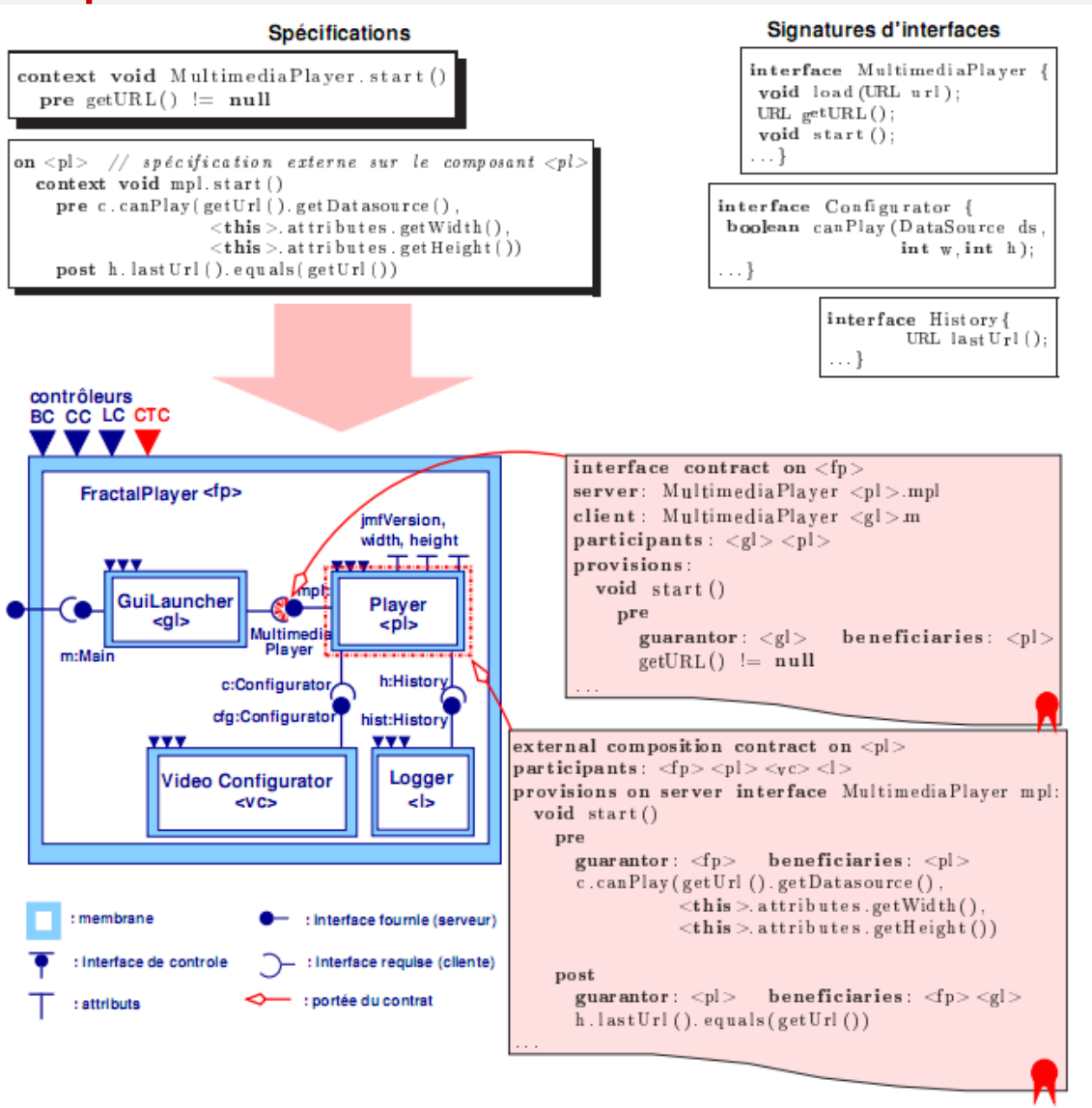
```
Contract {
  Participants : <component_name>* ;
  Clauses :
  {
   clause : <name>
     responsible : <component_name>;
     guarantee : <expression>;
     assumption : <expression>;
   }*
  Agreement : { agreement expression }
}
```

Fig. 2. Concrete syntax of a contract



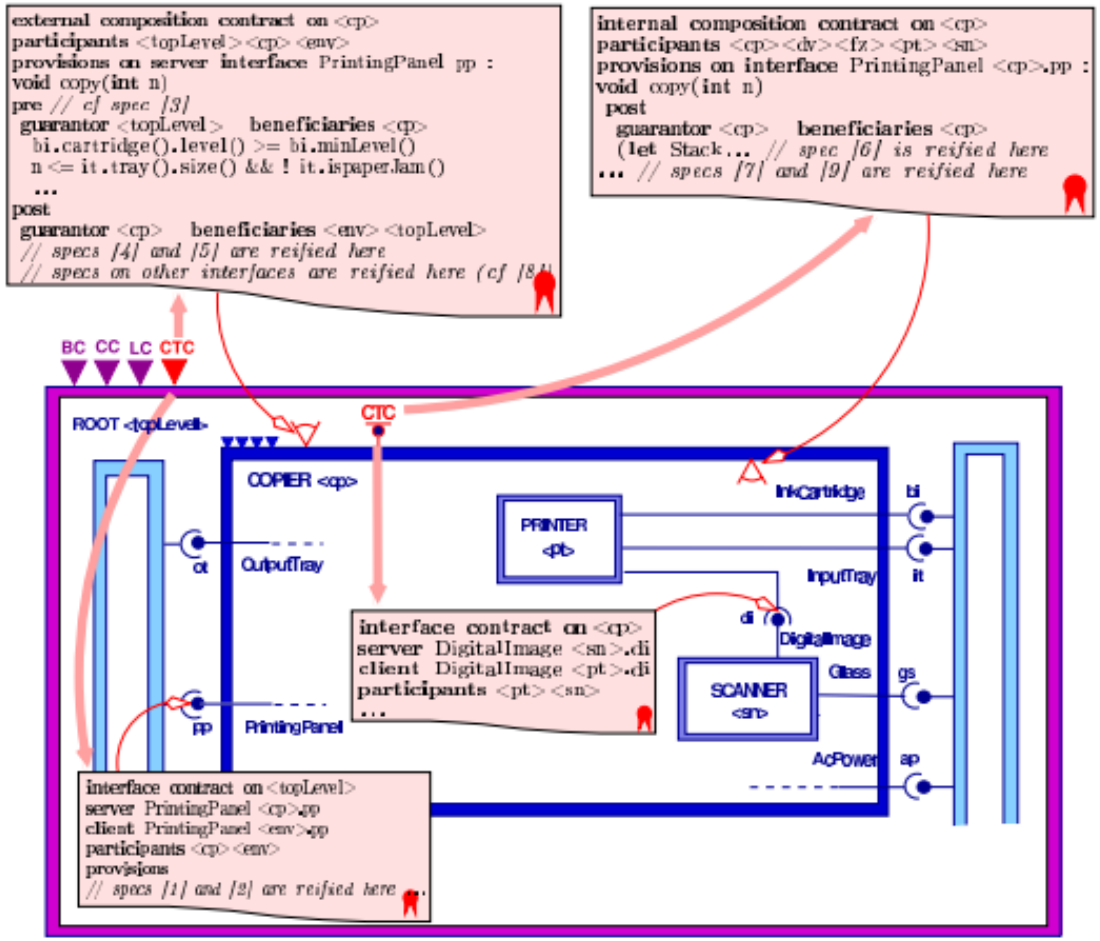
[Collet et al., 2007b]

ConFract exemple



[Collet et al., 2005]

ConFract exemple



[Chang and Collet, 2005]

Fractal Synthèse

Avantages

1. Préserve l'encapsulation via Fractal (interfaces, liaisons, membranes)
2. CCL-J proche d'OCL
3. Contrats *Assume/Guarantee* dissociés du modèle

Inconvénients

1. orthogonal à la dynamique
2. pas simple à comprendre
   - lien entre opération, contrat et interface : contrat sur un ensemble d'opérations dans un service
   - assertions sur les opérations mais prenant en compte d'autres interfaces
   - le contrat d'assemblage n'est pas la conformité mais vient se plaquer sur un assemblage
3. évolution des interfaces ?
4. cohérence globale des contrats ? ?

Promotion

1. lien interne/externe réalisé par un lien d'assemblage
2. tout ou rien : interface promue est identique
   - promotion directe des interfaces (cf Fractal)
   - prise en compte des autres composants dans le contrat promu

Contracts for BIP

[Graf and Quinton, 2007, Quinton and Graf, 2008]

Caractéristiques

1. *Assume/Guarantee*
2. contrats sur interfaces *Assume/Guarantee* et environnement
3. A et G sont des systèmes de transitions
4. Hierarchical components
5. promotion sur ports
6. inclusion basée sur le raffinement

Avantages

1. compositionnalité
2. dynamique intégrée
3.

Inconvénients

1. pas d'exemples
2. contrats comportementaux - seulement de la dynamique ?
3. mise à plat

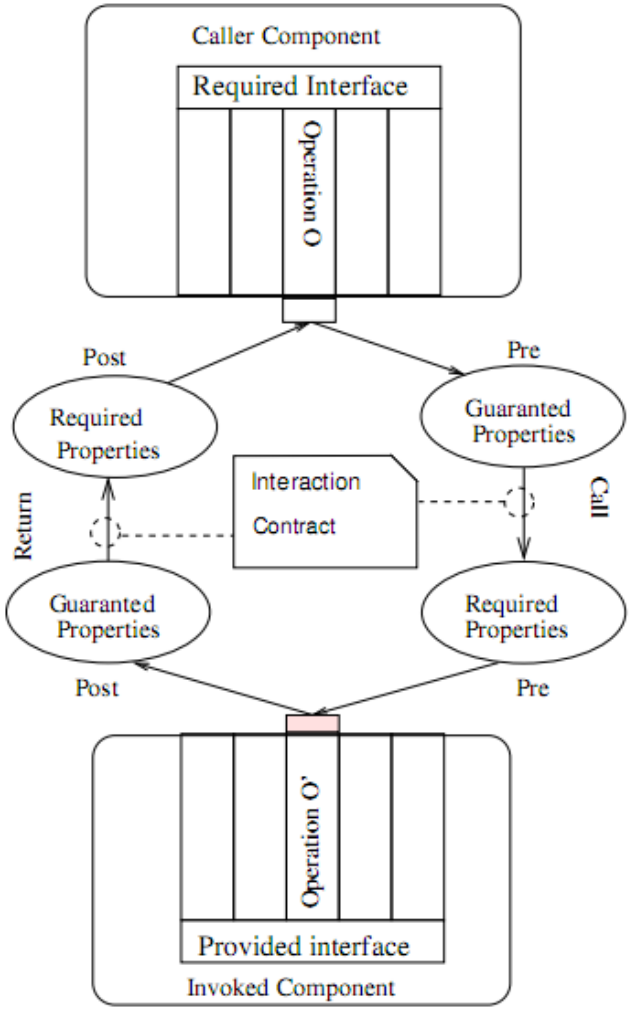
Voir formalisme de Benvenuti [Benvenuti et al., 2008] ?

Design by Contracts : Hidden dependencies

[Legond-Aubry et al., 2003, Enselme et al., 2004]

Contrats d'assemblages entre composants

1. spécification des requis (1/2 contrats)
2. contrat assemblage classique : contravariant (contrat simple)
3. pas clair
   - lien entre opération et interface mais semble comme Kmelia
   - hypothèses différentes O/R mais pas de lien de contextes
4. relation de compatibilité entre opérations
5. compléments (dans la thèse)
   - ports pour grouper des interfaces
   - connecteurs pour l'adaptation

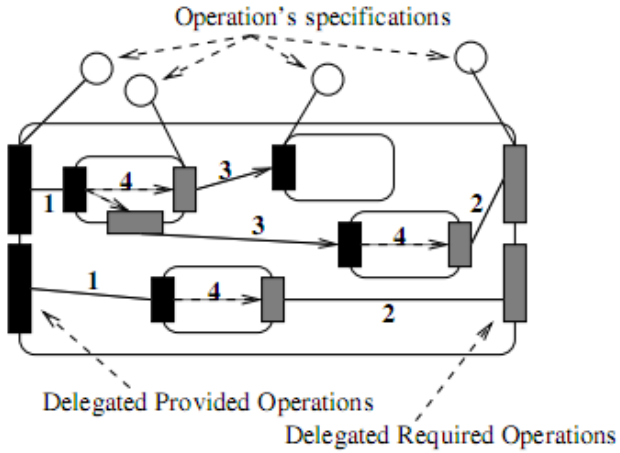


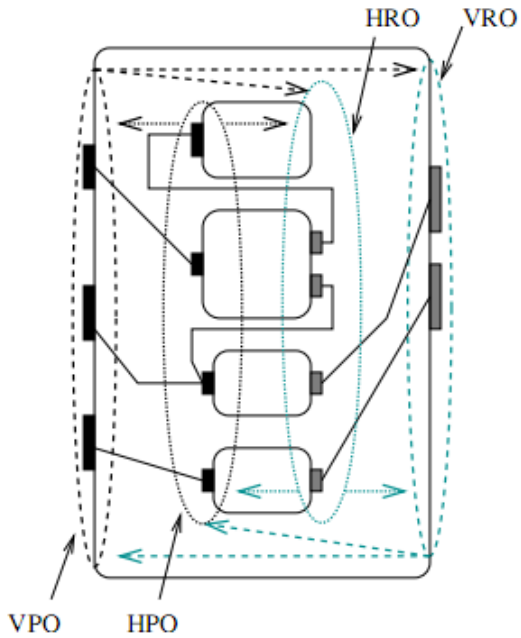
Design by Contracts : Hidden dependencies

[Legond-Aubry et al., 2003, Enselme et al., 2004]

Composite

1. inclusion des sous-composants
   - services offerts publics/privés (cachés)
   - services requis publics/privés (cachés)
2. 4 types de dépendances
3. délégation +/- complexe
   1. pure (1-1) : identitique
   2. en compatibilité (variante
   3. en délégation adaptation (multiple)
      1. offert : Façade (1-n) ou Mux (n-1)
      2. requis : n fois "pure"
4. le composite substitue (4 équations)
5. des interrogations
   - contexte non abordé
   - equation en conjonction et non implication
   - Vérification ?





2

Full contracts

[Fenech et al., 2009]
Objectifs

1. specifies not only the normal behaviour, but also special cases and tolerated exceptions
2. we focus on the behavioural properties of use cases
3. CoCoME

Proposition

1. $\mathcal{CLCL}$, a deontic-based formal language for contracts
2. formules de logique temporelle

Promotion

1. ? ?
2.

Contract-Based Component

[Lee et al., 2005, Lee et al., 2002]

Objectifs

1. Design by Contract : 3 niveaux (syntaxe, behaviour, QoS) - pas la dynamique
2. Specification with Z
3. Analysis of the relation between components using category theory.

Hypothèses

1. Schémas Z = components (including operations)
2. extraction des contrats

Interrogations

1. pas bien compris ? ? [Lee et al., 2002] non dispo
2. promotion ? (ca existe en Z)
3.

Safety systems

[Dong et al., 2007]
Objectif : specify and verify the contract of static structure, dynamic behavior and refinement of component systems

Caractéristiques

1. modèle de base UML2 : formalisation allégée dans un modèle mathématique
2. delegation connector
3. dynamic contract : (protocol state machines) , composition $\sim$ contract automata
4. pas de contrats fonctionnels
5. raffinement : relation de bisimulation

Verification

1. règles données
2. automatisation ?
3.

Plan

Contrats, services et promotion

- SCA [Fiadeiro et al., 2006, Ding et al., 2008b]
- Contract-based WS composition [Milanovic, 2005]
- Contract-based Service discovery and composition [Brogi, 2010]
- Kmelia
- ...

Contract Definition Language

[Milanovic, 2005]
a small, backwards-compatible extension to Java that integrates software architecture smoothly with Java implementation code

- Contract Definition Language (XML) et processus "intégré".
- Modélisation des services par AMN (*An element, which can be a class, a component, or a Web service, is represented as an abstract machine*).
- Composition de services = composition d'opérations (seq

Commentaire

- simple
- pas clair : composition de services = composition d'opérations + fusion de machines
- proche sur B mais pas de promotion
- Vérif : type checking, PO, correct termination $trm(S)$
- ...

Contract-based Service discovery and composition

Brogi [Brogi and Corfini, 2007, Brogi, 2010]
Suite travaux avec Canal et Pimentel *service contract* Stack

1. Behavior information (protocols)
2. Ontology annotations (for data) OWL-S, WSDL-S, ...
3. QoS information (availability, performancve, security)
4. Signature information

Commentaire

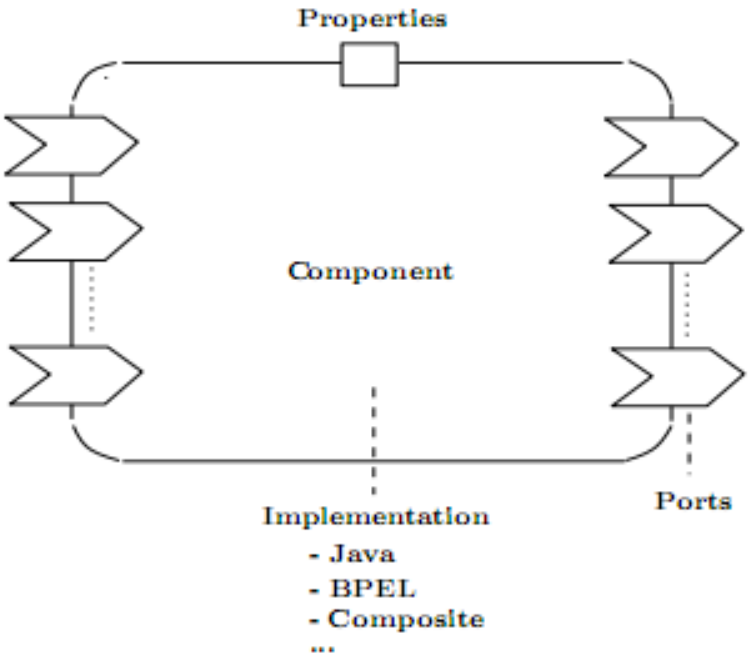
- pas de contrat classique
- pile contratuelle différente de celle de CBSE [Beugnard et al., 1999]
- protocoles mais pas de promotion, encapsulation
- ...

Service Component Architecture 1/4

[Fiadeiro et al., 2006, Ding et al., 2008b, Krämer, 2008]

SCA is a set of specifications which describe a model for building applications and systems using a Service-Oriented Architecture

`http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications`

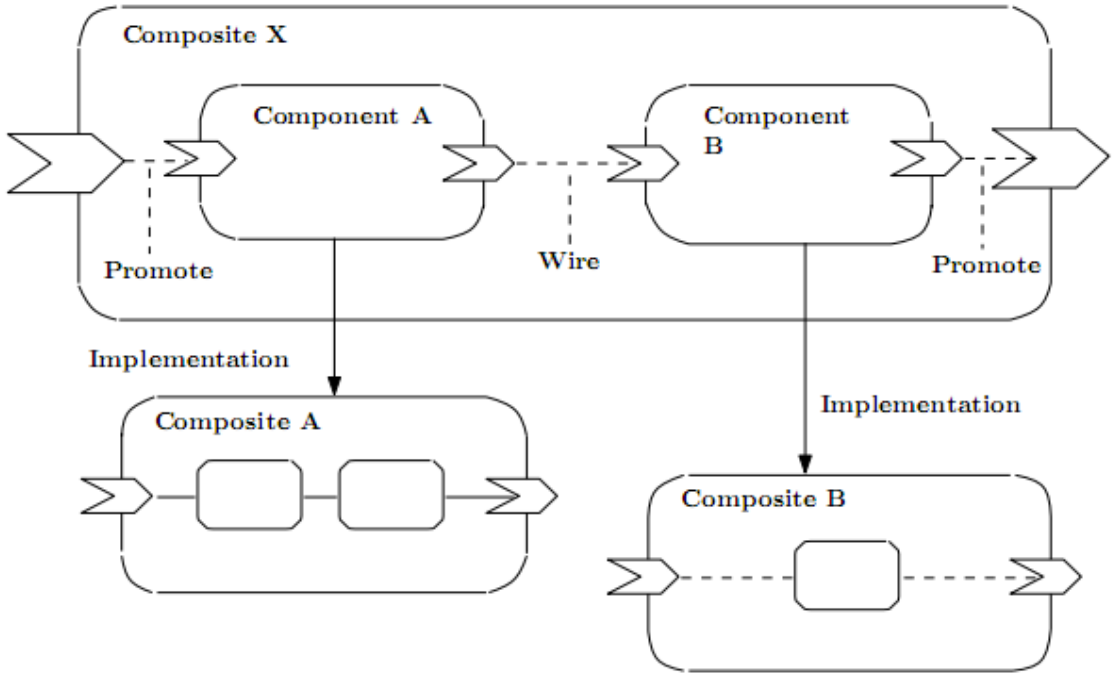


source : [Ding et al., 2008b]

- service : operation activities
- interaction : message exchange
- port = adressable interfaces (sync/async)
  - `service port` (prov) | Kml sens
  - `reference port` (req) | inversé
- *port activities to describe the component dynamic behavior, the basic activity of which is assumed to be the message exchange between two ports.* [Ding et al., 2008b]

Service Component Architecture 2/4

[Ding et al., 2008b]

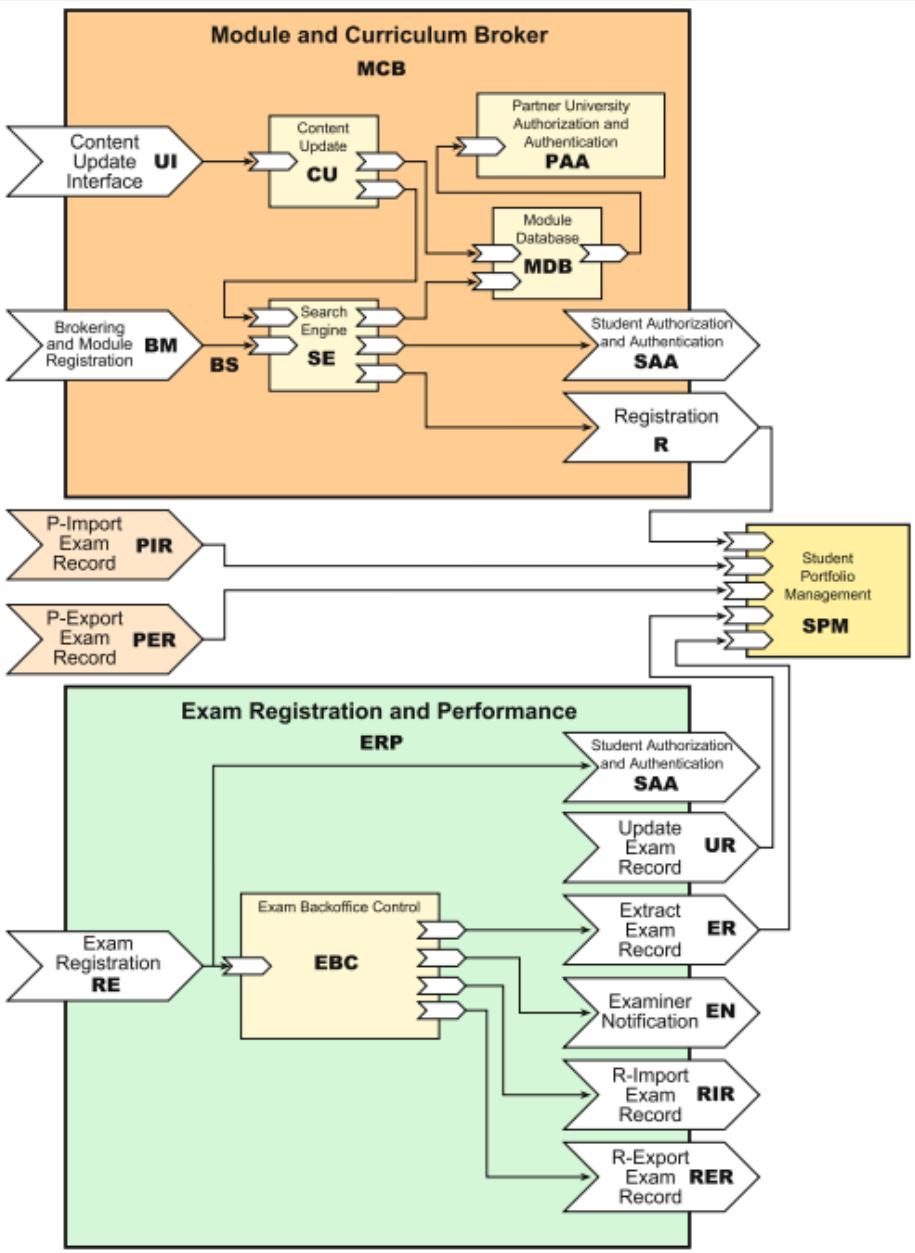


source : [Ding et al., 2008b]

un composite est un composant avec des sous-composants ($G \neq \emptyset$)

- promotion / wire
- simple liaison de port : pas vu de transformation
- pas vu de promotion de variables
- composition implicite de composants sur les ports

Service Component Architecture 3/4



Service Component Architecture 4/4

Commentaires

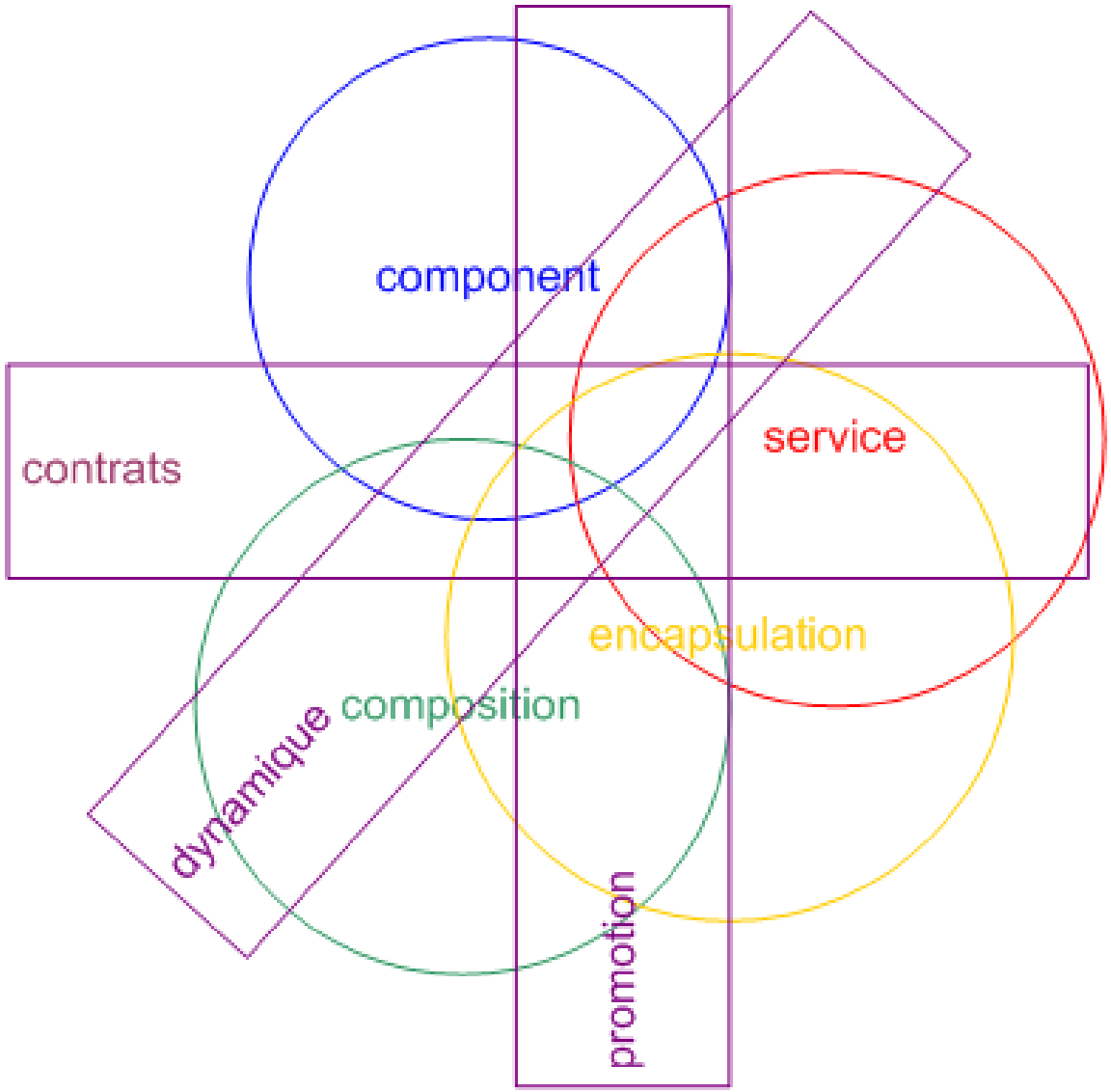
1. standard émergent CBSE/SOA (depuis 2007)
2. une spec $\Longrightarrow$ le modèle est ouvert, notamment pour la dynamique
3. plusieurs versions du modèle, ici [Fiadeiro et al., 2006] et [Ding et al., 2008b]
4. dans [Krämer, 2008]
   - orchestration de l'activité avec BPEL
   - signale contrats
     - behavioral contracts (OCL, CASL)
     - synchronization contract (not yet -> en fait [Ding et al., 2008b])
     - QoS contract (not yet)
   - component compatibility
5. pas de connecteurs
6. raffinement, encapsulation : par composition
7. reste un modèle de services non-hiérarchiques

Autres

1. Contract-Based Collaborative WS [Bai et al., 2007]
   - plutôt lié au test
2. A Foundational Theory of Contracts forMulti-party Service Composition [Bravetti and Zavattaro, 2008]
   - s'attaque à la dynamique
   - composition et raffinement de contrats
3. A Theory of Contracts for WebServices [Carpineti et al., 2006, Castagna et al., 2008]
   - algèbres de processus sans τ
   - sous-contrats
4. On the Observable Behaviour of Composite Components[Hennicker et al., 2010]
   - observabilité (et la promotion)
   - on arrive rapidement sur les problèmes "classiques" d'équivalence observable
   - petit exemple sur un proxy de conversion `CompressingProxy` modélisé par un composite à trois composants

Plan

Domaine complexe



Bilan

Typologie Promotion :

- cela dépend souvent de l'unité exhibée par les composants : port/interface/service, on trouvera donc
  - des promotions de ports
  - des délégations d'interfaces (ex : Fractal)
  - des connecteurs spécifiques, notamment Delegation connector (voir article Lau Exogenous Connectors for Software Components)
- la promotion de service (pas forcément de contrats) existe dans SCA

Contrats et composites (contrats/services/delegation)

- le terme 'contract' est surchargé
- on trouve plus de choses sur la dynamique que sur les contrats
- QoS contract (not yet)

Exemples

- Firewall (Vercors)
- ProxyConverter (Hennicker et Knapp)
- ATM (Enselme)
- Photocopieur, Lecteur Multimedia, Contrôleur de vitesse (ConFract)
- Détection d'erreur (Safety)
- CoCoME (FullContract)
- Achat en ligne (SCA/Ding)
- Gestion de stock (SCA/Fiadeiro)
- Gestion d'étudiants (SCA/Kramer)
- ...

Comparaison Kmelia

Non trivial Kmelia est seul sur son nuage
(services hiérarchiques, contrats fonctionnels et dynamiques)

- des modèles proches
  - Hidden dependencies [Enselme et al., 2004]
  - ConFract for composites [Collet et al., 2007a]
  - Contract-Based Component (Z) [Lee et al., 2005, Lee et al., 2002]
  - Contract-based WS composition (B) [Milanovic, 2005]
  - SCA [Fiadeiro et al., 2006, Ding et al., 2008b]
- les travaux autour de B, ceux de Zaremski vus précédemment restent d'actualité
- promotion de contrat
  - fonctionnel : ca reste original
  - dynamique : revu via l'équivalence observationnelle

Travaux Kmelia

Publi en cours : s'abstraire de Kmelia/ finaliser approche + progression

1. PB attaqué
2. principes
3. situer
4. version 1 modèle simple
5. vérification -> B ? ? ?
6. promotion et observation, observabilité et interface
7. modèle + ex + vérification
8. perspectives : évolution et substituabilité

Travaux Kmelia

Publi en cours : s'abstraire de Kmelia/ finaliser approche + progression

1. PB attaqué
   - composition verticale de composants
   - safe
   - scalable (spec + preuve)
2. principes
3. situer
4. version 1 modèle simple
5. vérification -> B ? ? ?
6. promotion et observation, observabilité et interface
7. modèle + ex + vérification
8. perspectives : évolution et substituabilité

Travaux Kmelia

Publi en cours : s'abstraire de Kmelia/ finaliser approche + progression

1. PB attaqué
2. principes
   - abstraction + encapsulation
   - bien indiquer qu'on ne propage pas vers les composants : pas de mise à plat pour la vérification l'effort se fait dans l'abstraction : assemblage -> composite
3. situer
4. version 1 modèle simple
5. vérification -> B ? ? ?
6. promotion et observation, observabilité et interface
7. modèle + ex + vérification
8. perspectives : évolution et substituabilité

Travaux Kmelia

Publi en cours : s'abstraire de Kmelia/ finaliser approche + progression

1. PB attaqué
2. principes
3. situer
   - agrégation – promotion de var Kml
   - délégation / binding – promotion serv + visib / connectors
4. version 1 modèle simple
5. vérification -> B ? ? ?
6. promotion et observation, observabilité et interface
7. modèle + ex + vérification
8. perspectives : évolution et substituabilité

Travaux Kmelia

Publi en cours : s'abstraire de Kmelia/ finaliser approche + progression

1. PB attaqué
2. principes
3. situer
4. version 1 modèle simple Hypothèse : travailler sur un modèle général - (Kmelia vient plus tard)
   - composant + services (owned/required)
   - (état) (contrat + dyn)
   - signaler sans détailler la structuration des services expliquer la dessus formalisme : à la Kml simple : composants + boîtes services
   - cas de promotion + exemples
5. vérification -> B ? ? ?
6. promotion et observation, observabilité et interface
7. modèle + ex + vérification
8. perspectives : évolution et substituabilité

Travaux Kmelia

Publi en cours : s'abstraire de Kmelia/ finaliser approche + progression

1. PB attaqué
2. principes
3. situer
4. version 1 modèle simple
5. vérification -> B ? ? ?
6. promotion et observation, observabilité et interface
7. modèle + ex + vérification
8. perspectives : évolution et substituabilité

finaliser modèle Kmelia cf annexe (vocabulaire figé ?)

Références I


Abi-Antoun, M., Aldrich, J., and Coelho, W. (2007).
A case study in re-engineering to enforce architectural control flow and data sharing.
J. Syst. Softw., 80(2) :240–264.


Aldrich, J. (2008).
Using types to enforce architectural structure.
In *WICSA '08 : Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*, pages 211–220, Washington, DC, USA. IEEE Computer Society.


Aldrich, J., Chambers, C., and Notkin, D. (2002a).
Architectural reasoning in ArchJava.
In *Proceedings ECOOP 2002*, volume 2374 of *LNCS*, pages 334–367. Springer Verlag.


Aldrich, J., Chambers, C., and Notkin, D. (2002b).
ArchJava : connecting software architecture to implementation.
In *ICSE '02 : Proceedings of the 24th International Conference on Software Engineering*, pages 187–197, New York, NY, USA. ACM.


André, D., Chiorean, D., Plasil, F., and Royer, J.-C. (2007).
Prague 2007 - workshop report.
Workshop Report.

Références II


Baduel, L., Baude, F., Caromel, D., Contes, A., Huet, F., Morel, M., and Quilici, R. (2006).
Grid Computing : Software Environments and Tools, chapter Programming, Deploying, Composing, for the Grid.
Springer-Verlag.


Bai, X., Wang, Y., Dai, G., Tsai, W.-T., and Chen, Y. (2007).
A framework for contract-based collaborative verification and validation of web services.
In Schmidt, H. W., Crnkovic, I., Heineman, G. T., and Stafford, J. A., editors, *CBSE*, volume 4608 of *Lecture Notes in Computer Science*, pages 258–273. Springer.


Barros, T., Cansado, A., Madelaine, E., and Rivera, M. (2007).
Model-checking distributed components : The vercors platform.
Electronic Notes in Theoretical Computer Science, 182 :3–16.


Benvenuti, L., Ferrari, A., Mangeruca, L., Mazzi, E., Passerone, R., and Sofronis, C. (2008).
A contract-based formalism for the specification of heterogeneous systems (invited).
In *FDL*, pages 142–147. IEEE.


Beugnard, A., Jézéquel, J.-M., Plouzeau, N., and Watkins, D. (1999).
Making components contract aware.
Computer, 32(7) :38–45.


Bravetti, M., Núñez, M., and Zavattaro, G., editors (2006).
Web Services and Formal Methods, Third International Workshop, WS-FM 2006 Vienna, Austria, September 8-9, 2006, Proceedings, volume 4184 of *Lecture Notes in Computer Science*. Springer.

Références III


Bravetti, M. and Zavattaro, G. (2008).
A foundational theory of contracts for multi-party service composition.
Fundam. Inform., 89(4) :451–478.


Brogi, A. (2010).
On the Potential Advantages of Exploiting Behavioural Information for Contract-based Service Discovery and Composition.
Journal of Logic and Algebraic Programming.


Brogi, A. and Corfini, S. (2007).
Behaviour-aware discovery of web service compositions.
Int. J. Web Service Res., 4(3) :1–25.


Bruneton, E., Coupaye, T., Leclercq, M., Quéma, V., and Stefani, J.-B. (2006).
The Fractal Component Model and Its Support in Java.
Software Practice and Experience, 36(11-12).


Bulej, L., Bureš, T., Děcký, M., Ježek, P., Parízek, P., Plášil, F., Poch, T., Rivierre, N., and Šerý, O. (2008).
CoCoME in Fractal, pages 357–387.
Volume 5153 of [Rausch et al., 2008].

Références IV


Bureš, T., Děcký, M., Hnětynka, P., Kofroň, J., Parízek, P., Plášil, F., Poch, T., Šerý, O., and Tůma, P. (2008).
CoCoME in SOFA, pages 388–417.
Volume 5153 of [Rausch et al., 2008].


Bures, T., Hnetynka, P., and Plasil, F. (2006).
Sofa 2.0 : Balancing advanced features in a hierarchical component model.
In *SERA '06 : Fourth IC on Software Engineering Research, Management and Applications*, pages 40–48. IEEE Computer Society.


Carpineti, S., Castagna, G., Laneve, C., and Padovani, L. (2006).
A formal account of contracts for web services.
In [Bravetti et al., 2006], pages 148–162.


Castagna, G., Gesbert, N., and Padovani, L. (2008).
A theory of contracts for web services.
SIGPLAN Not., 43(1) :261–272.


Chang, H. and Collet, P. (2005).
Vers la négociation de contrats dans les composants logiciels hiérarchiques.
L'OBJET, 11(1-2) :239–252.

Références V


Collet, P., Malenfant, J., Ozanne, A., and Rivierre, N. (2007a).
Composite contract enforcement in hierarchical component systems.
In Lumpe, M. and Vanderperren, W., editors, *Software Composition*, volume 4829 of *LNCS*, pages 18–33. Springer.


Collet, P., Ozanne, A., and Rivierre, N. (2007b).
Towards a versatile contract model to organize behavioral specifications.
In van Leeuwen, J., Italiano, G. F., van der Hoek, W., Meinel, C., Sack, H., and Plasil, F., editors, *SOFSEM (1)*, volume 4362 of *Lecture Notes in Computer Science*, pages 844–855. Springer.


Collet, P., Rousseau, R., Coupaye, T., and Rivierre, N. (2005).
A Contracting System for Hierarchical Components.
In Heineman, G. T., Crnkovic, I., Schmidt, H. W., Stafford, J. A., Szyperski, C. A., and Wallnau, K. C., editors, *CBSE*, volume 3489 of *Lecture Notes in Computer Science*, pages 187–202. Springer.


Crnkovic, I. and Larsson, M. (2000).
Component based software engineering - state of the art.
Technical report, Internal.


Ding, Z., Chen, Z., and Liu, J. (2008a).
A rigorous model of service component architecture.
Electr. Notes Theor. Comput. Sci., 207 :33–48.

Références VI


Ding, Z., Chen, Z., and Liu, J. (2008b).
A rigorous model of service component architecture.
Electr. Notes Theor. Comput. Sci., 207 :33–48.


Dong, W., Chen, Z., and Wang, J. (2007).
A contract-based approach to specifying and verifying safety critical systems.
Electron. Notes Theor. Comput. Sci., 176(2) :89–103.


Enselme, D., Florin, G., and Legond-Aubry, F. (2004).
Design by contract : analysis of hidden dependencies in component based application.
Journal of Object Technology, 3(4) :23–45.


Fenech, S., Pace, G. J., Okika, J. C., Ravn, A. P., and Schneider, G. (2009).
On the specification of full contracts.
Electr. Notes Theor. Comput. Sci., 253(1) :39–55.


Fiadeiro, J. L., Lopes, A., and Bocchi, L. (2006).
A formal approach to service component architecture.
In [Bravetti et al., 2006], pages 193–213.


Graf, S. and Quinton, S. (2007).
Contracts for bip : Hierarchical interaction models for compositional verification.
In *FORTE '07 : Proceedings of the 27th IFIP WG 6.1 international conference on Formal Techniques for Networked and Distributed Systems*, pages 1–18, Berlin, Heidelberg. Springer-Verlag.

Références VII


Hennicker, R., Janisch, S., and Knapp, A. (2010).
On the observable behaviour of composite components.
Electr. Notes Theor. Comput. Sci., 260 :125–153.


Krämer, B. J. (2008).
Component meets service : what does the mongrel look like ?
ISSE, 4(4) :385–394.


Lee, J.-H., Noh, H.-M., Yoo, C.-J., and Chang, O.-B. (2005).
Component contract-based formal specification technique.
In Gervasi, O., Gavrilova, M. L., Kumar, V., Laganà, A., Lee, H. P., Mun, Y., Taniar, D., and Tan, C. J. K., editors, *ICCSA (3)*, volume 3482 of *Lecture Notes in Computer Science*, pages 836–845. Springer.


Lee, J.-H., Yoo, C.-J., and Chang, O.-B. (2002).
Component contract-based interface specification technique using z.
International Journal of Software Engineering and Knowledge Engineering, 12(4) :453–.


Legond-Aubry, F., Enselme, D., and Florin, G. (2003).
Contrat d'assemblage de composants.
In *Conférence Française sur les Systèmes d'Exploitation (CFSE)*, pages 586–601.
NAT LIP6 SRC.

Références VIII


Milanovic, N. (2005).
Contract-based web service composition framework with correctness guarantees.
In Malek, M., Nett, E., and Suri, N., editors, *ISAS*, volume 3694 of *Lecture Notes in Computer Science*, pages 52–67. Springer.


Ozanne, A. (2007).
Interact : un modèle général de contrat pour la garantie des assemblage de composants et services.
PhD thesis, Université Pierre et Marie Curie (Paris VI).


Quinton, S. and Graf, S. (2008).
Contract-based verification of hierarchical systems of components.
In *SEFM '08 : Proceedings of the 2008 Sixth IEEE International Conference on Software Engineering and Formal Methods*, pages 377–381, Washington, DC, USA. IEEE Computer Society.


Rausch, A., Reussner, R., Mirandola, R., and Plasil, F., editors (2008).
The Common Component Modeling Example : Comparing Software Component Models, volume 5153 of *LNCS*.
Springer, Heidelberg.


Zaremski, A. M. and Wing, J. M. (1997).
Specification matching of software components.
ACM Transaction on Software Engeniering Methodolology, 6(4) :333–369.