UNIVERSITÉ DE NANTES

cnrs dépasser les frontières

Reconfigurations d’architectures à composants

Arnaud Lanoix

Séminaire interne AeLoS – 27 juin 2013

lina LABORATOIRE D'INFORMATIQUE DE NANTES ATLANTIQUE

Historique, contexte …

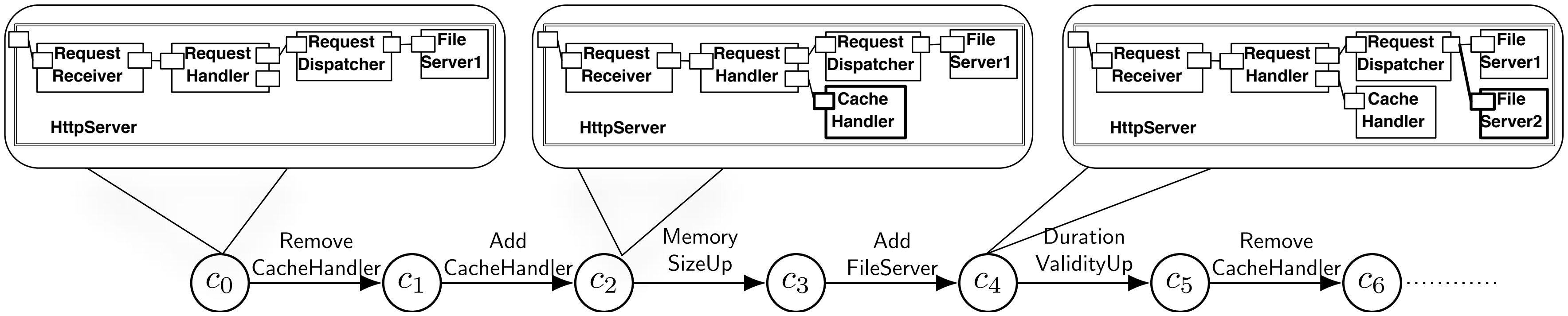
- Travail avec **Olga Kouchnarenko** et **Julien Dormoy** (Vesontio@FEMTO-ST, Besançon) depuis 2010.

- **Publications :**
  - **[AFADL 10]** J. Dormoy, Al. Dreyfus, and O. Kouchnarenko. *EVA4Fractal : adaptation de composants Fractal basée sur des événements.*
  - **[AICCSA 10]** J. Dormoy and O. Kouchnarenko. *Event-based Adaptation Policies for Fractal Components*
  - **[FACS 10]** J. Dormoy, O. Kouchnarenko, and A. Lanoix. *Using temporal logic for dynamic reconfigurations of components*
  - **[FESCA 11]** A. Lanoix, J. Dormoy, and O. Kouchnarenko. *Combining Proof and Model-checking to Validate Reconfigurable Architectures*
  - **[FACS 11]** J. Dormoy, O. Kouchnarenko, and A. Lanoix. *Runtime Verification of Temporal Patterns for Dynamic Reconfigurations of Components*
  - **[FM 12]** J. Dormoy, O. Kouchnarenko, and A. Lanoix. *When Structural Refinement of Components Keeps Temporal Properties Over Reconfigurations*
  - ***[FACS 13 ???] en cours de rédaction***

- **Thèse de Julien soutenue en décembre 2011**

Idées générales / problématique

- Etre capable d'**exprimer** puis de **vérifier** des propriétés sur une séquence de reconfigurations dynamiques
- Utiliser ces propriétés pour **guider le choix** d'une reconfiguration à appliquer

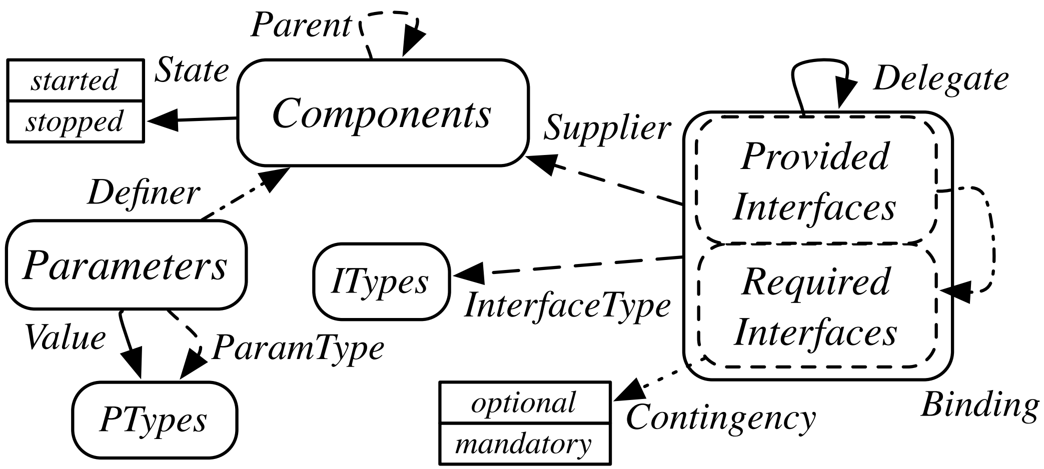


- Etudier l'impact de l'**évolution** de l'architecture
  - remplacement d'un composant simple par un composite
  - …

Un modèle pour une architecture de composants

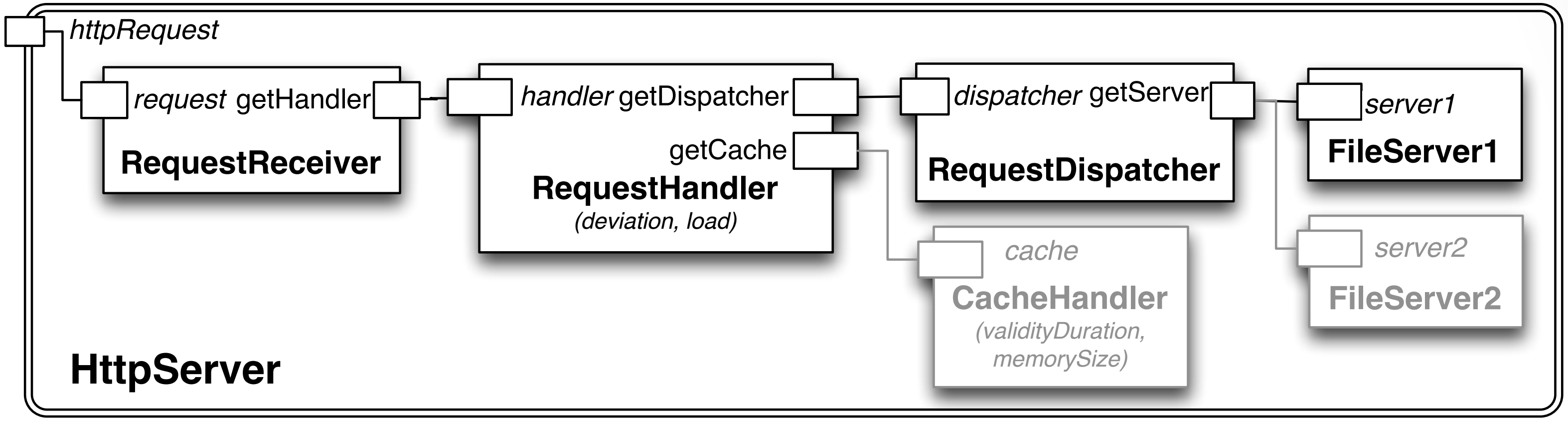
Une configuration *c = <Elem,Rel>*

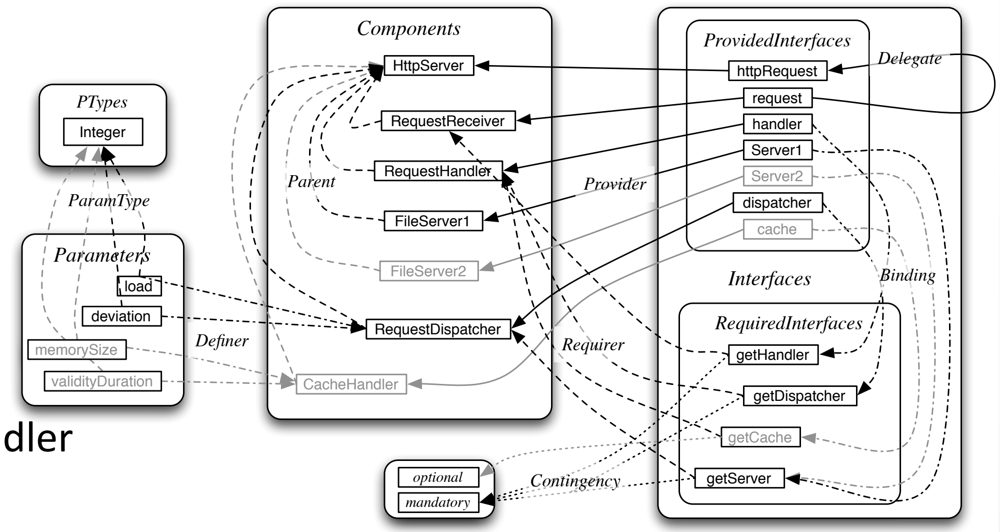
- *Elem = Components* $\uplus$ *Interfaces* $\uplus$ *Parameters* $\uplus$ *Types*
- *Rel = Container* $\uplus$ *ContainerType* $\uplus$ *Parent* $\uplus$ *Binding* $\uplus$ *Delegate* $\uplus$ *State* $\uplus$ *Value*



= Capture la sémantique « architecturale » de FRACTAL, par exemple

Exemple





Reconfigurations possible :

- AddCacheHandler /RemoveCacheHandler
- AddFileServer / removeFileServer
- MemorySizeUp MemorySizeDown
- DurationValidityUp / DurationValidityDown

Configurations consistantes

$$\forall c.(c \in Components \Rightarrow (\exists ip.(ip \in ProvInterfaces \land Container(ip) = c))) \quad (CC.1)$$

$$\forall c, c' \in Components.(c \neq c' \land (c, c') \in Parent \Rightarrow \forall\ p.(p \in Parameters \Rightarrow Container(p) \neq c')) \quad (CC.2)$$

$$\forall c, c' \in Components.((c, c') \in Parent^{+} \Rightarrow c \neq c') \quad (CC.3)$$

$$\begin{array}{l}\forall ip \in ProvInterfaces,\\ \forall ir \in ReqInterfaces\end{array} . \left(Binding(ip) = ir \Rightarrow \begin{array}{l} InterfaceType(ip) = InterfaceType(ir) \\ \land Container(ip) \neq Container(ir) \end{array} \right) \quad (CC.4)$$

$$\begin{array}{l}\forall ip \in ProvInterfaces,\\ \forall ir \in ReqInterfaces\end{array} . \left(Binding(ip) = ir \Rightarrow \exists c \in Components. \left(\begin{array}{l} (Container(ip), c) \in Parent \\ \land (Container(ir), c) \in Parent \end{array} \right) \right) \quad (CC.5)$$

$$\begin{array}{r}\forall ip \in ProvInterfaces,\\ \forall ir \in ReqInterfaces, \forall id \in Interfaces\end{array} . \left(Binding(ip) = ir \Rightarrow \begin{array}{l} Delegate(ip) \neq id \\ \land Delegate(ir) \neq id \end{array} \right) \quad (CC.6)$$

$$\forall i, i' \in Interfaces. \left(Delegate(i) = i' \Rightarrow \begin{array}{l} \forall ip.(ip \in ProvInterfaces \Rightarrow Binding(ip) \neq i) \\ \land \forall ir.(ir \in ReqInterfaces \Rightarrow Binding(i) \neq ir) \end{array} \right) \quad (CC.7)$$

$$\forall i, i' \in Interfaces.(Delegate(i) = i' \land i \in ProvInterfaces \Rightarrow i' \in ProvInterfaces) \quad (CC.8)$$

$$\forall i, i' \in Interfaces.(Delegate(i) = i' \land i \in ReqInterfaces \Rightarrow i' \in ReqInterfaces) \quad (CC.9)$$

$$\forall i, i' \in Interfaces. \left(Delegate(i) = i' \Rightarrow \begin{array}{l} InterfaceType(i) = InterfaceType(i') \\ \land\ (Container(i), Container(i')) \in Parent \end{array} \right) \quad (CC.10)$$

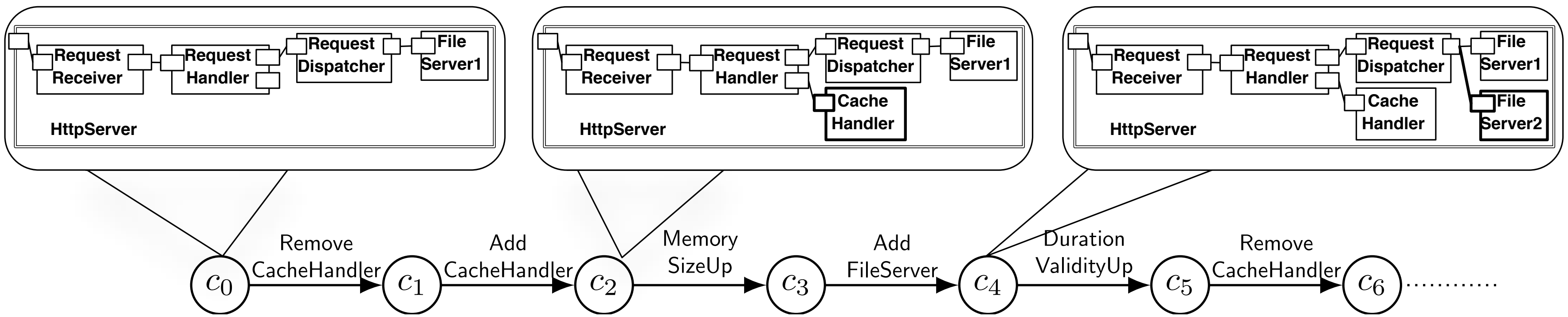
$$\forall i, i', i" \in Interfaces. \left(\begin{array}{l} (Delegate(i) = i' \land Delegate(i) = i" \Rightarrow i' = i") \\ \land (Delegate(i) = i" \land Delegate(i') = i" \Rightarrow i = i') \end{array} \right) \quad (CC.11)$$

$$\forall ir \in ReqInterfaces. \left(\begin{array}{l} State(Supplier(ir)) = started \\ \land Contingency(ir) = mandatory \end{array} \Rightarrow \exists i \in Interfaces. \left(\begin{array}{l} Binding(i) = ir \\ \lor \\ Delegate(i) = ir \end{array} \right) \right) \quad (CC.12)$$

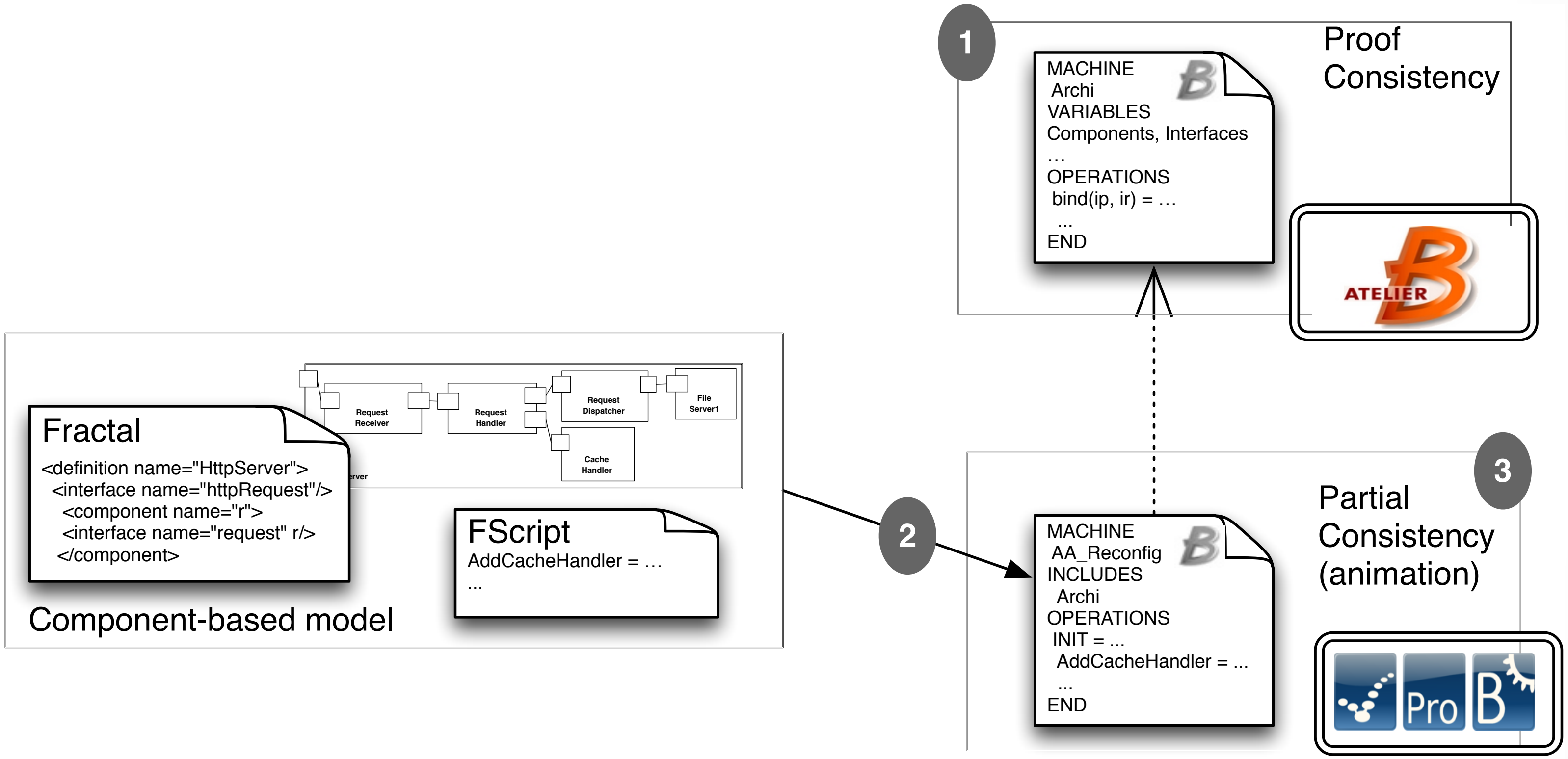
Un modèle des séquences de reconfigurations

= un système de transition S = ⟨C, C0, R, →⟩

- C = {c, c1, c2, . . .} : ensemble de configurations **consistantes**
- C0 ⊆ C : des configurations initiales
- **R : ensemble des opérations de reconfiguration**
- **→ : relation de reconfiguration**



Validation du modèle grâce aux outils B



Validation du modèle (2)

```
MACHINE
 Archi
VARIABLES
 Components, Interfaces, ProvInterfaces, ReqInterfaces, Supplier, Parent, Binding, ...
INVARIANT
 ProvInterfaces ⊆ Interfaces
 ∧ ReqInterfaces ⊆ Interfaces
 ∧ ProvInterfaces ∪ ReqInterfaces = Interfaces ∧ ProvInterfaces ∩ ReqInterfaces = ∅
 ∧ Supplier ∈ Interfaces → Components
 ∧ Parent ∈ Components ↔ Components
 ∧ Binding ∈ ProvInterfaces ⇸ ReqInterfaces
 ∧ closure1(Parent) ∩ id(Components) = ∅                        // CC.3        CC.4 ∧ CC.5
 ∧ ∀ (ip, ir ).( ip ↦ ir ∈ Binding ⇒ Provider(ip) ≠ Requirer(ir) ∧ Parent(Supplier(iprov)) = Parent(Supplier(ireq)) )
 ∧ ...
OPERATIONS
bind(ip, ir) =
PRE
    ip ∈ ProvInterfaces ∧ ir ∈ ReqInterfaces ∧ ip↦ir ∉ Binding
    ∧ ip ∉ dom(Binding) ∧ ip ∉ dom(Delegate) ∧ ir ∉ dom(Delegate)
THEN
    Binding(ip) := ir
END ;
 ...
END
```

Validation des contraintes de consistance par preuve

```
MACHINE
 Reconfig
INCLUDES
 Archi
OPERATIONS
INIT =
 BEGIN
  Components := { HttpServer, RequestReceiver, RequestHandler, CacheHandler, RequestDispatcher,
                  FileServer1, FileServer2 }
  || ProvInterfaces := { httpRequest, request, handler, cache, dispatcher, server1, server2 }
  || ReqInterfaces := { getHandler, getDispatcher, getCache, getServer }
  || Parent := { RequestReceiver↦HttpServer, RequestHandler↦HttpServer, CacheHandler↦HttpServer,
                 RequestDispatcher↦HttpServer, FileServer1↦HttpServer }
  || Binding := { handler↦getHandler, cache↦getCache, dispatcher↦getDispatcher, server1↦getServer }
  ...
 END ;
 AddCacheHandler =
 BEGIN
   instantiate (CacheHandler) ;
  add(CacheHandler, HttpServer) ;
  bind(cache, getCache) ;
   start (CacheHandler)
 END ;
 ...
END
```

Vérification partielle par animation/model-checking

FTPL : une logique temporelle pour des séquences de reconfigurations

- Inspirée d'une extension temporelle de JML, etc.

$$
\begin{array}{lll}
\langle temp \rangle & ::= & \textbf{after } \langle event \rangle \ \langle temp \rangle \ \mid \ \textbf{before } \langle event \rangle \ \langle trace \rangle \\
 & \mid & \langle trace \rangle \ \textbf{until } \langle event \rangle \\
\langle trace \rangle & ::= & \textbf{always } \mathrm{config} \ \mid \ \textbf{eventually } \mathrm{config} \\
 & \mid & \langle trace \rangle \wedge \langle trace \rangle \ \mid \ \langle trace \rangle \vee \langle trace \rangle \\
\langle event \rangle & ::= & ope \ \textbf{normal} \ \mid \ ope \ \textbf{exceptional} \ \mid \ ope \ \textbf{terminates}
\end{array}
$$

For the events:

$$
\begin{array}{lll}
[\![\sigma(i) \models ope\ \textbf{normal}]\!] & = & \begin{cases} \top & if\ i > 0 \wedge \sigma(i-1) \neq \sigma(i) \wedge \sigma(i-1) \overset{ope}{\rightarrow} \sigma(i) \in \rightarrow \\ \bot & otherwise \end{cases} \\
[\![\sigma(i) \models ope\ \textbf{exceptional}]\!] & = & \begin{cases} \top & if\ i > 0 \wedge \sigma(i-1) = \sigma(i) \wedge \sigma(i-1) \overset{ope}{\rightarrow} \sigma(i) \in \rightarrow \\ \bot & otherwise \end{cases} \\
[\![\sigma(i) \models ope\ \textbf{terminates}]\!] & = & \begin{cases} \top & if\ [\![\sigma(i) \models ope\ \textbf{normal}]\!] = \top \vee [\![\sigma(i) \models ope\ \textbf{exceptional}]\!] = \top \\ \bot & otherwise \end{cases}
\end{array}
$$

For the trace properties:

$$
\begin{array}{lll}
[\![\sigma \models \textbf{always}\ cp]\!] & = & \begin{cases} \top & if\ \forall i.(i \geqslant 0 \ \Rightarrow \ [\![\sigma(i) \models cp]\!] = \top) \\ \bot & otherwise \end{cases} \\
[\![\sigma \models \textbf{eventually}\ cp]\!] & = & \begin{cases} \top & if\ \exists i.(i \geqslant 0 \ \wedge \ [\![\sigma(i) \models cp]\!] = \top) \\ \bot & otherwise \end{cases} \\
[\![\sigma \models trp_1 \wedge trp_2]\!] & = & [\![\sigma \models trp_1]\!] \wedge [\![\sigma \models trp_2]\!] \\
[\![\sigma \models trp_1 \vee trp_2]\!] & = & [\![\sigma \models trp_1]\!] \vee [\![\sigma \models trp_2]\!]
\end{array}
$$

For the temporal properties:

$$
\begin{array}{lll}
[\![\sigma \models \textbf{after}\ e\ tpp]\!] & = & \begin{cases} \top & if\ \forall i.(i \geqslant 0 \wedge [\![\sigma(i) \models e]\!] = \top \Rightarrow [\![\sigma_i \models tpp]\!] = \top) \\ \bot & otherwise \end{cases} \\
[\![\sigma \models \textbf{before}\ e\ trp]\!] & = & \begin{cases} \top & if\ \forall i.(i > 0 \wedge [\![\sigma(i) \models e]\!] = \top \Rightarrow [\![\sigma_0^{i-1} \models trp]\!] = \top) \\ \bot & otherwise \end{cases} \\
[\![\sigma \models trp\ \textbf{until}\ e]\!] & = & \begin{cases} \top & if\ \exists i.(i > 0 \wedge [\![\sigma(i) \models e]\!] = \top \wedge [\![\sigma_0^{i-1} \models trp]\!] = \top) \\ \bot & otherwise \end{cases}
\end{array}
$$

Traduction vers LTL
+ model-checking

Exemples de propriétés FTPL

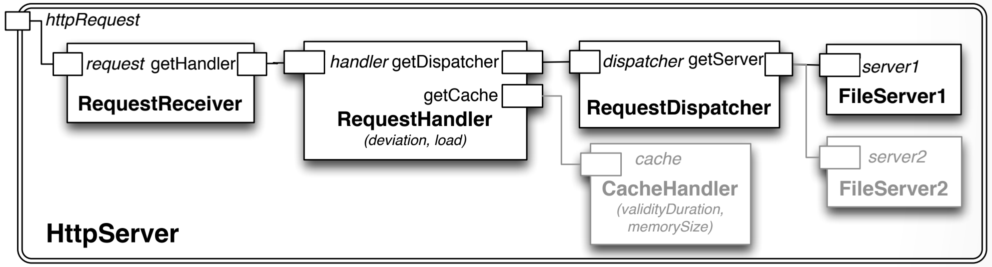
- Pour une configuration :

CacheConnected := $\exists$ *cache,getCache* $\in$ *Interfaces* .(*Provider(cache)=CacheHandler*

$\wedge$ *Requirer(getCache) = RequestHandler*

$\wedge$ *Binding(cache) = getCache*)

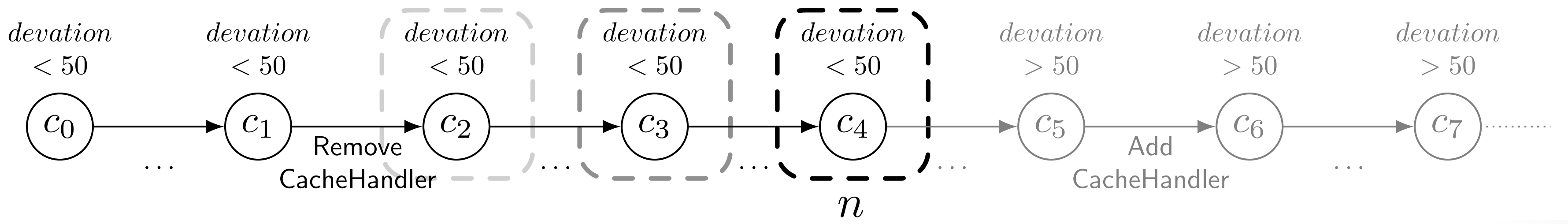
- pour l'ensemble du système :

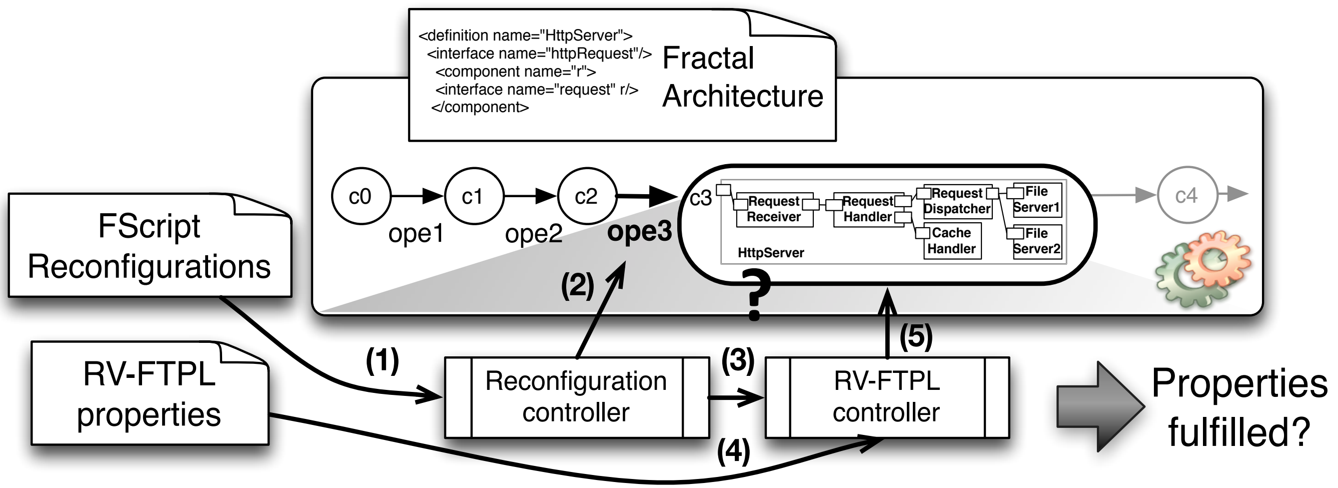


after `AddCacheHandler` **normal always** *CacheConnected*

after `RemoveCacheHandler` **terminates**
((eventually deviation > 50) **until** `AddCacheHandler` **terminates)**

Evaluation à l'exécution de propriétés FTPL





RV-FTPL : « nouvelle » sémantique pour FTPL

- B4 = {⊥, ⊥p, ⊤p, ⊤}
  - ⊥ : faux
  - ⊥p : « plutôt » faux
  - ⊤p : « plutôt » vrai
  - ⊤ : vrai

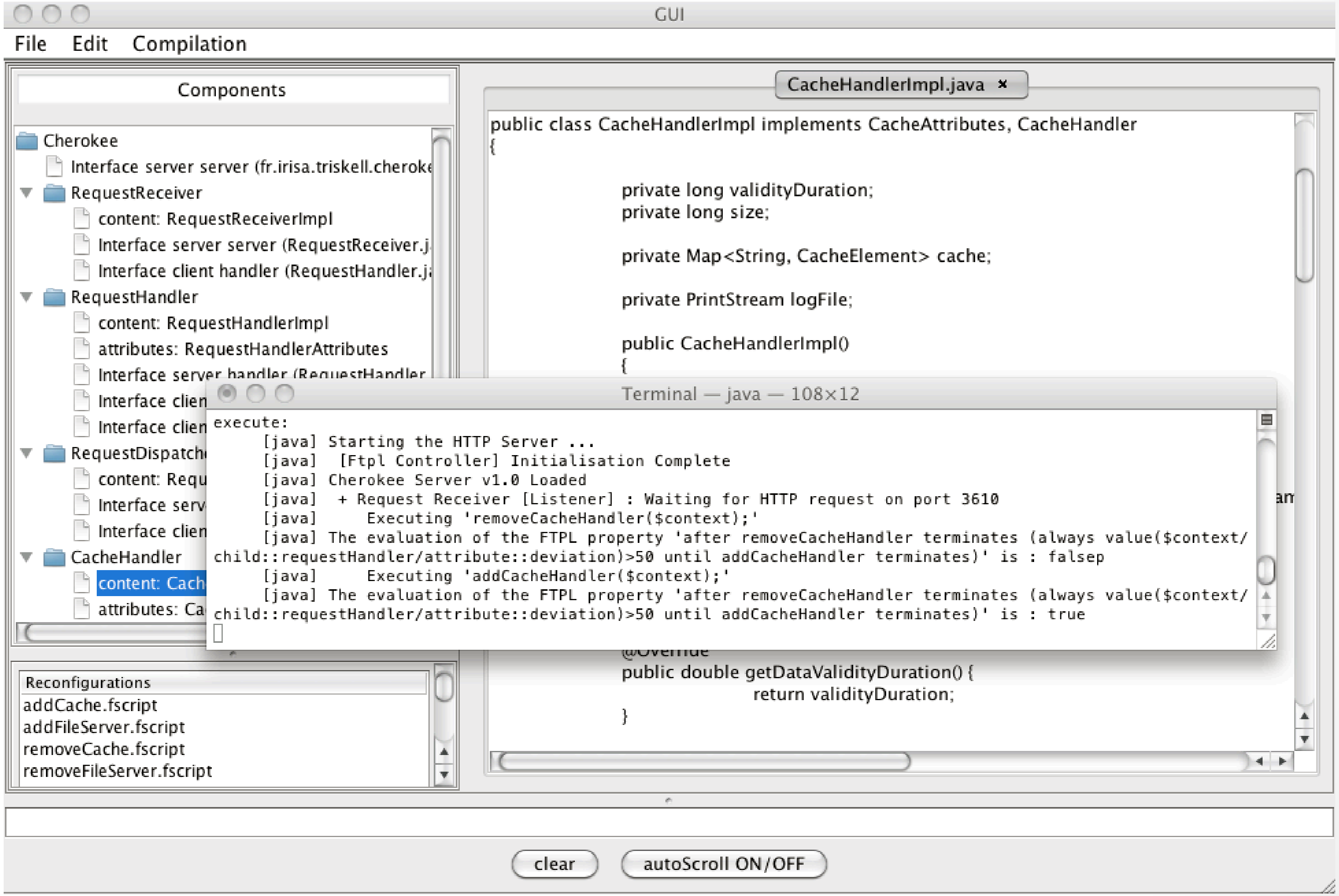
$$
\begin{aligned}
[\sigma_0^n(i) \models conf]_{rv} &= \begin{cases} \top \; if \; [\sigma_0^n(i) \models conf] = \top \\ \bot \; otherwise \end{cases} \\
[\sigma_0^n(i) \models ope\ \mathbf{normal}]_{rv} &= \begin{cases} \top \; if \; 0 < i \leqslant n \; \wedge \sigma_0^n(i-1) \neq \sigma_0^n(i) \\ \qquad \wedge\, \sigma_0^n(i-1) \overset{ope}{\to} \sigma_0^n(i) \in \to \\ \bot \; otherwise \end{cases} \\
[\sigma_0^n(i) \models ope\ \mathbf{exceptional}]_{rv} &= \begin{cases} \top \; if \; 0 < i \leqslant n \; \wedge \sigma_0^n(i-1) = \sigma_0^n(i) \\ \qquad \wedge\, \sigma_0^n(i-1) \overset{ope}{\to} \sigma_0^n(i) \in \to \\ \bot \; otherwise \end{cases} \\
[\sigma_0^n(i) \models ope\ \mathbf{terminates}]_{rv} &= \begin{cases} \top \; if \; ope\ \mathbf{normal} \;\vee\; ope\ \mathbf{exceptional} \\ \bot \; otherwise \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
[\sigma_0^n \models \mathbf{always}\ conf]_{rv} &= \begin{cases} \bot \;\; if \; \exists i.(0 \leqslant i \leqslant n \; \wedge [\sigma_0^n(i) \models conf]_{rv} = \bot) \\ \top^p \; otherwise \end{cases} \\
[\sigma_0^n \models \mathbf{eventually}\ conf]_{rv} &= \begin{cases} \top \;\; if \; \exists i.(0 \leqslant i \leqslant n \; \wedge [\sigma_0^n(i) \models conf]_{rv} = \top) \\ \bot^p \; otherwise \end{cases} \\
[\sigma_0^n \models trace_1 \wedge trace_2]_{rv} &= [\sigma_0^n \models trace_1]_{rv} \sqcap [\sigma_0^n \models trace_2]_{rv} \\
[\sigma_0^n \models trace_1 \vee trace_2]_{rv} &= [\sigma_0^n \models trace_1]_{rv} \sqcup [\sigma_0^n \models trace_2]_{rv}
\end{aligned}
$$

$$
\begin{aligned}
[\sigma_0^n \models \mathbf{after}\ event\ temp]_{rv} &= \begin{cases} \top^p \; if \; \forall i.(0 \leqslant i \leqslant n \; \wedge [\sigma_0^n(i) \models event]_{rv} = \top \\ \qquad \Rightarrow [\sigma_i^n \models temp]_{rv} = \top) \vee \forall i.(0 < i \leqslant n \\ \qquad \Rightarrow [\sigma_0^n(i) \models event]_{rv} = \bot) \\ \bot \;\; if \; \exists i.(0 \leqslant i \leqslant n \; \wedge [\sigma_0^n(i) \models event]_{rv} = \top \\ \qquad \wedge [\sigma_i^n \models temp]_{rv} = \bot) \\ \bot^p \; if \; \exists i.(0 \leqslant i \leqslant n \; \wedge [\sigma_0^n(i) \models event]_{rv} = \top \\ \qquad \wedge [\sigma_i^n \models temp]_{rv} = \bot^p) \end{cases} \\
[\sigma_0^n \models \mathbf{before}\ event\ trace]_{rv} &= \begin{cases} \top^p \; if \; \forall i.(0 < i \leqslant n \; \wedge [\sigma_0^n(i) \models event]_{rv} = \top \\ \qquad \Rightarrow [\sigma_0^{i-1} \models trace]_{rv} \in \{\top, \top^p\}) \vee \\ \qquad \forall i.(0 < i \leqslant n \Rightarrow [\sigma_0^n(i) \models event]_{rv} = \bot) \\ \bot \;\; if \; \exists i.(0 < i \leqslant n \; \wedge [\sigma_0^n(i) \models event]_{rv} = \top \\ \qquad \wedge [\sigma_0^{i-1} \models trace]_{rv} \in \{\bot, \bot^p\}) \end{cases} \\
[\sigma_0^n \models trace\ \mathbf{until}\ event]_{rv} &= \begin{cases} \top^p \; if \; \forall i.(0 < i \leqslant n \; \wedge [\sigma_0^n(i) \models event]_{rv} = \top \\ \qquad \Rightarrow [\sigma_0^{i-1} \models trace]_{rv} \in \{\top, \top^p\}) \\ \bot \;\; if \; ([\sigma_0^n \models trace]_{rv} = \bot) \vee \\ \qquad (\exists i.(0 < i \leqslant n \; \wedge [\sigma_0^n(i) \models event]_{rv} = \top \\ \qquad \wedge [\sigma_0^{i-1} \models trace]_{rv} = \bot^p) \\ \bot^p \; if \; \forall i.(0 < i \leqslant n \; \Rightarrow [\sigma_0^n(i) \models event]_{rv} = \bot \end{cases}
\end{aligned}
$$

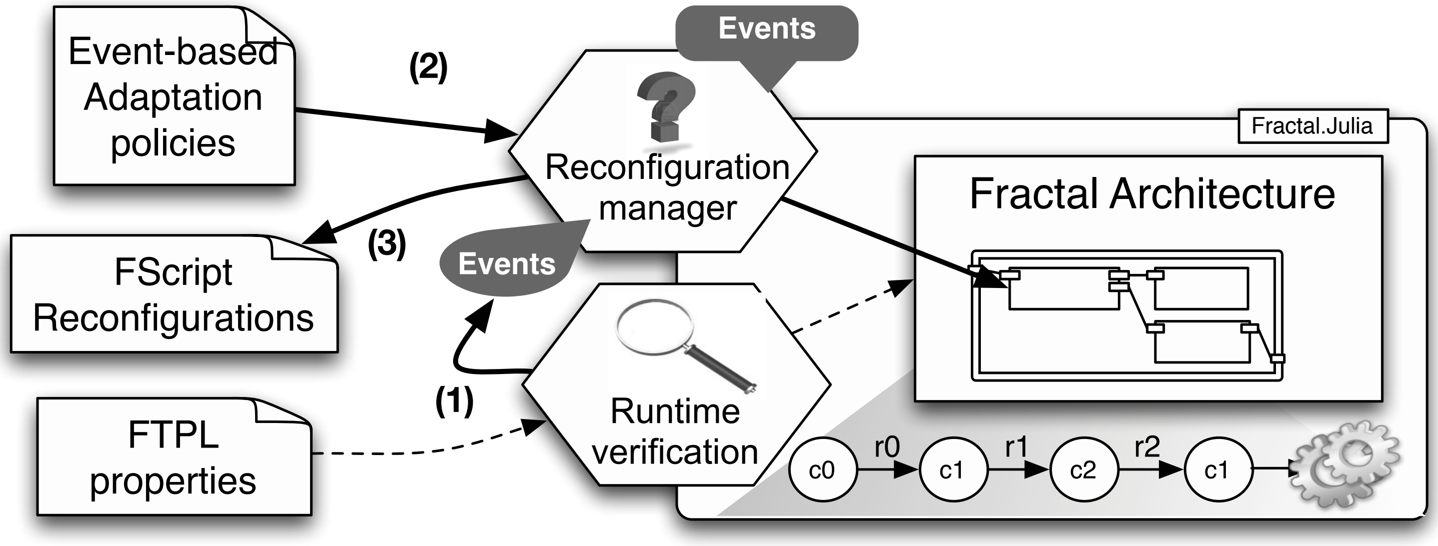
Intéret : On n'a pas besoin d'avoir un historique

Expérimentations



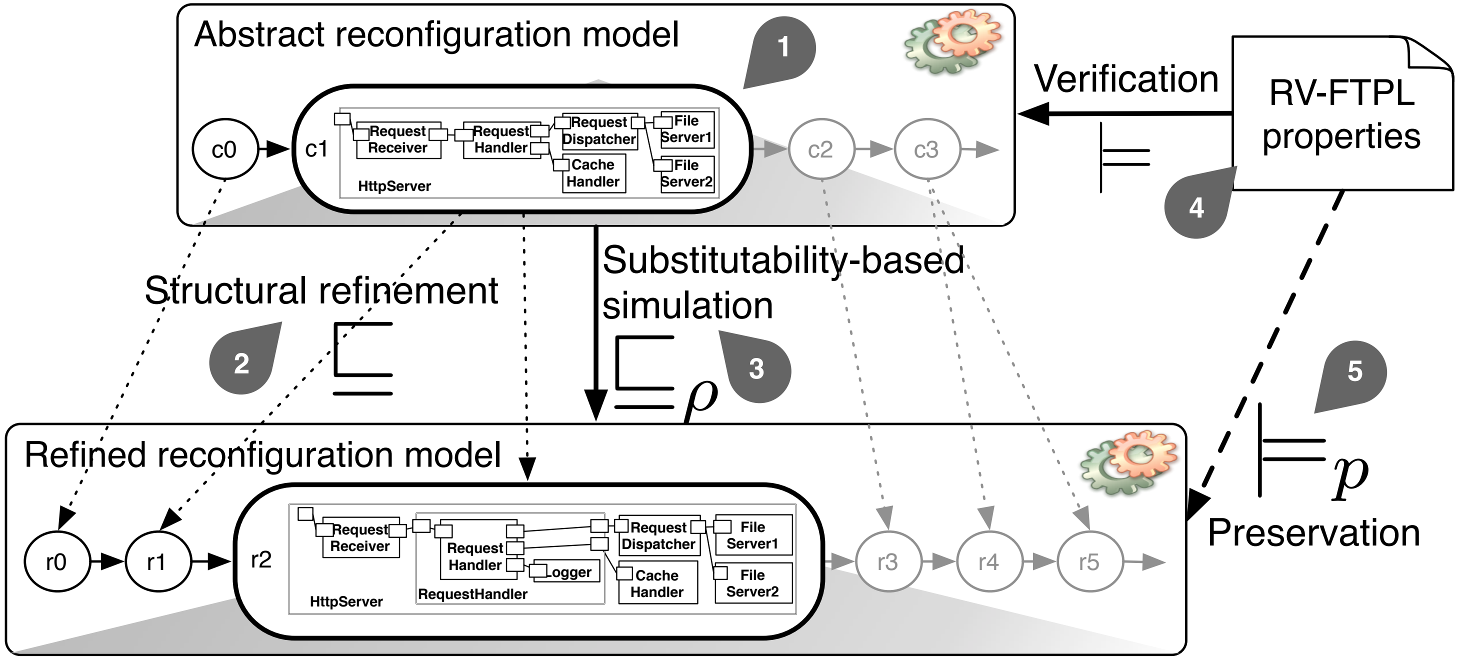
Guider le choix des reconfigurations ??

- Les réponses ⊥p / ⊤p peuvent guider le choix d'une reconfiguration

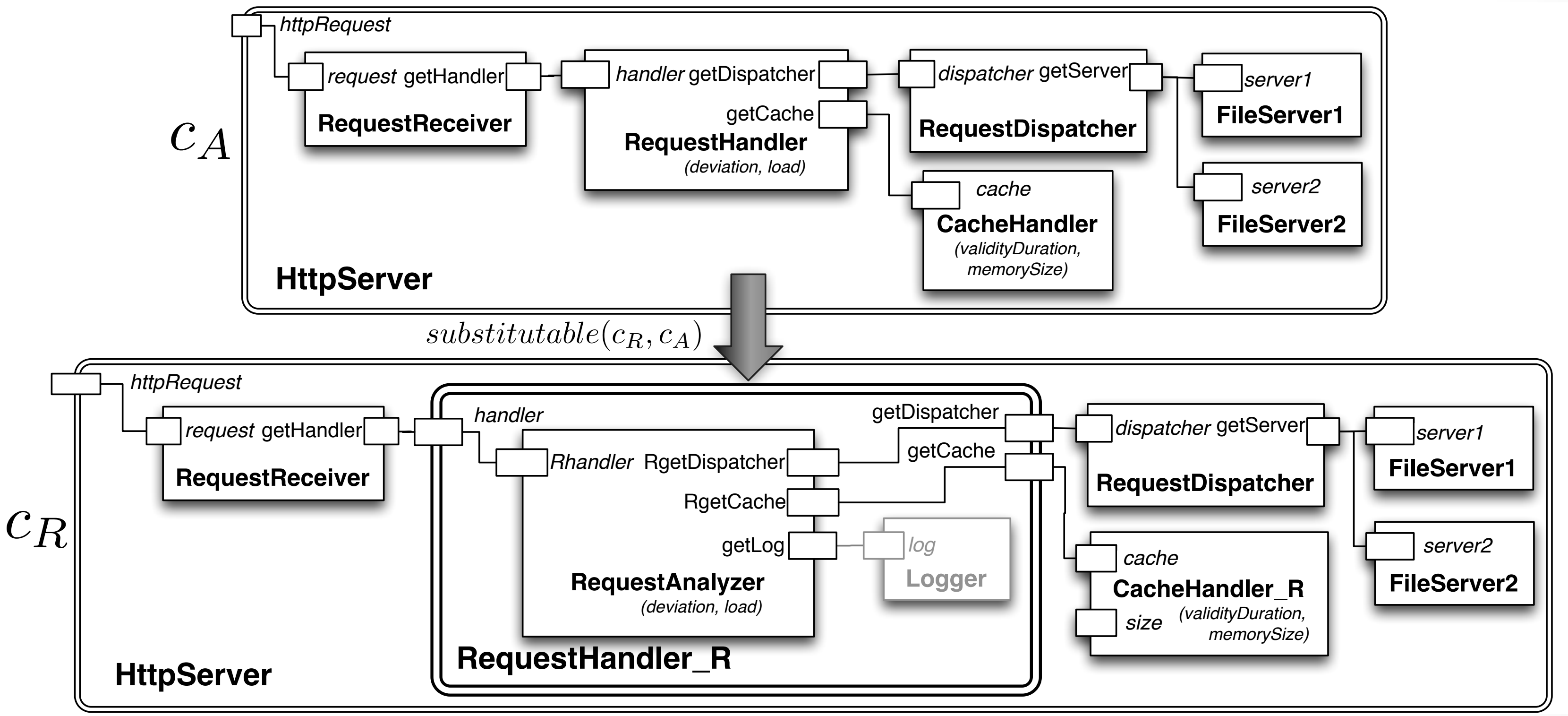


Evolution de l'architecture

= substitution/raffinement de certains composants

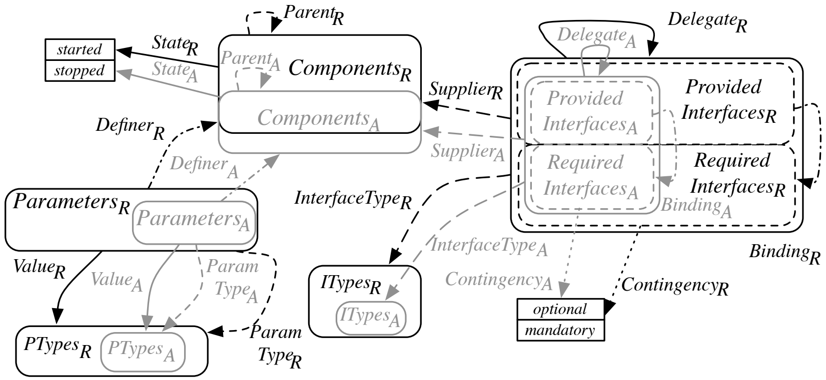


Exemple de raffinement structurel



Raffinement structurel

- Fonction de substitution :
  - Subst : ComponentA -> ComponentR



$$Parameters_A \subseteq Parameters_R \wedge PTypes_A \subseteq PTypes_R \quad (SC.1)$$

$$\forall p.(p \in Parameters_A \Rightarrow (ParamType_A(p) = ParamType_R(p) \wedge Value_A(p) = Value_R(p))) \quad (SC.2)$$

$$\forall p.\left(\begin{array}{r} p \in Parameters_R \\ \setminus Parameters_A \end{array} \Rightarrow \begin{array}{r} Container_R(p) \in \\ Components_R \\ \setminus Components_A \end{array} \wedge \forall c_A.\left(\begin{array}{r} c_A \in Components_A \setminus Components_R \Rightarrow \\ Subst(c_A) \neq Container_R(p) \end{array} \right)\right) \quad (SC.3)$$

$$\forall c.(c \in Components_A \cap Components_R \Rightarrow State_A(c) = State_R(c)) \quad (SC.4)$$

$$\forall c \in Components_A \cap Components_R, \forall p \in Parameters_A.(Container_A(p) = c \Rightarrow Container_R(p) = c) \quad (SC.5)$$

$$\forall c \in Components_A \cap Components_R, \forall i \in Interfaces_A.(Container_A(i) = c \Rightarrow Container_R(i) = c) \quad (SC.6)$$

$$\begin{array}{r} \forall c \in Components_A \\ \cap Components_R, \\ \forall c' \in Components_A \end{array} . \left((c, c') \in Parent_A \Rightarrow \left(\begin{array}{l} c' \in Components_R \wedge (c, c') \in Parent_R \\ \vee \\ c' \notin Components_R \wedge \\ \quad \exists c'' \in Components_R \setminus Components_A . \\ \quad (Subst(c') = c'' \wedge (c, c'') \in Parent_R) \end{array} \right)\right) \quad (SC.7)$$

$$\forall c_A.\left(c_A \in \begin{array}{r} Components_A \\ \setminus Components_R \end{array} \Rightarrow \left(\exists c_R \in \begin{array}{r} Components_R \\ \setminus Components_A \end{array} . (Subst(C_A) = c_R) \right)\right) \quad (SC.8)$$

$$\begin{array}{r} \forall c_A \in Components_A \setminus Components_R, \forall c_R \in Components_R \setminus Components_A \\ .(Subst(c_A) = c_R \Rightarrow State_A(c_A) = State_R(c_R)) \end{array} \quad (SC.9)$$

$$\begin{array}{r} \forall c_A \in Components_A \setminus Components_R, \forall c_R \in Components_R \setminus Components_A \\ .(Subst(c_A) = c_R \Rightarrow (\forall i \in Interfaces_A.Container_A(i) = c_A \Rightarrow Container_R(i) = c_R)) \end{array} \quad (SC.10)$$

$$\begin{array}{r} \forall c_A \in Components_A \\ \setminus Components_R, \\ \forall c_R \in Components_R \\ \setminus Components_A, \\ \forall p \in Parameters_A \end{array} . \left(\begin{array}{r} Subst(c_A) = c_R \\ \wedge Container_A(p) = c_A \\ \Rightarrow \end{array} \left(\begin{array}{l} Container_R(p) = c_R \\ \vee \\ \exists c'_R \in Components_R \setminus Components_A . \\ \quad ((c'_R, c_R) \in Parent^+ \wedge Container_R(p) = c'_R) \end{array} \right)\right) \quad (SC.11)$$

$$\begin{array}{r} \forall c_A \in Components_A \\ \setminus Components_R, \\ \forall c_R \in Components_R \\ \setminus Components_A, \\ \forall c'_A \in Components_A \end{array} . \left(\begin{array}{r} Subst(c_A) = c_R \\ \wedge (c_A, c'_A) \in Parent_A \\ \Rightarrow \end{array} \left(\begin{array}{l} c'_A \in Components_R \wedge (c_R, c'_A) \in Parent_R \\ \vee \\ c'_A \notin Components_R \wedge \\ \exists c'_R \in Components_R \setminus Components_A . \\ ((c_R, c'_R) \in Parent \wedge Subst(c'_A) = c'_R) \end{array} \right)\right) \quad (SC.12)$$

$$\begin{array}{r} \forall c_R \in Components_R \\ \setminus Components_A, \\ \forall c_A \in Components_A \\ \setminus Components_R \end{array} . \left(\begin{array}{l} Subst(c_A) \neq c_R \Rightarrow \\ \exists c'_R \in Components_R \setminus Components_A . \\ \quad ((c_R, c'_R) \in Parent_R) \end{array} \right) \quad (SC.13)$$

$$\begin{array}{r} ITypes_A \subseteq ITypes_R \wedge Interfaces_A \subseteq Interfaces_R \\ \wedge ProvInterfaces_A \subseteq ProvInterfaces_R \wedge ReqInterfaces_A \subseteq ReqInterfaces_R \end{array} \quad (SC.14)$$

$$\forall i.(i \in Interfaces_A \Rightarrow InterfaceType_A(i) = InterfaceType_R(i)) \quad (SC.15)$$

$$\forall i.(i \in ReqInterfaces_A \Rightarrow Contingency_A(i) = Contingency_R(i)) \quad (SC.16)$$

$$\forall i.\left(\begin{array}{l} i \in ReqInterfaces_R \\ \setminus ReqInterfaces_A \end{array} \Rightarrow \begin{array}{r} Container_R(i) \in \\ Components_R \\ \setminus Components_A \end{array} \wedge \forall c_A.\left(\begin{array}{l} c_A \in Components_A \setminus Components_R \\ \Rightarrow Subst(c_A) \neq Container_R(i) \end{array} \right)\right) \quad (SC.17)$$

$$\forall i.\left(i \in \begin{array}{l} ProvInterfaces_R \\ \setminus ProvInterfaces_A \end{array} \Rightarrow Container_R(i) \in \begin{array}{l} Components_R \\ \setminus Components_A \end{array} \right) \quad (SC.18)$$

$$\begin{array}{l} \forall pi \in ProvInterfaces_A, \\ \forall ri \in ReqInterfaces_A \end{array} .(Binding_A(pi) = ri \Rightarrow Binding_R(pi) = ri) \quad (SC.19)$$

$$\forall i, i' \in Interfaces_A.(Delegate_A(i) = i' \Rightarrow Delegate_R(i) = i') \quad (SC.20)$$

$$\begin{array}{l} \forall pi \in ProvInterfaces_R, \forall ri \in ReqInterfaces_R \\ .\left(\left(\begin{array}{l} Binding_R(pi) = ri \\ \wedge Binding_A(pi) \neq ri \end{array} \right) \Rightarrow \left(\begin{array}{l} pi \in ProvInterfaces_R \setminus ProvInterfaces_A \\ \wedge ri \in ReqInterfaces_R \setminus ReqInterfaces_A \end{array} \right)\right) \end{array} \quad (SC.21)$$

$$\begin{array}{l} \forall i, i' \in Interfaces_R \\ .\left(\begin{array}{l} Delegate_R(i) = i' \\ \wedge Delegate_A(i) \neq i' \end{array} \Rightarrow \left(\begin{array}{l} i \in Interfaces_R \setminus Interfaces_A \\ \wedge \left(\begin{array}{l} i' \in Interfaces_R \setminus Interfaces_A \\ \vee \\ i' \in Interfaces_A \wedge \\ \quad \exists c_A \in Components_A \setminus Components_R . \\ \quad (Subst(c_A) = Container_R(i')) \end{array} \right) \end{array} \right)\right) \end{array} \quad (SC.22)$$

- Contraintes de substitution SC

A. Lanoix

Simulation entre séquences de reconfigurations

$$c_R \sqsubseteq c_A \quad \text{(i)}$$

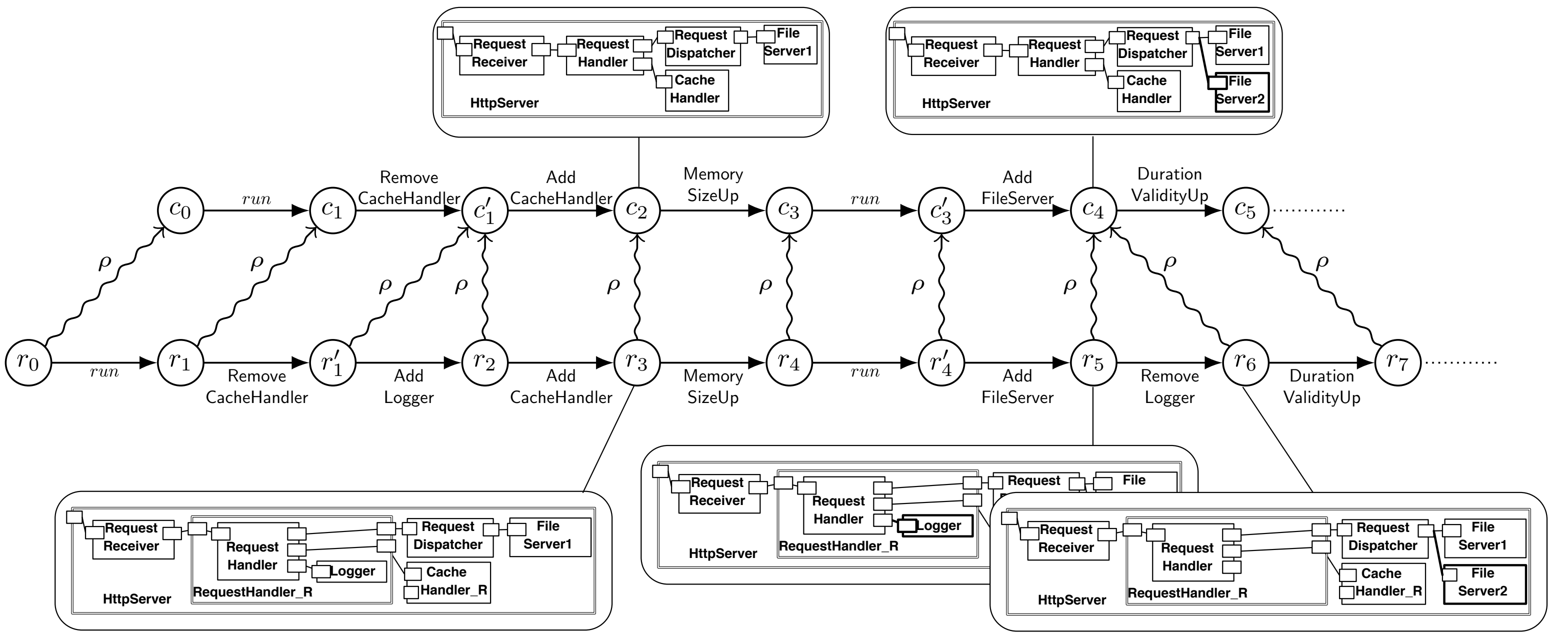
$$\forall c'_R \in \mathcal{C}_R.(c_R \stackrel{ope}{\rightarrow} c'_R \;\Rightarrow\; \exists c'_A \in \mathcal{C}_A.(c_A \stackrel{ope}{\rightarrow} c'_A \wedge c'_R \;\rho\; c'_A)) \quad \text{(ii)}$$

$$\forall c'_R \in \mathcal{C}_R.(c_R \stackrel{ope'}{\rightarrow} c'_R \;\Rightarrow\; c'_R \;\rho\; c_A) \quad \text{(iii)}$$

$$\forall c'_R \in \mathcal{C}_R, \forall k.(k \geq 0 \wedge (c_R =)\sigma_R(k) \stackrel{ope'}{\rightarrow} \sigma_R(k+1) = c'_R \;\Rightarrow\; \exists k'.(k' > k \wedge \sigma_R(k'-1) \stackrel{ope}{\rightarrow} \sigma_R(k'))) \quad \text{(iv)}$$

$$\forall c_A \in \mathcal{C}_A, \forall c_R \in \mathcal{C}_R.(c_R \;\rho\; c_A \wedge c_R \not\rightarrow \;\Rightarrow\; c_A \not\rightarrow) \quad \text{(v)}$$

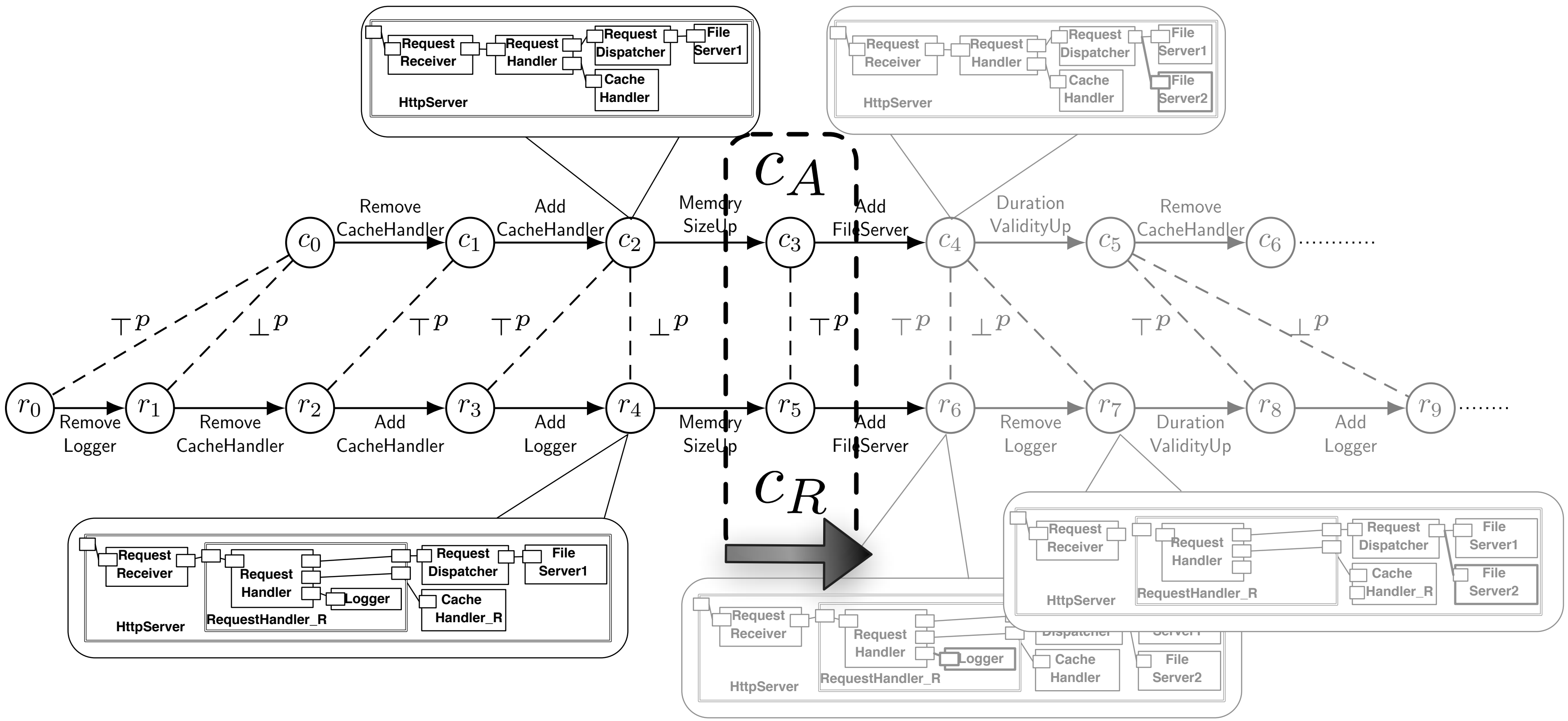
(i) Raffinement structurel
(ii) Simulation stricte
(iii) Tau-simulation
(iv) Non divergence
(v) Non nouveau blocage



En pratique, ce n'est pas vérifiable

A. Lanoix

Evaluation à l'exécution

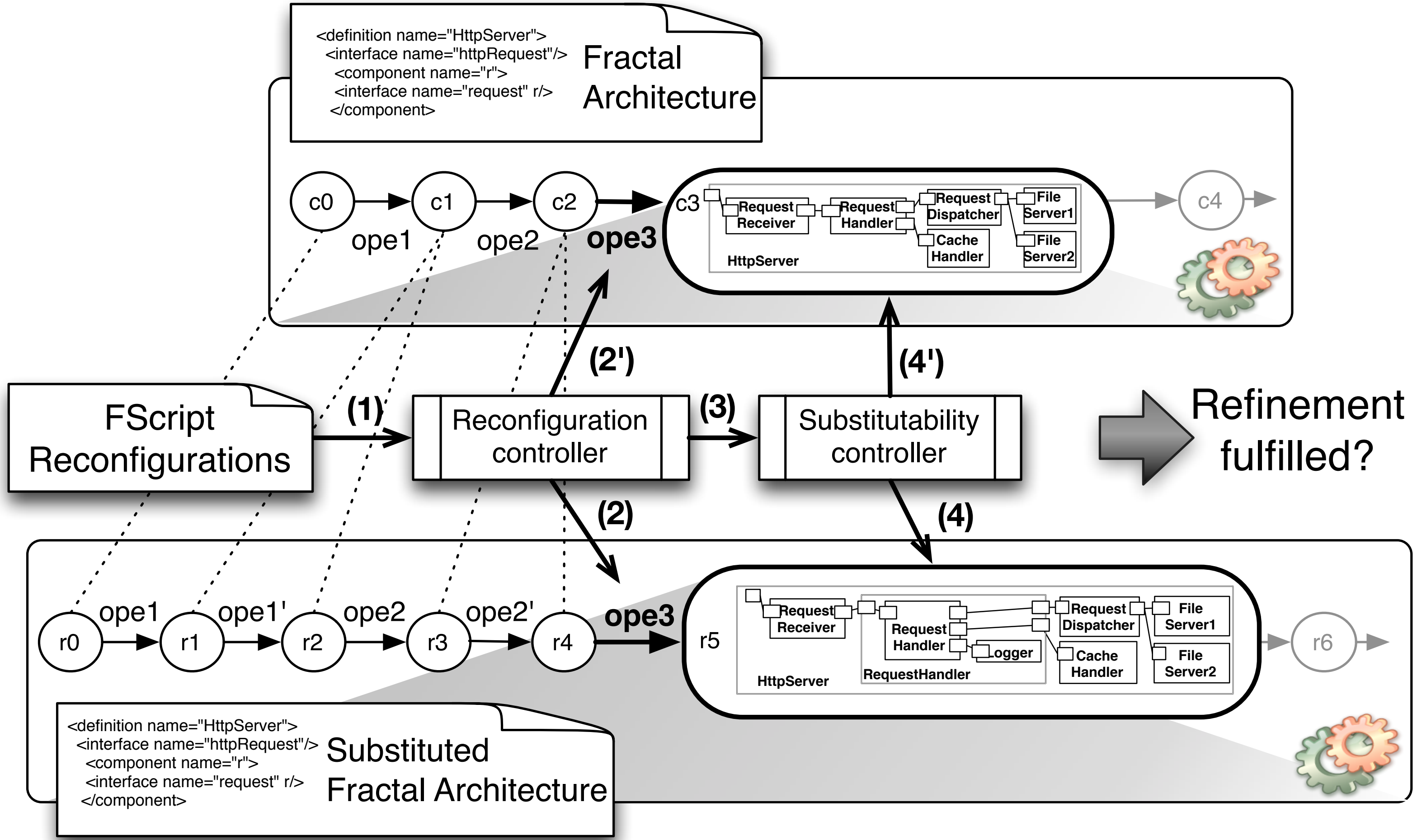


Algorithme de vérification (à la volée)

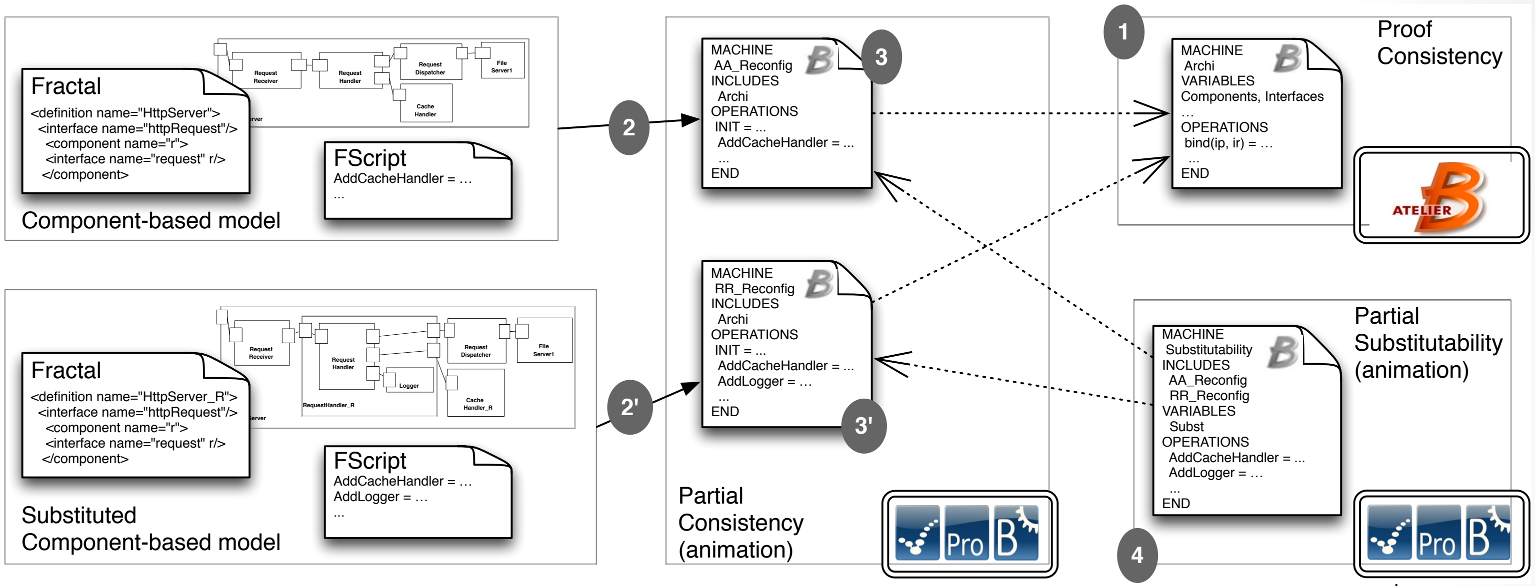
Algorithm 1: Path substitutability

```
Data: c_R^0 ∈ C_R^0, c_A^0 ∈ C_A^0, R_R and R_A
Result: res ∈ {⊥, ⊤}, if terminates
c_R ← c_R^0 ;
c_A ← c_A^0 ;
while ⊤ do
    if substitutable(c_R, c_A) then
        E_R ← enabled(c_R, R_R) ;
        E_A ← enabled(c_A, R_A) ;
        if E_R = ∅ then
            if E_A = ∅ then res ← ⊤ ; break ;
            else res ← ⊥ ; break ;
            end if
        else
            ope ← pick-up(E_R) ;
            c_R ← apply(ope, c_R) ;
            if ope ∈ R_R \ R_A then  print(⊥^p) ;
            else if ope ∈ R_R ∩ R_A and ope ∈ E_A then
                c_A ← apply(ope, c_A) ;
                print(⊤^p)
            else res ← ⊥ ; break;
            end if
        end if
    else res ← ⊥ ; break;
    end if
end while
```

Expérimentation



Validation par animation




```
MACHINE
  Substitutability
INCLUDES
  AA_Reconfig
  RR_Reconfig
VARIABLES
  Subst
INVARIANT
 Subst ∈ AA.Components ⤔ RR.Components
 ∧   AA.Parameters ⊆ RR.Parameters
 ∧ ∀ (pp) . (pp ∈ AA.Parameters ⇒ AA.Value(pp) = RR.Value(pp))
 ∧ ∀ (cc) . (cc ∈ AA.InstanciatedComponents ∩ RR.InstanciatedComponents ⇒ AA.State(cc) = RR.State(cc))
 ...
INITIALISATION
 Subst := {CacheHandler↦CacheHandlerR, RequestHandler↦RequestHandlerR}
OPERATIONS
 AddCacheHandler =
 BEGIN
  AA_AddCacheHandler || RR_AddCacheHandler
 END;
 AddLogger =
 BEGIN
  RR_AddLogger
 END ;
  ...
END
```

Vérification partielle par animation/model-checking

A. Lanoix


That's all Folks!