Component Substitution through Dynamic Reconfigurations

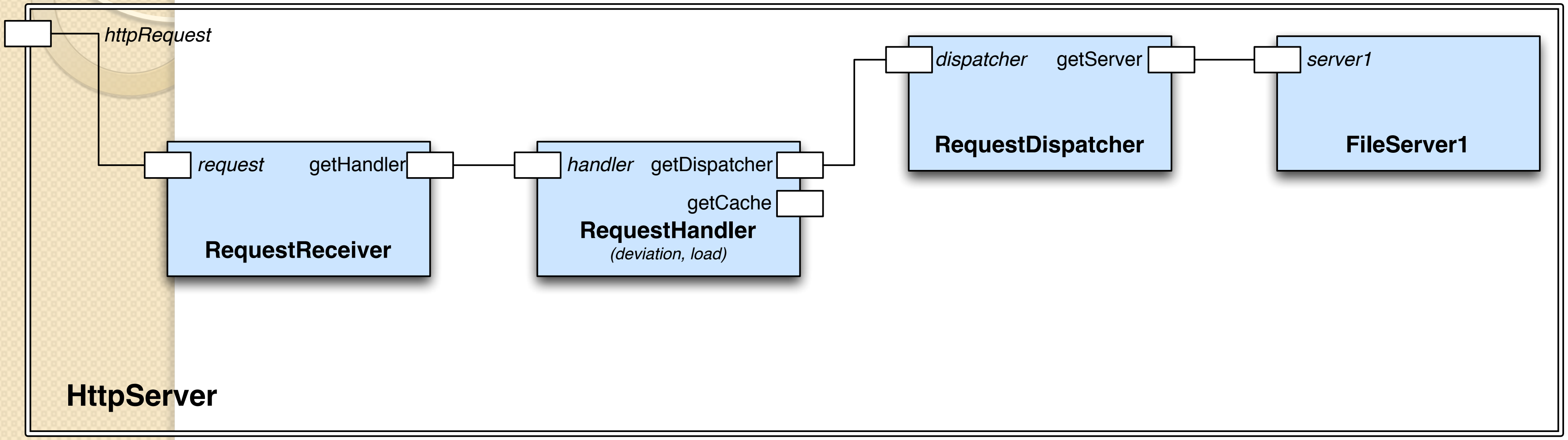
FESCA @ ETAPS 2014 Grenoble - April 12th, 2014

Olga Kouchnarenko
Femto-ST, University of Franche-Comté, France
olga.kouchnarenko@univ-fcomte.fr

Arnaud Lanoix
LINA, Nantes University, France
arnaud.lanoix@univ-nantes.fr

Outline

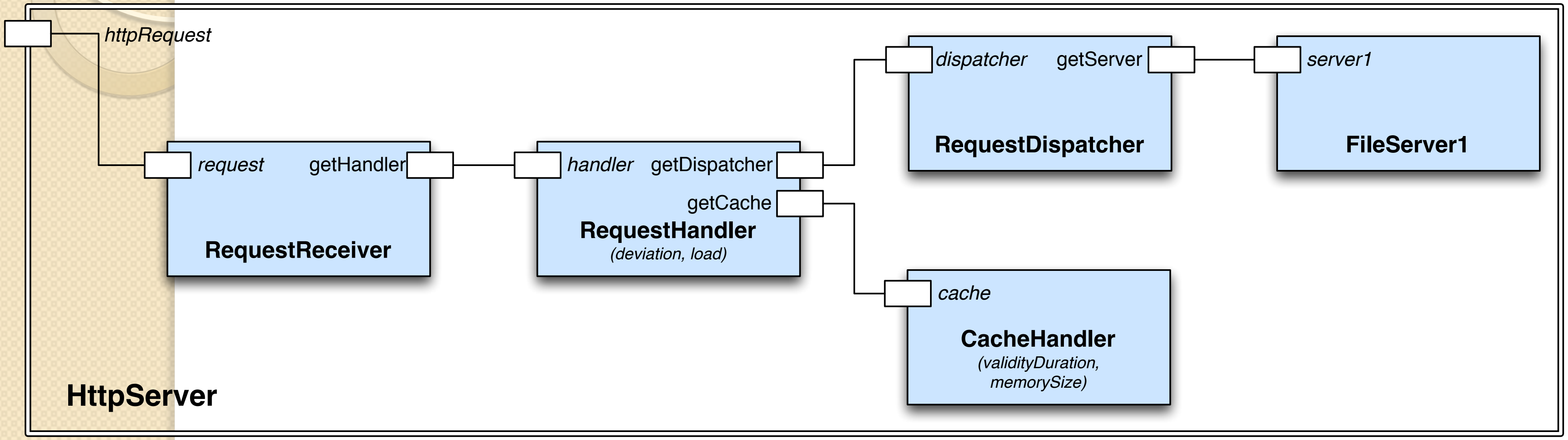
Running example: HTTP Server



Dynamic reconfigurations:

- `AddCacheHandler / RemoveCacheHandler`
- `AddFileServer / RemoveFileServer`

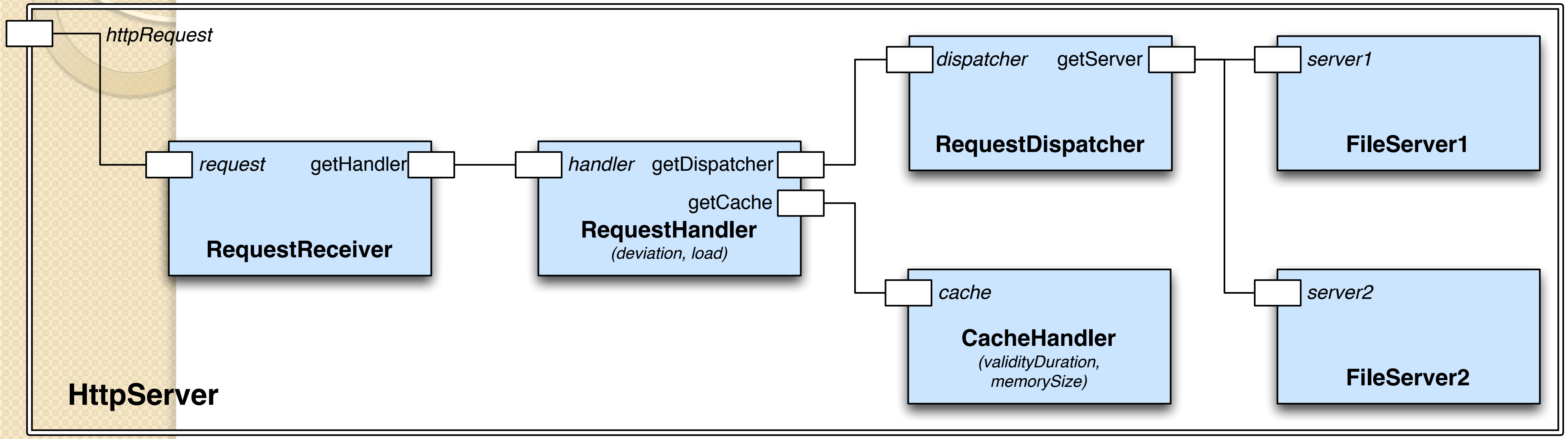
Running example: HTTP Server



Dynamic reconfigurations:

- `AddCacheHandler / RemoveCacheHandler`
- `AddFileServer / RemoveFileServer`

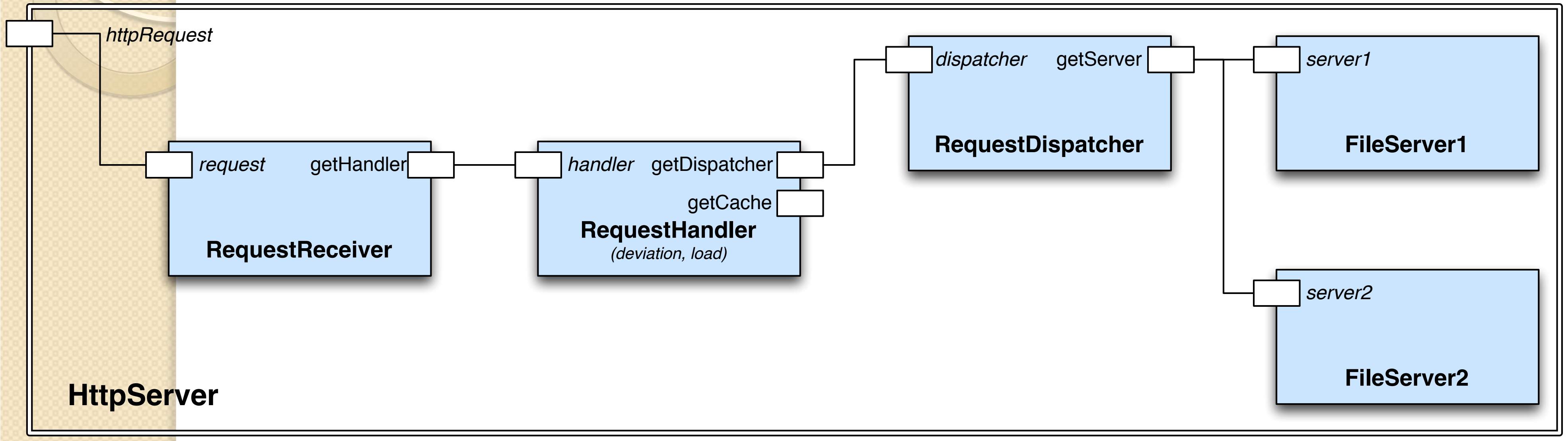
Running example: HTTP Server



Dynamic reconfigurations:

- `AddCacheHandler / RemoveCacheHandler`
- `AddFileServer / RemoveFileServer`

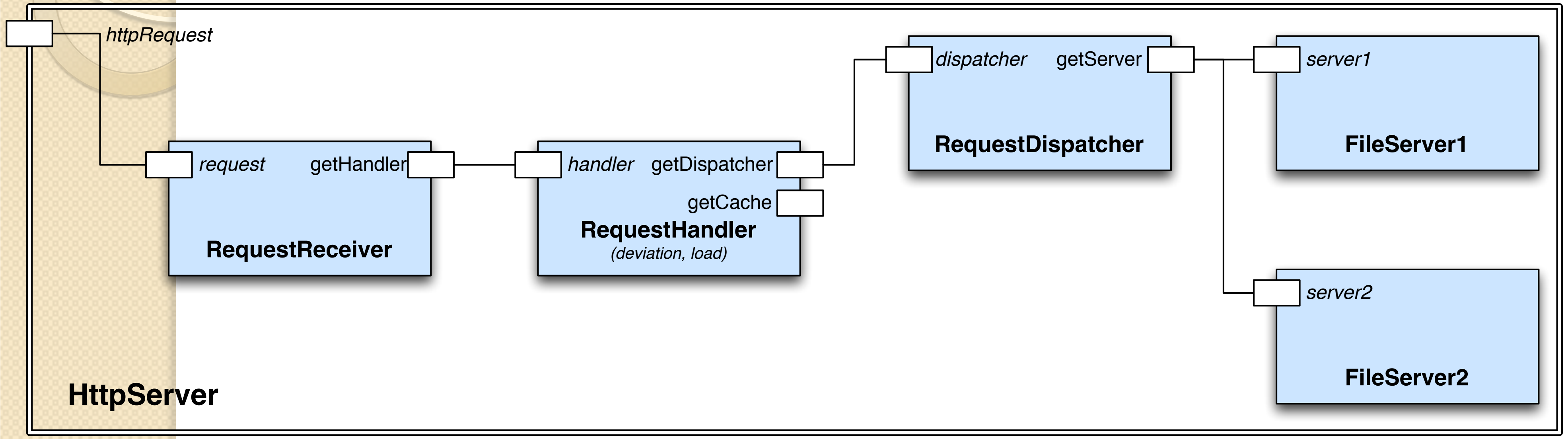
Running example: HTTP Server



Dynamic reconfigurations:

- `AddCacheHandler / RemoveCacheHandler`
- `AddFileServer / RemoveFileServer`

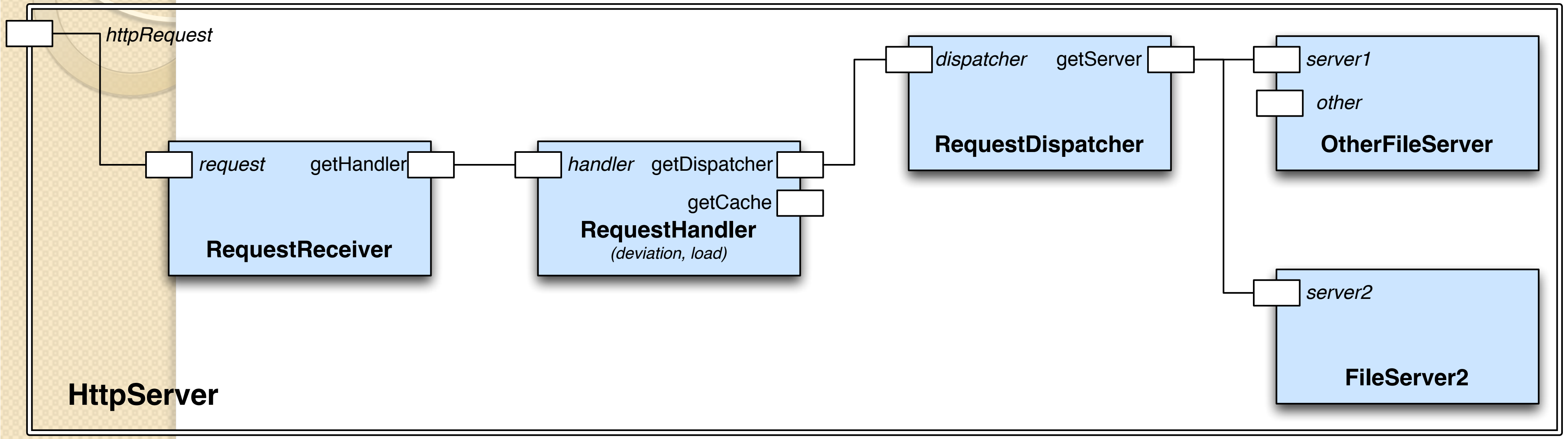
Running example: HTTP Server



Reconfiguration by substitution:

- Replace **RequestHandler** by **RequestHandler_R**
- Replace **FileServer1** by **AnotherFileServer**

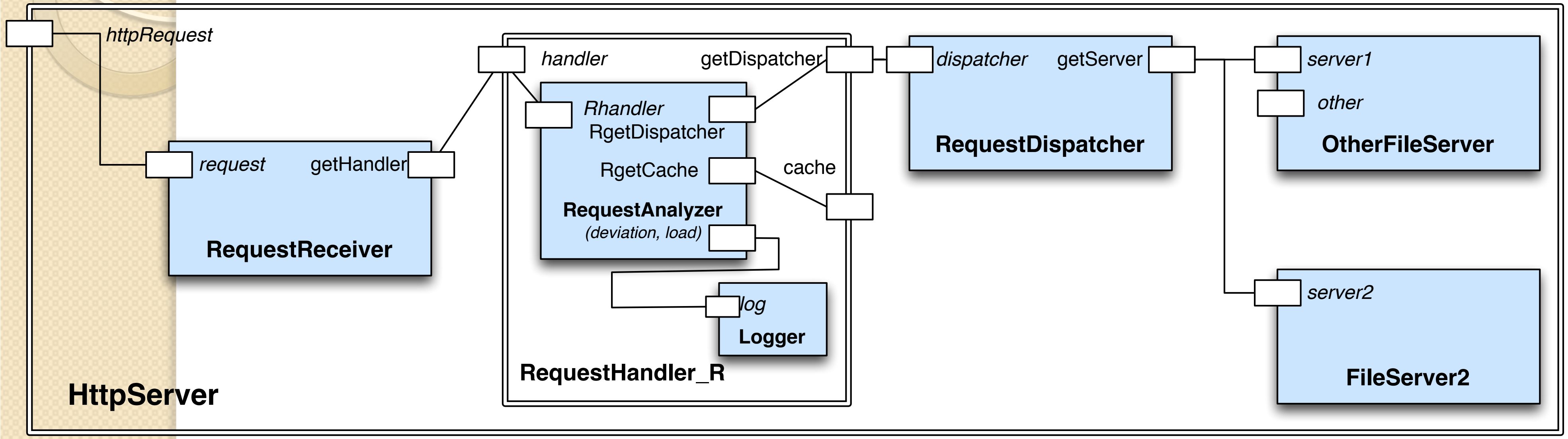
Running example : HTTP Server



Reconfiguration by substitution:

- Replace **RequestHandler** by **RequestHandler_R**
- Replace **FileServer1** by **AnotherFileServer**

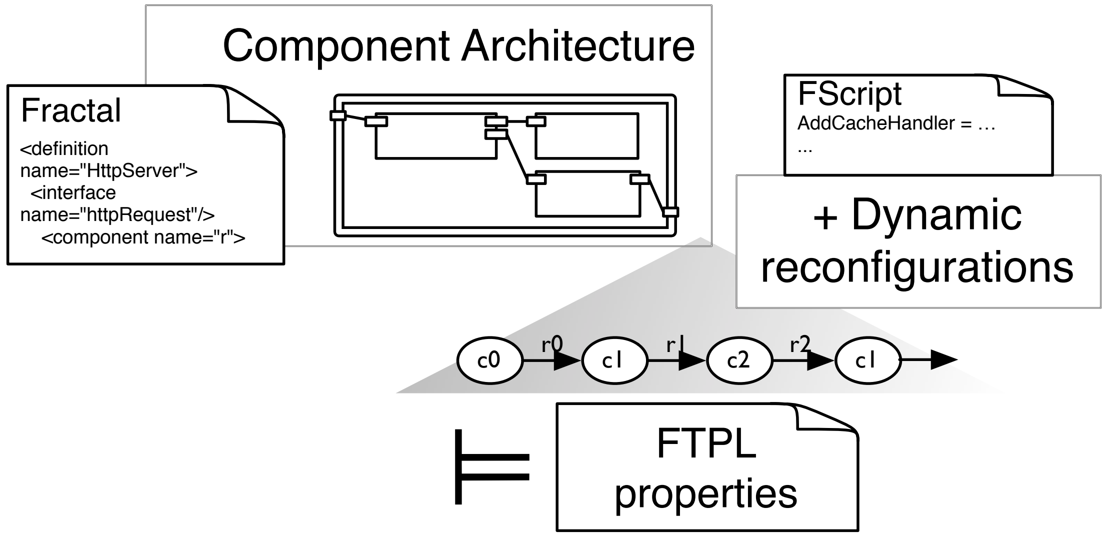
Running example: HTTP Server



Reconfiguration by substitution:

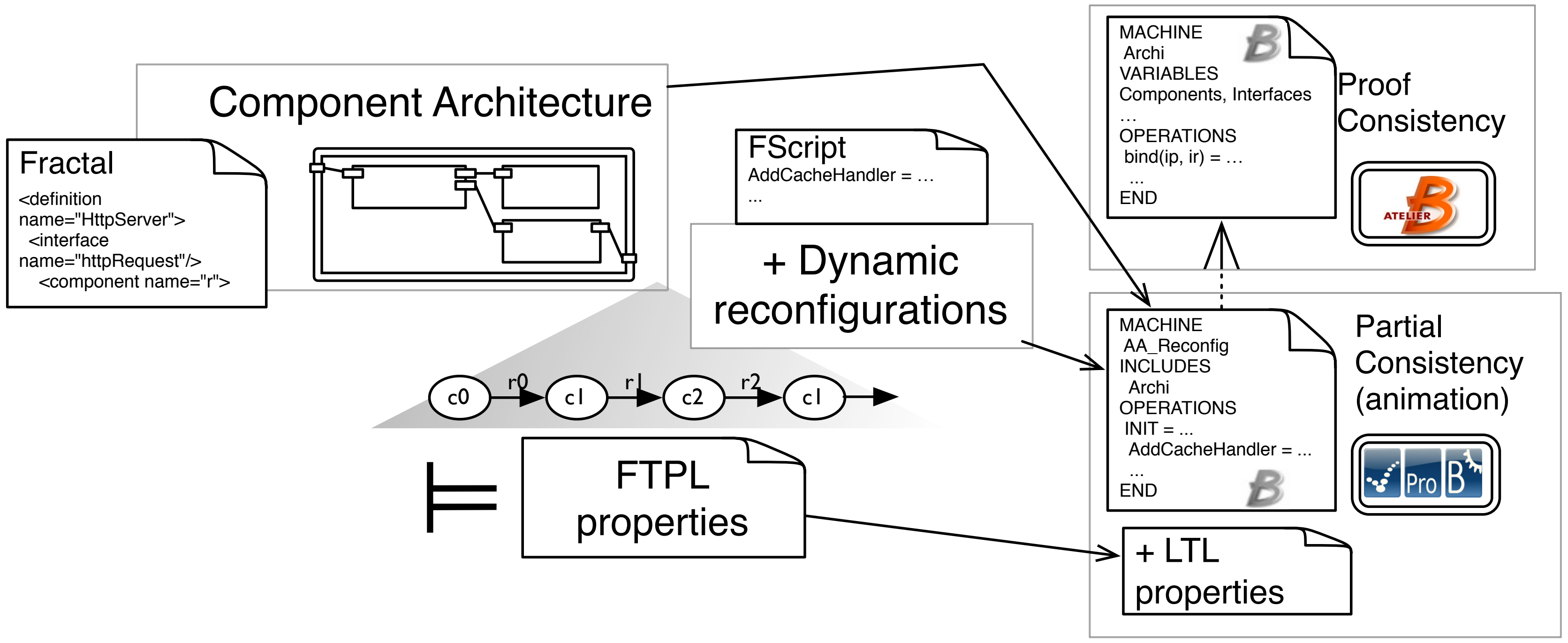
- Replace **RequestHandler** by **RequestHandler_R**
- Replace **FileServer1** by **AnotherFileServer**

Previous works



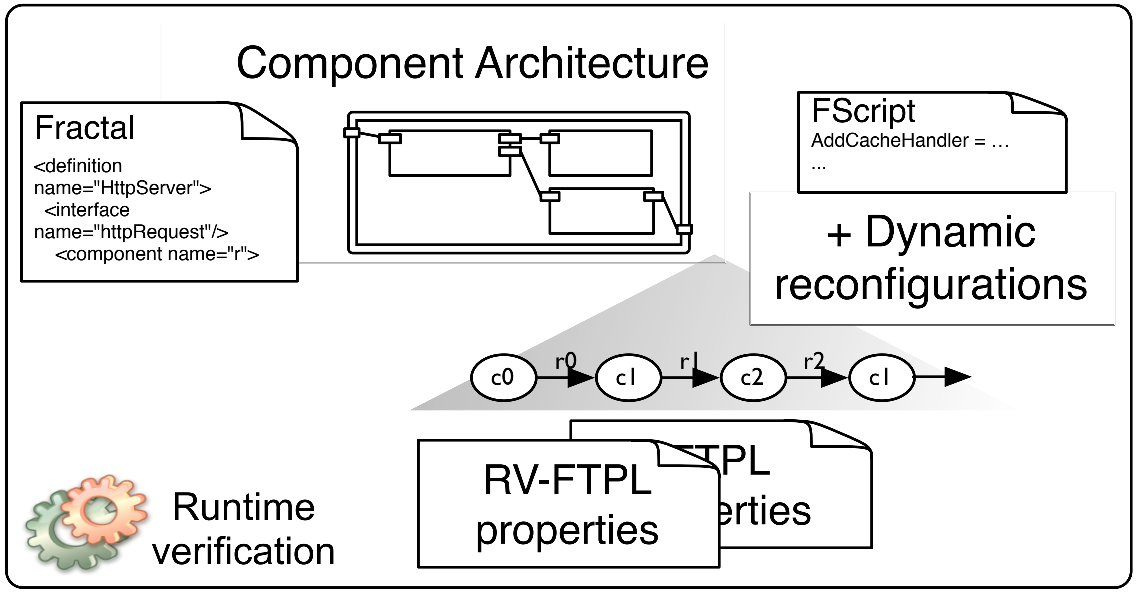
J. Dormoy, O. Kouchnarenko & A. Lanoix
Using Temporal Logic for Dynamic Reconfigurations of Components.
FACS 2010

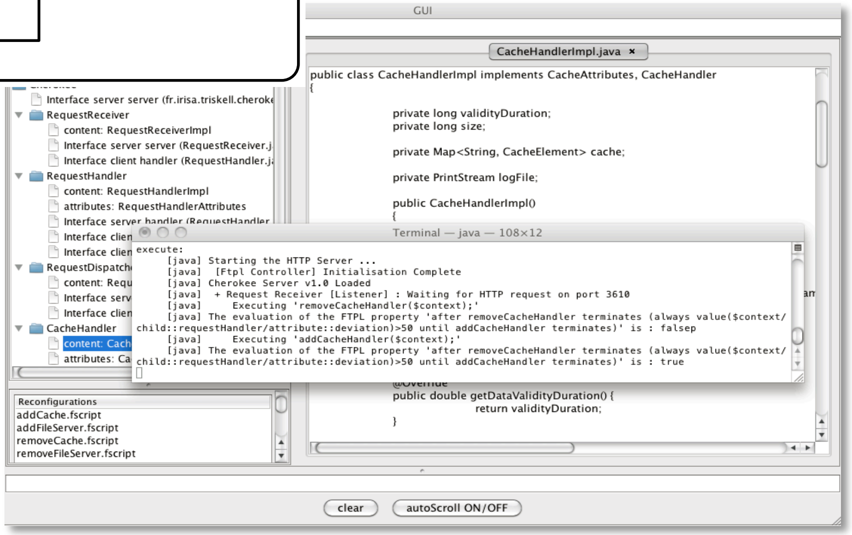
Previous works



A.Lanoix, J.Dormoy & O.Kouchnarenko
Combining Proof and Model-checking to Validate Reconfigurable Architectures.
FESCA 2011

Previous works





```
Interface server server (fr.irisa.triskell.cheroke
RequestReceiver
    content: RequestReceiverImpl
    Interface server server (RequestReceiver.j
    Interface client handler (RequestHandler.j
RequestHandler
    content: RequestHandlerImpl
    attributes: RequestHandlerAttributes
    Interface server handler (RequestHandler
    Interface clien
    Interface clien
RequestDispatch
    content: Requ
    Interface serv
    Interface clien
CacheHandler
    content: Cach
    attributes: Ca
```

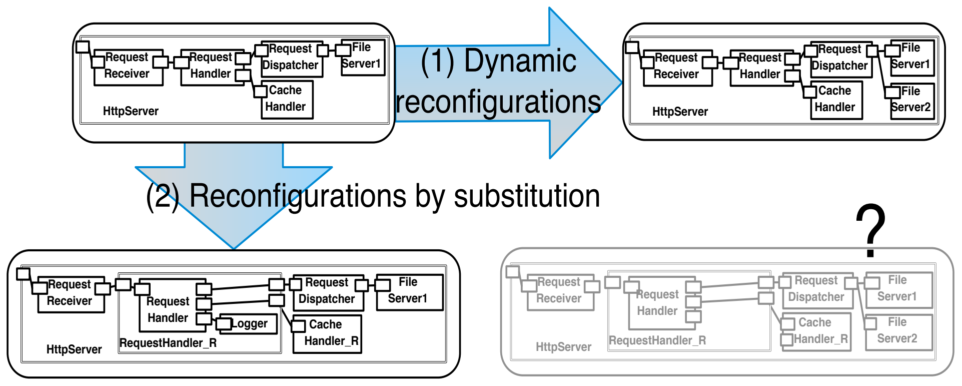
```
addCache.fscript
addFileServer.fscript
removeCache.fscript
removeFileServer.fscript
```

```
public class CacheHandlerImpl implements CacheAttributes, CacheHandler
{

        private long validityDuration;
        private long size;

        private Map<String, CacheElement> cache;

        private PrintStream logFile;

        public CacheHandlerImpl()
        {

        @Override
        public double getDataValidityDuration() {
                return validityDuration;
        }
```

```
Terminal — java — 108×12
execute:
     [java] Starting the HTTP Server ...
     [java]  [Ftpl Controller] Initialisation Complete
     [java] Cherokee Server v1.0 Loaded
     [java]  + Request Receiver [Listener] : Waiting for HTTP request on port 3610
     [java]     Executing 'removeCacheHandler($context);'
     [java] The evaluation of the FTPL property 'after removeCacheHandler terminates (always value($context/
child::requestHandler/attribute::deviation)>50 until addCacheHandler terminates)' is : falsep
     [java]     Executing 'addCacheHandler($context);'
     [java] The evaluation of the FTPL property 'after removeCacheHandler terminates (always value($context/
child::requestHandler/attribute::deviation)>50 until addCacheHandler terminates)' is : true
```

J. Dormoy, O. Kouchnarenko & A. Lanoix
Runtime Verification of Temporal Patterns for Dynamic Reconfigurations of Components.
FACS 2011

Motivations

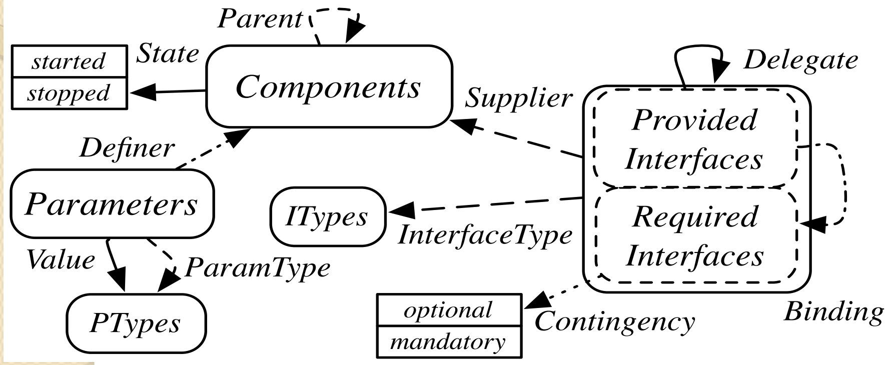


- *Needs*
  - Validate component architectures evolution through reconfigurations
  - Combine different kinds of reconfigurations
    - **Dynamic reconfigurations** (1)
    - **Reconfigurations by substitution** (2)
- *Proposals*
  - Define **Substitutability Constraints**
  - Integrate them into a **substitutability-based simulation**
    - Propose a semi-algorithm to check it on the fly

Outline

Architectural reconfiguration model

- **Graph-based representation** of the component architecture



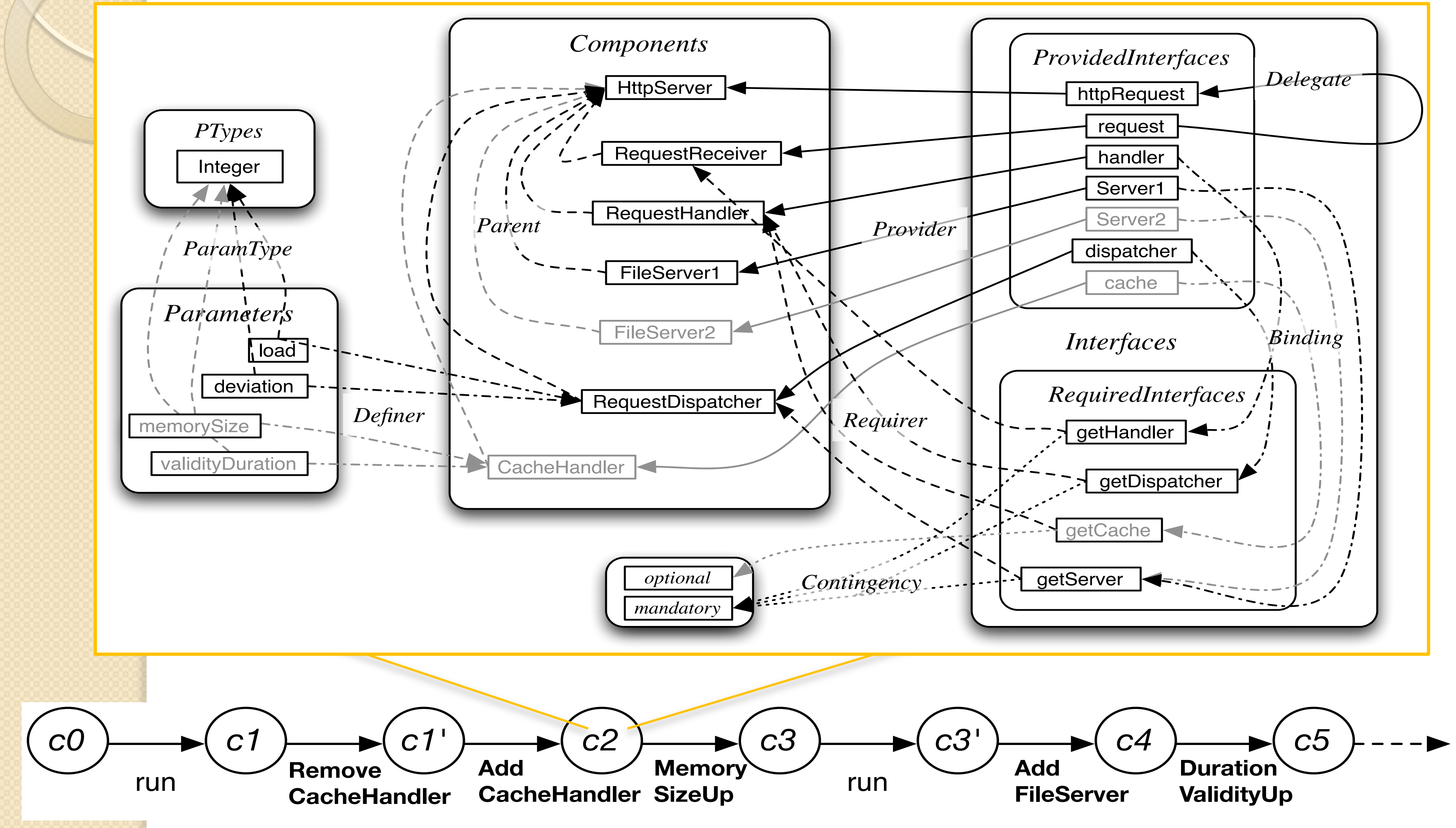
Configurations = architectural elements + relations

- **Consistent configurations**
  - Configurations respect the Consistency Constraints CC

- **Dynamic reconfigurations** = graph transformations
  - add/delete components
  - bind/unbind interfaces
  - change value of parameters

combination of them

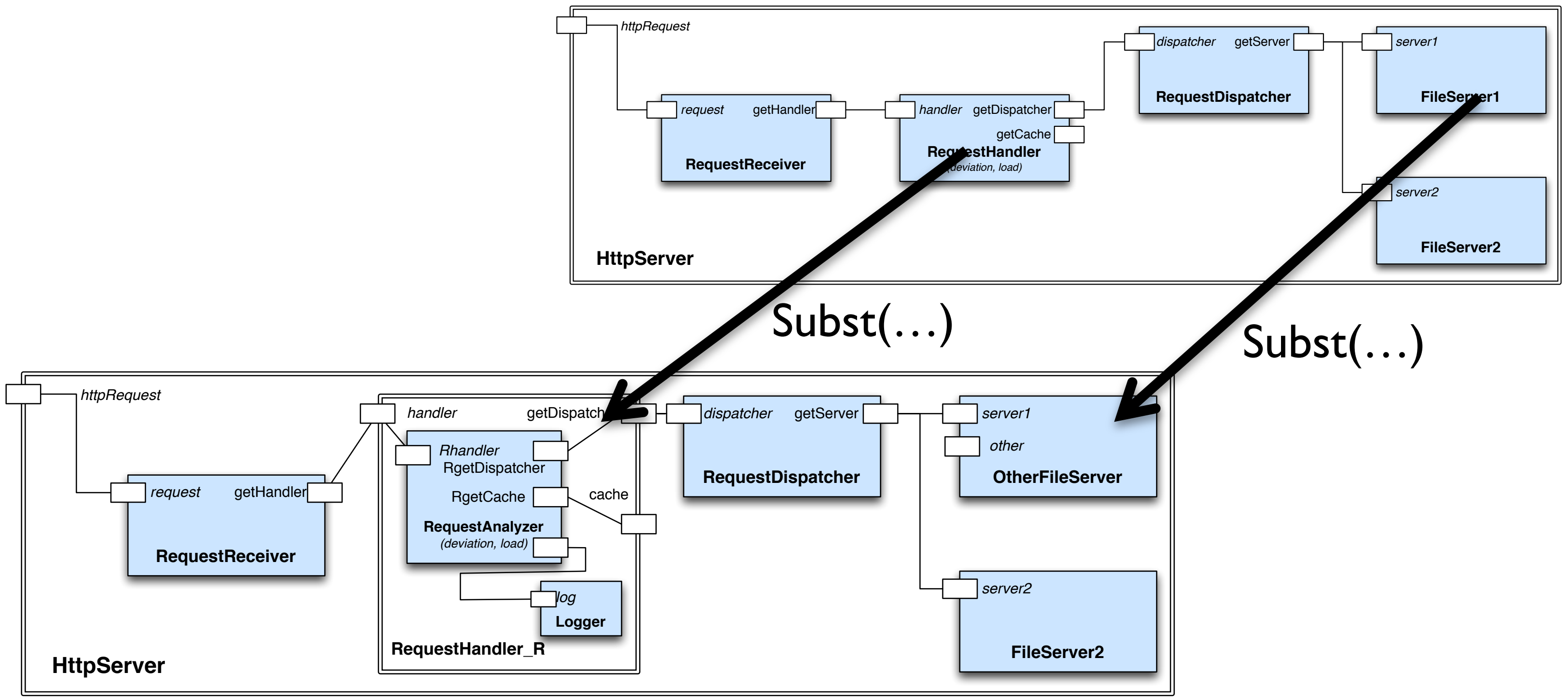
Architectural reconfiguration model



Outline

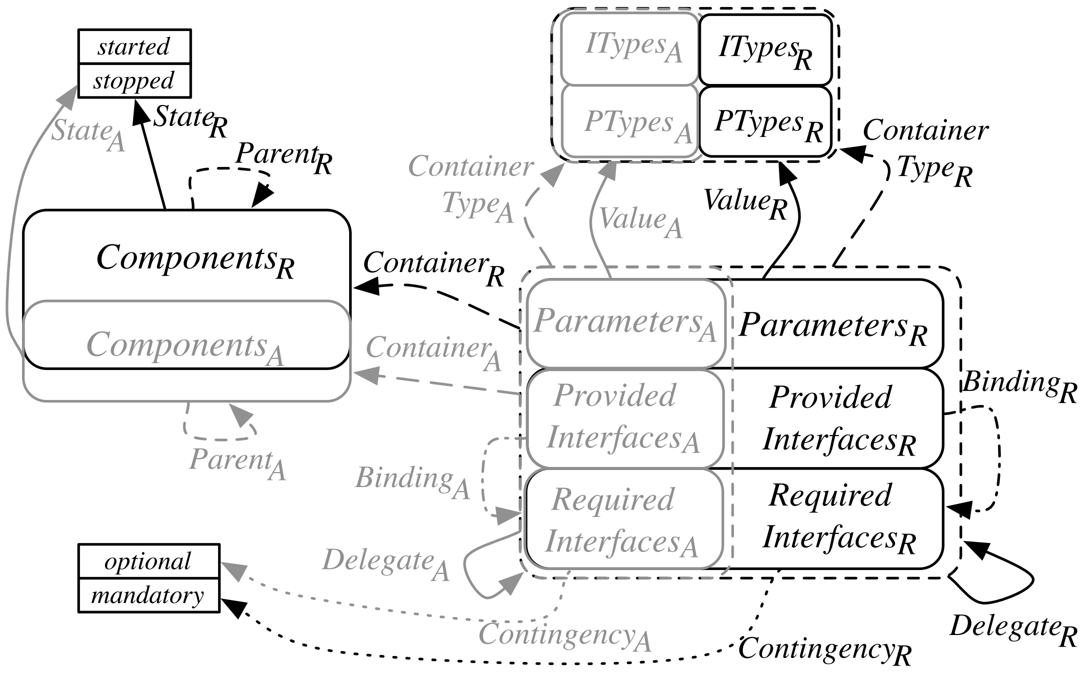
Reconfigurations by component substitution

- Two kinds of component substitutions:
  - **new version** of the component
  - **composite component** encapsulates new sub-components



Reconfigurations by component substitution

- Component **structural substitution** vs. Component encapsulation
  - Substituted components supply the **same interfaces** as before



➔ (Structural) **substitutability constraints**

Some substitutability constraints

- *"the old components remain unchanged"* [5/21]

$$\boxed{\begin{matrix}\forall c \in Components_A \cap Components_R, \\ \forall x \in Interfaces_A \cup Parameters_A\end{matrix} . \begin{pmatrix} Container_A(x) = c \\ \Rightarrow Container_R(x) = c \end{pmatrix}}$$

- *"an old component completely disapears only if it is substituted by a new version of itself"* [7/21]

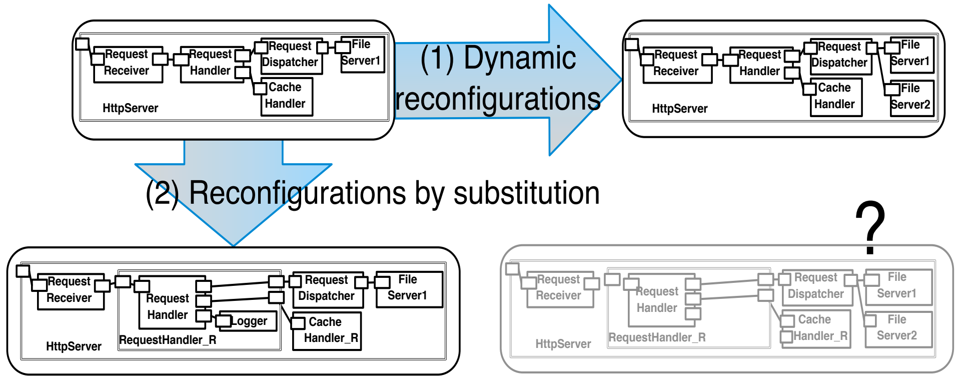
$$\boxed{\begin{matrix}\forall c_A \in Components_A \setminus Components_R \Rightarrow \\ (\exists c_R \in Components_R \setminus Components_A .(Subst(c_A) = c_R))\end{matrix}}$$

- *"the newly introduced components must be subcomponents of some substituted components"* [12/21]

$$\boxed{\begin{matrix}\forall c_R \in Components_R \setminus Components_A \\ \forall c_A \in Components_A \setminus Components_R\end{matrix} . \begin{pmatrix} Subst(c_A) \neq c_R \Rightarrow \\ \exists c'_R \in Components_R \setminus Components_A .((c_R, c'_R) \in Parent_R) \end{pmatrix}}$$

Combining component substitution with dynamic reconfigurations

- **Newly** substituted components introduce **new** dynamic reconfigurations

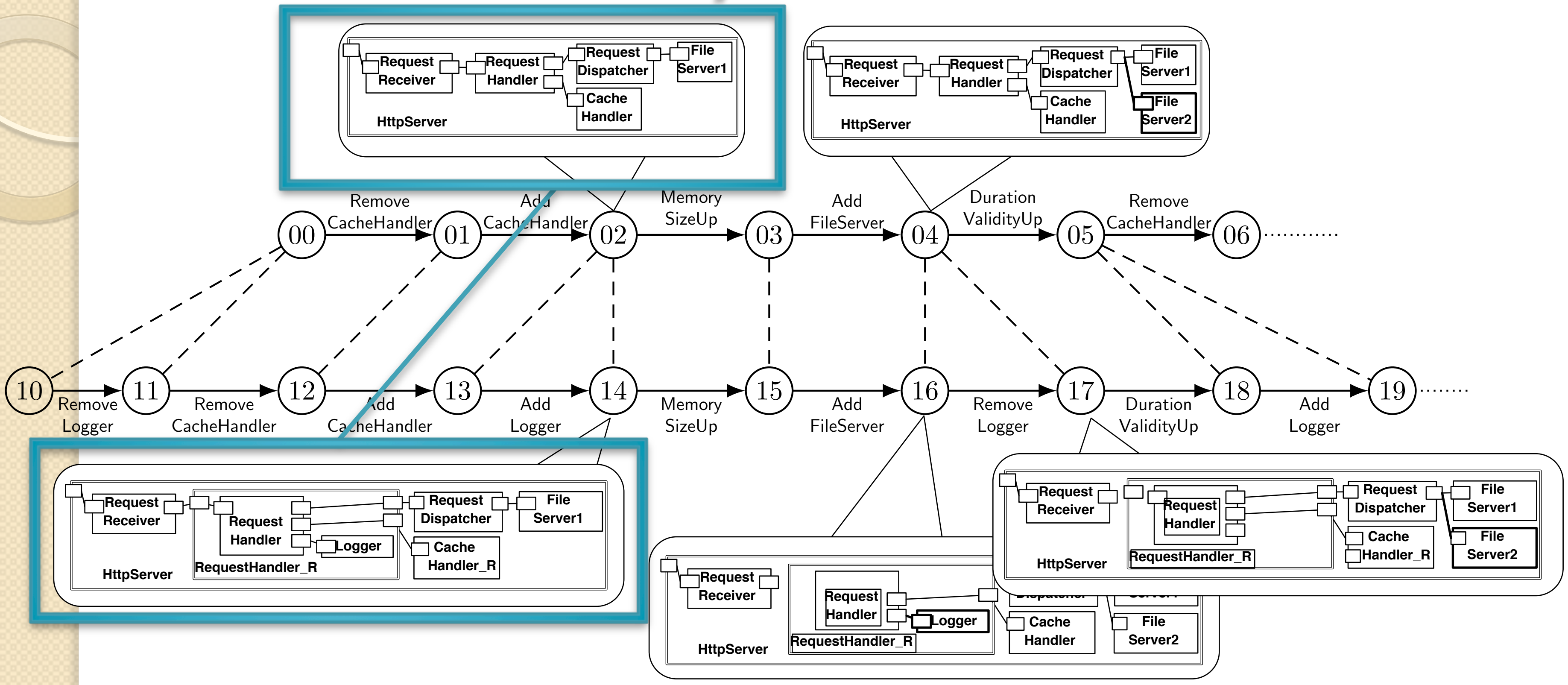


  - How to make new reconfigurations **preserve** the old reconfiguration sequences?

- **➔ Substitutability-based simulation**
  - a kind of weak simulation [Milner-Park]

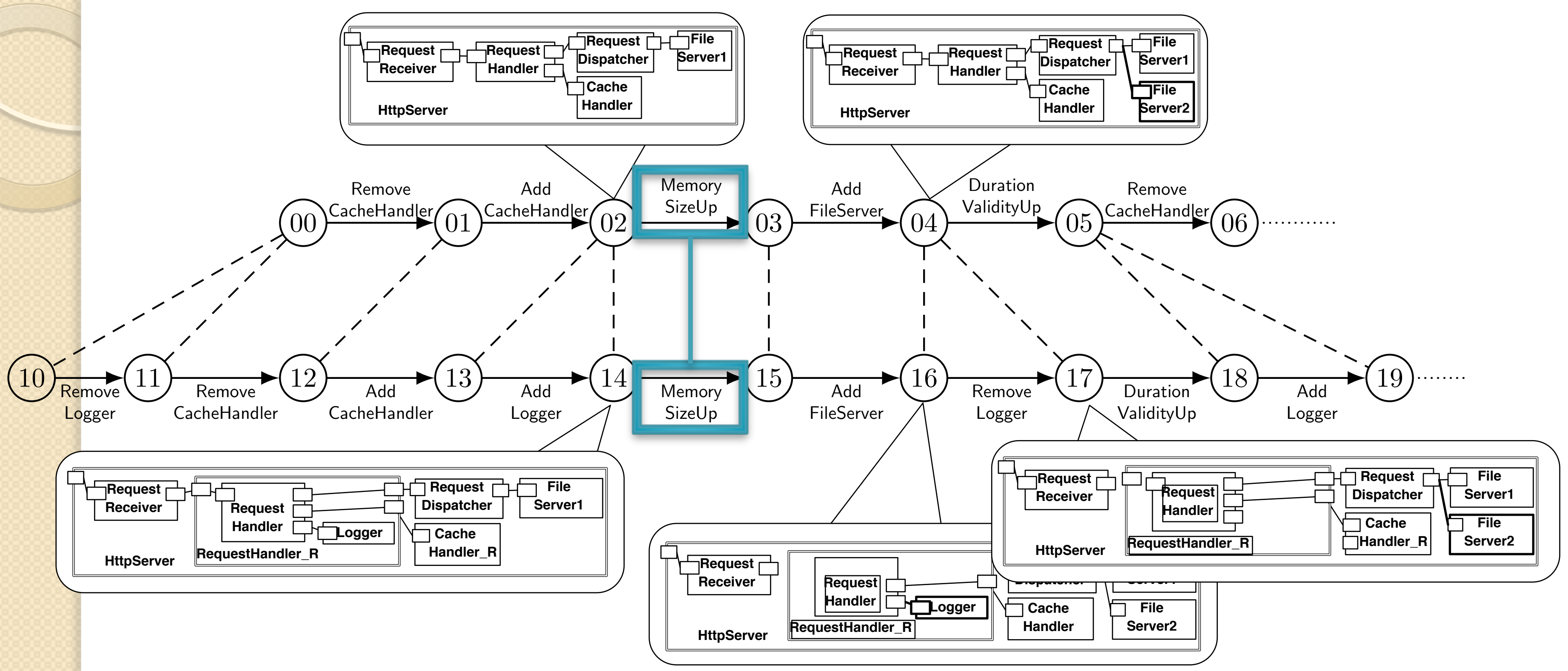
Outline

Substitutability-based simulation



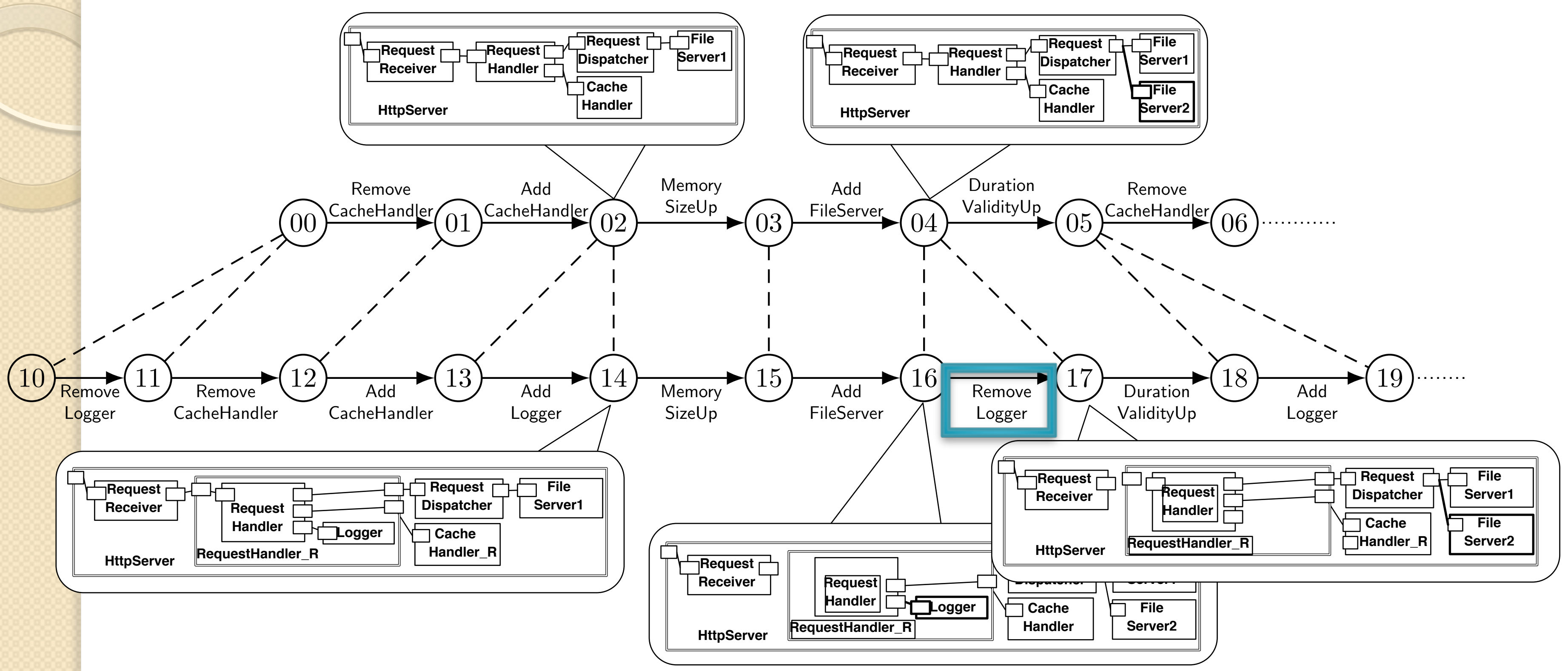
1. **Substitutability constraints**

Substitutability-based simulation



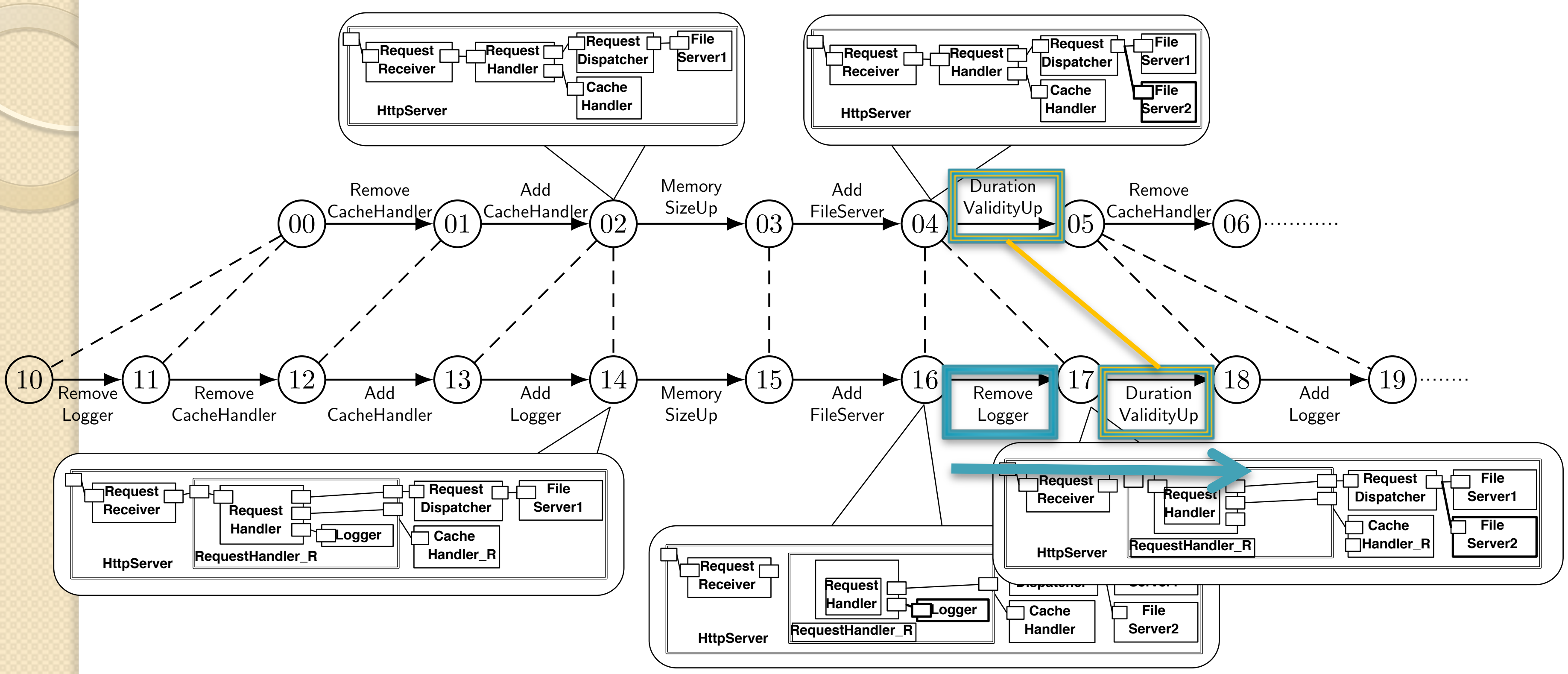
1. Substitutability constraints
2. **Strict reconfiguration simulation**

Substitutability-based simulation



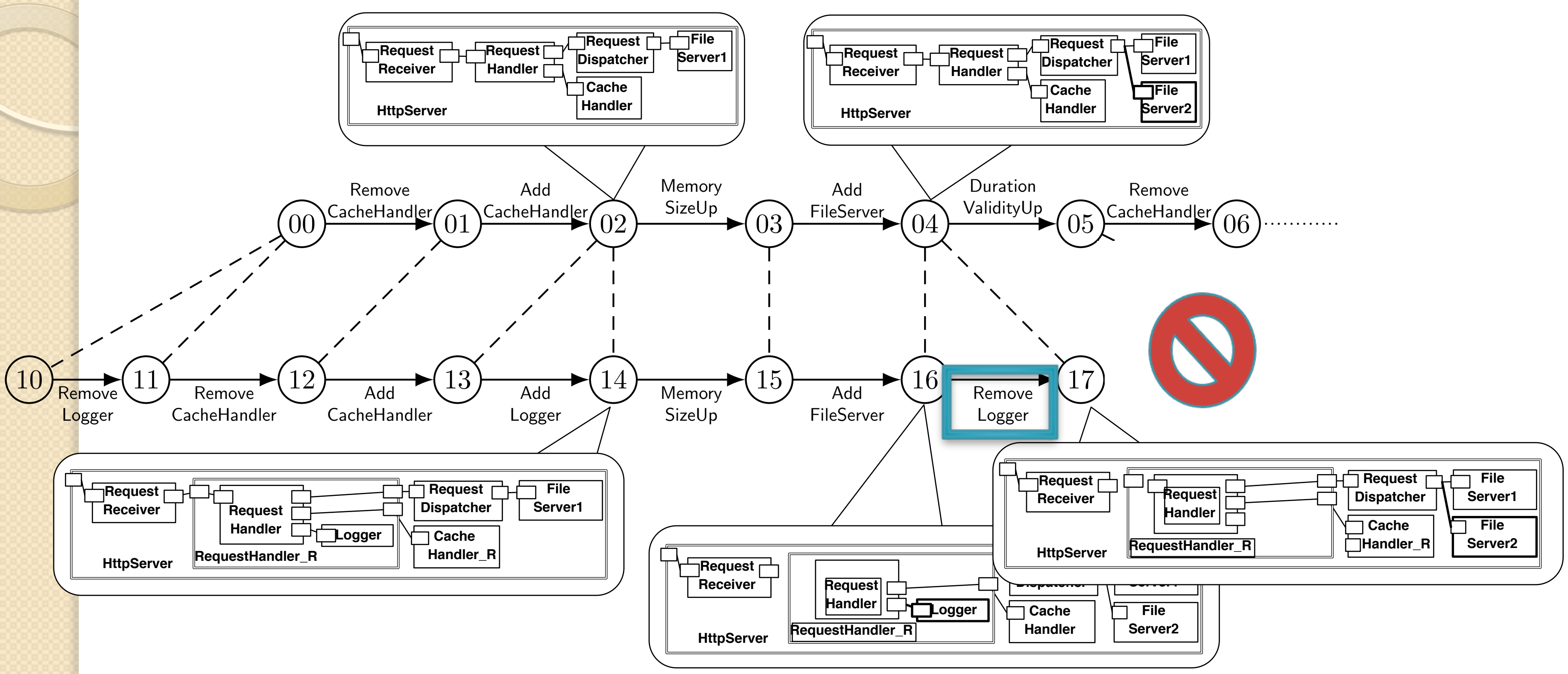
1. Substitutability constraints
2. Strict reconfiguration simulation
3. **Stuttering reconfiguration simulation**

Substitutability-based simulation



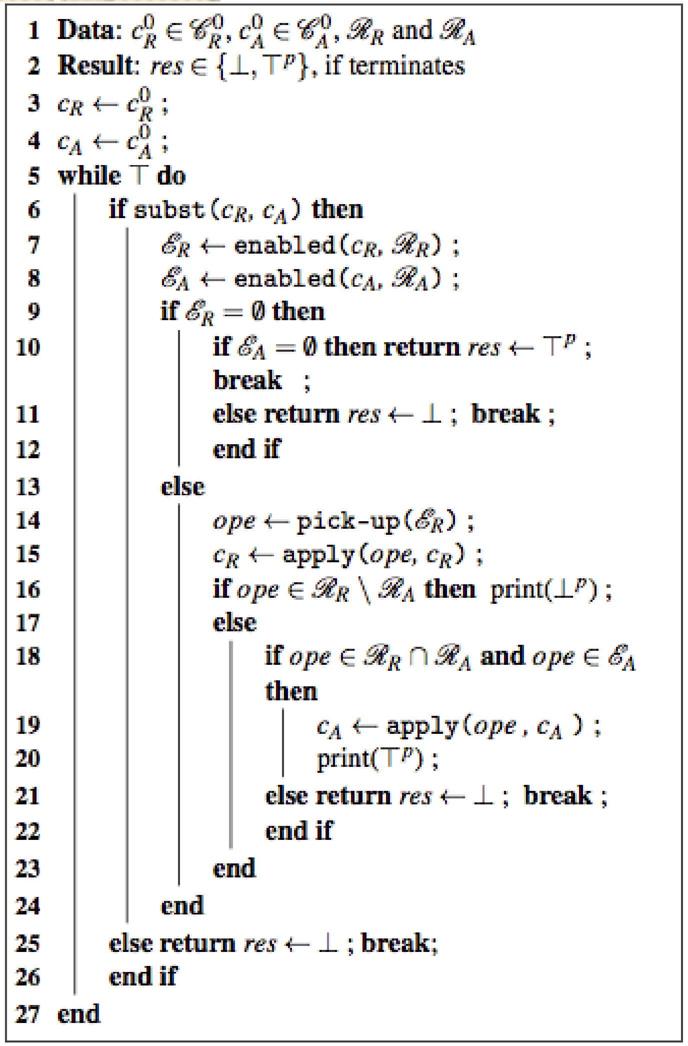
1. Substitutability constraints
2. Strict reconfiguration simulation
3. Stuttering reconfiguration simulation
4. **No cycle of new reconfigurations**

Substitutability-based simulation



1. Substitutability constraints
2. Strict reconfiguration simulation
3. Stuttering reconfiguration simulation
4. No cycle of new reconfigurations
5. **No new deadlocks**

Substitutability-based simulation, at runtime



```
Data: $c^0_R \in \mathscr{C}^0_R$, $c^0_A \in \mathscr{C}^0_A$, $\mathscr{R}_R$ and $\mathscr{R}_A$
Result: $res \in \{\bot, \top^p\}$, if terminates
$c_R \leftarrow c^0_R$ ;
$c_A \leftarrow c^0_A$ ;
while $\top$ do
    if subst($c_R$, $c_A$) then
        $\mathscr{E}_R \leftarrow$ enabled($c_R$, $\mathscr{R}_R$) ;
        $\mathscr{E}_A \leftarrow$ enabled($c_A$, $\mathscr{R}_A$) ;
        if $\mathscr{E}_R = \emptyset$ then
            if $\mathscr{E}_A = \emptyset$ then return $res \leftarrow \top^p$ ;
            break ;
            else return $res \leftarrow \bot$ ; break ;
            end if
        else
            $ope \leftarrow$ pick-up($\mathscr{E}_R$) ;
            $c_R \leftarrow$ apply($ope$, $c_R$) ;
            if $ope \in \mathscr{R}_R \setminus \mathscr{R}_A$ then print($\bot^p$) ;
            else
                if $ope \in \mathscr{R}_R \cap \mathscr{R}_A$ and $ope \in \mathscr{E}_A$
                then
                    $c_A \leftarrow$ apply($ope$, $c_A$) ;
                    print($\top^p$) ;
                else return $res \leftarrow \bot$ ; break ;
                end if
            end
        end
    else return $res \leftarrow \bot$ ; break;
    end if
end
```

- A kind of
  - Weak simulation [Milner-Park]
  - Divergence-sensitive stability simulation [van Glabbeek]
- ➔ **undecidable**, in general
- **Runtime evaluation:**

$\bot$ { **1. substitutability constraints** / **2. strict simulation** / **3. stutterring simulation** } is broken

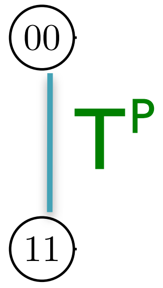
$\bot^P$ when **stuttering clause** is correct

*(new reconfigurations **could** introduce potential new cycles, potential new deadlocks)*

$\top^P$ otherwise

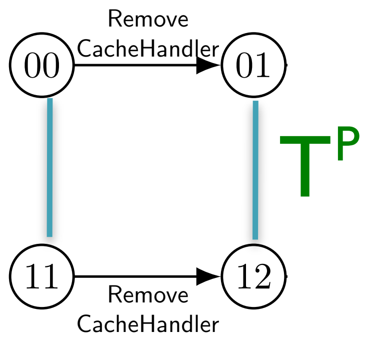
(the evaluation must be continued…)

Principle of the runtime evaluation



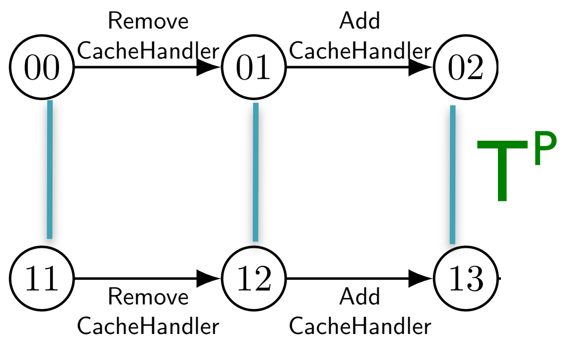
Substitutability constraints?	OK
Strict reconfiguration simulation?	
Stuttering reconfiguration simulation?	
Cycle of new reconfigurations?	
New deadlocks?	

Principle of the runtime evaluation



Substitutability constraints?	OK
Strict reconfiguration simulation?	OK
Stuttering reconfiguration simulation?	
Cycle of new reconfigurations?	No
New deadlocks?	No

Principle of the runtime evaluation



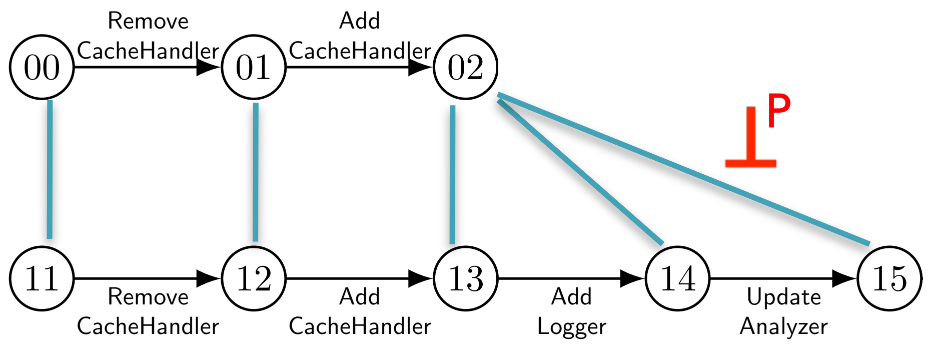
Substitutability constraints?	OK
Strict reconfiguration simulation?	OK
Stuttering reconfiguration simulation?	
Cycle of new reconfigurations?	No
New deadlocks?	No

Principle of the runtime evaluation

$$\begin{tikzcd}
00 \arrow[r, "\text{Remove CacheHandler}"] \arrow[d, dash] & 01 \arrow[r, "\text{Add CacheHandler}"] \arrow[d, dash] & 02 \arrow[d, dash] \arrow[dr, dash, "\perp^{P}"] & \\
11 \arrow[r, "\text{Remove CacheHandler}"'] & 12 \arrow[r, "\text{Add CacheHandler}"'] & 13 \arrow[r, "\text{Add Logger}"'] & 14
\end{tikzcd}$$

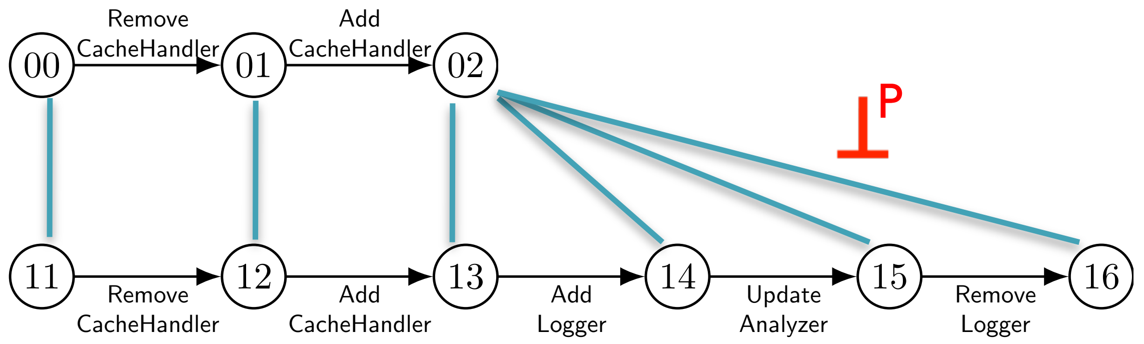
Substitutability constraints?	OK
Strict reconfiguration simulation?	
Stuttering reconfiguration simulation?	OK
Cycle of new reconfigurations?	Potentially
New deadlocks?	Potentially

Principle of the runtime evaluation



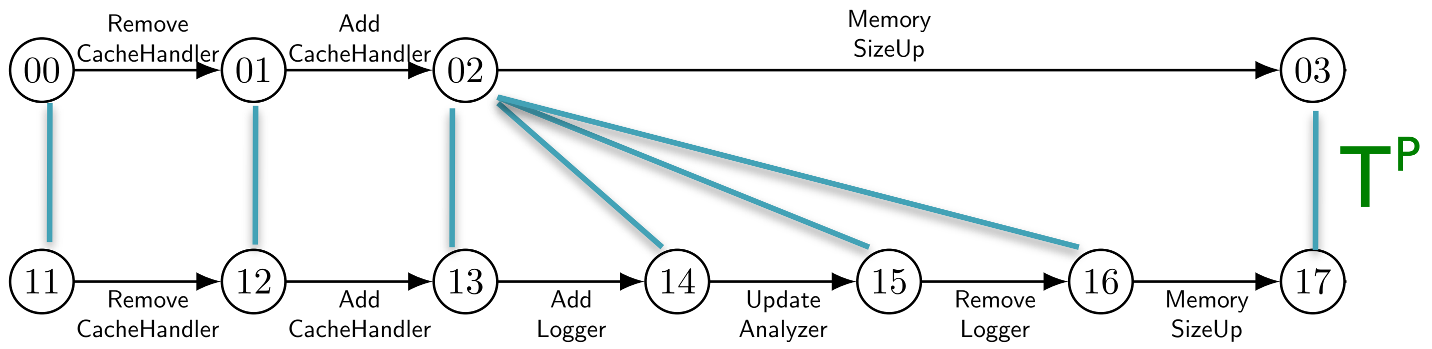
Substitutability constraints?	OK
Strict reconfiguration simulation?	
Stuttering reconfiguration simulation?	OK
Cycle of new reconfigurations?	Potentially
New deadlocks?	Potentially

Principle of the runtime evaluation



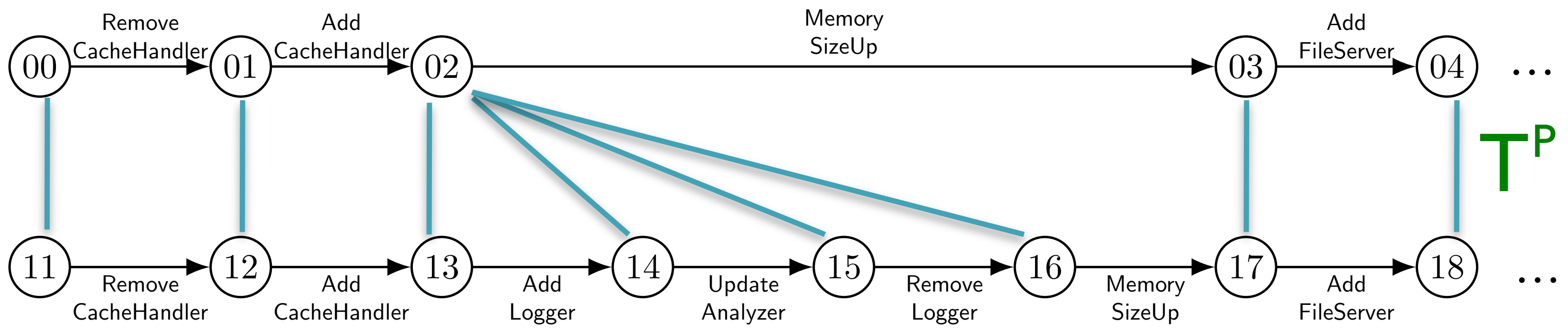
Substitutability constraints?	OK
Strict reconfiguration simulation?	
Stuttering reconfiguration simulation?	OK
Cycle of new reconfigurations?	Potentially
New deadlocks?	Potentially

Principle of the runtime evaluation



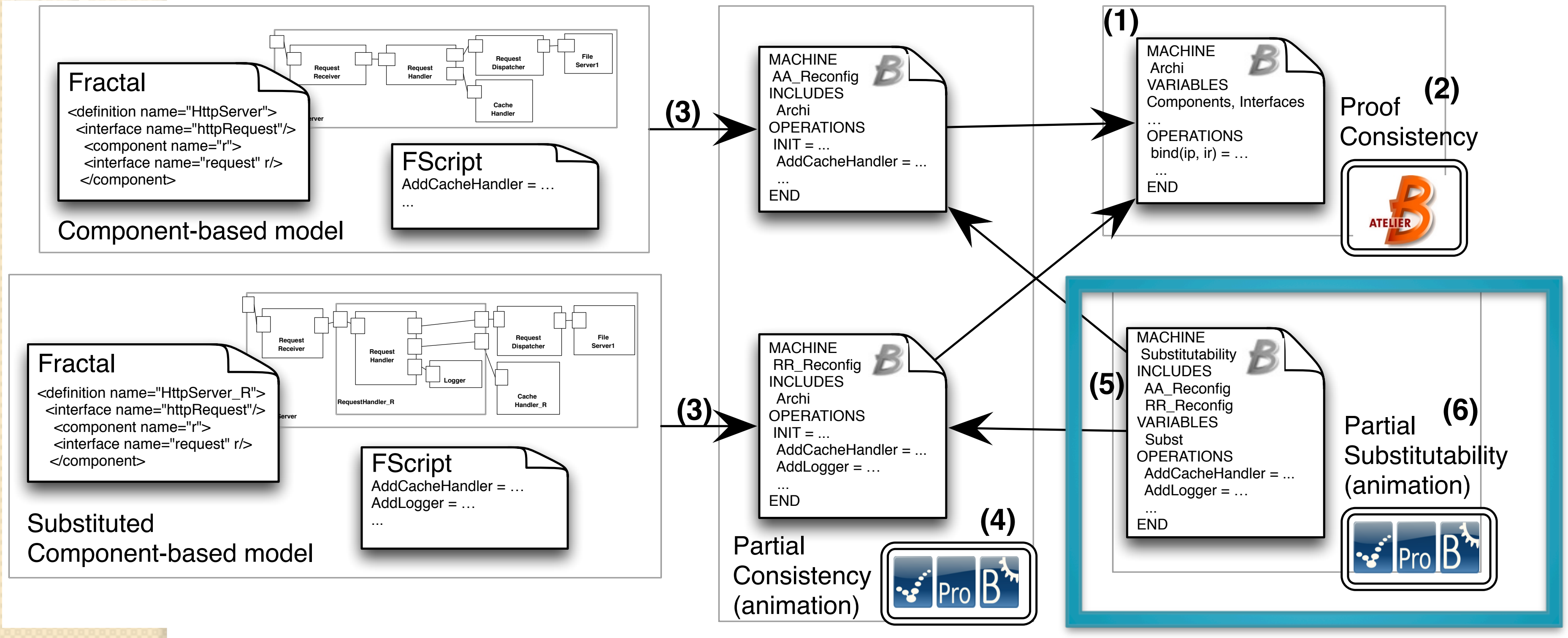
Substitutability constraints?	OK
Strict reconfiguration simulation?	OK
Stuttering reconfiguration simulation?	
Cycle of new reconfigurations?	No
New deadlocks?	No

Principle of the runtime evaluation



Substitutability constraints?	OK
Strict reconfiguration simulation?	OK
Stuttering reconfiguration simulation?	
Cycle of new reconfigurations?	No
New deadlocks?	No

Some experimentations



```
Fractal
<definition name="HttpServer">
  <interface name="httpRequest"/>
    <component name="r">
    <interface name="request" r/>
  </component>
```

```
FScript
AddCacheHandler = …
…
```

```
Fractal
<definition name="HttpServer_R">
  <interface name="httpRequest"/>
    <component name="r">
    <interface name="request" r/>
  </component>
```

```
FScript
AddCacheHandler = …
AddLogger = …
…
```

```
MACHINE
 AA_Reconfig
INCLUDES
 Archi
OPERATIONS
 INIT = ...
  AddCacheHandler = ...
  ...
END
```

```
MACHINE
 RR_Reconfig
INCLUDES
 Archi
OPERATIONS
 INIT = ...
  AddCacheHandler = ...
  AddLogger = …
  ...
END
```

```
MACHINE
 Archi
VARIABLES
Components, Interfaces
 …
OPERATIONS
 bind(ip, ir) = …
  …
END
```

```
MACHINE
 Substitutability
INCLUDES
  AA_Reconfig
  RR_Reconfig
VARIABLES
  Subst
OPERATIONS
  AddCacheHandler = ...
  AddLogger = …
  ...
END
```

Conclusion

Component-based model with reconfigurations

Reconfigurations by component substitution

Substitutability-based simulation, at runtime

Implementation with the B-tools

Thanks for your attention!