Distributed Components

Eric Madelaine
INRIA Sophia-Antipolis, Oasis team

– ProActive-Fractal : main concepts
– Behaviour models for components
– Deployment, management, transformations
– Examples of properties

Fractive's components

- FRACTAL : Component* model specification, implemented using
- ProActive : Java library for distributed applications

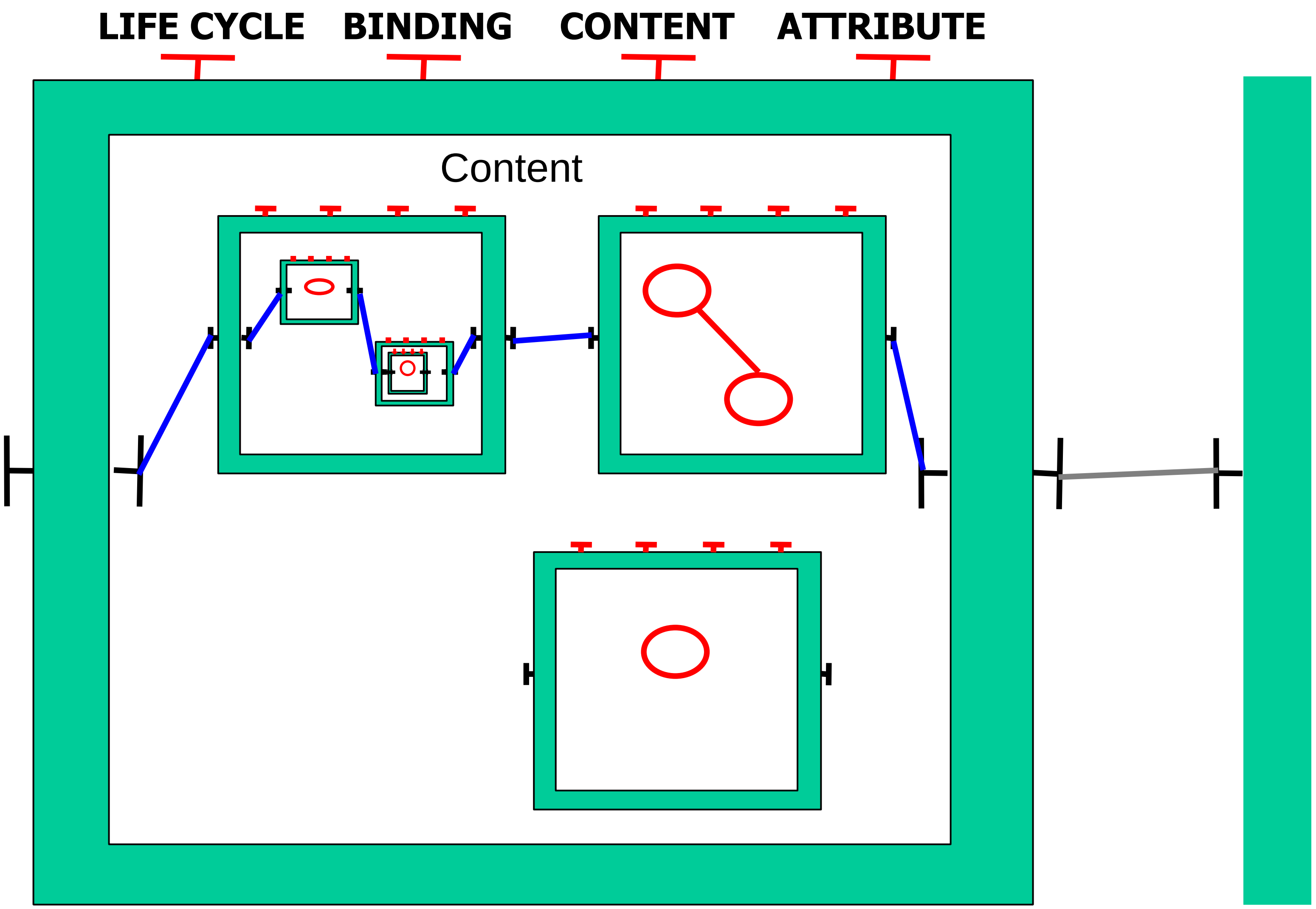
= **Fractive**

- Features:
  - Hierarchical Component Model
  - Separation of functionality / control
  - ADL description (Fractal's XML Schema/DTD)
  - Distributed components (from distributed objects)
  - Asynchronous method calls (non-blocking)
  - Strong Formal Semantics (ASP) => properties and guarantees

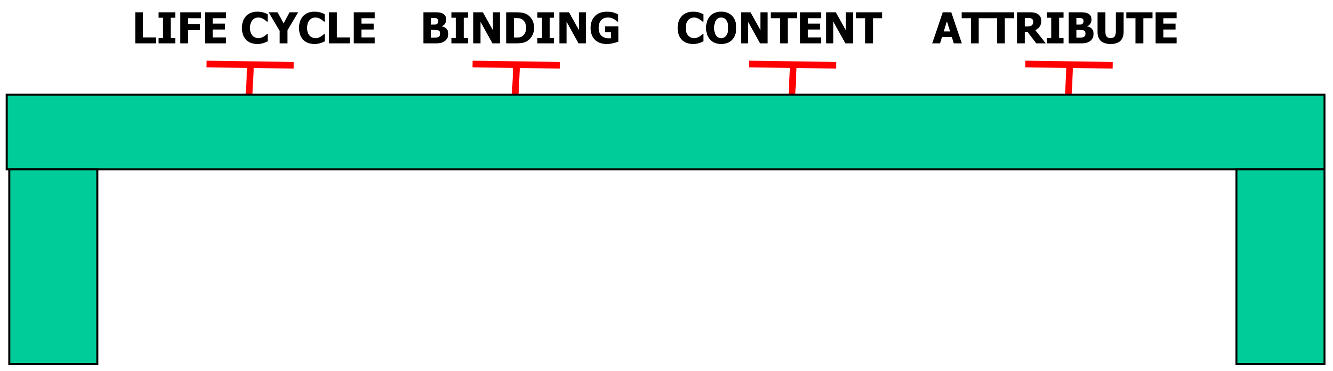
***Component :**

self-contained entity, with well-defined interfaces, reusable, composable (hierarchically)

Fractal's Components



Fractal's Components



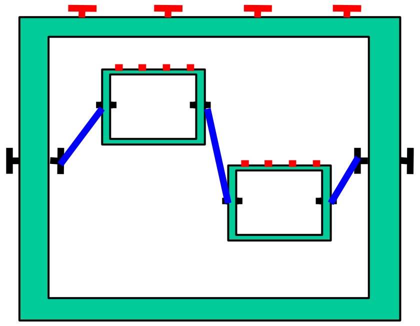
Life-cycle : start / stop the component

Binding : bind / unbind a connection between interfaces

Content : add / remove sub-components

Attribute : get set the value of attribute values

Fractive Behavioural model build

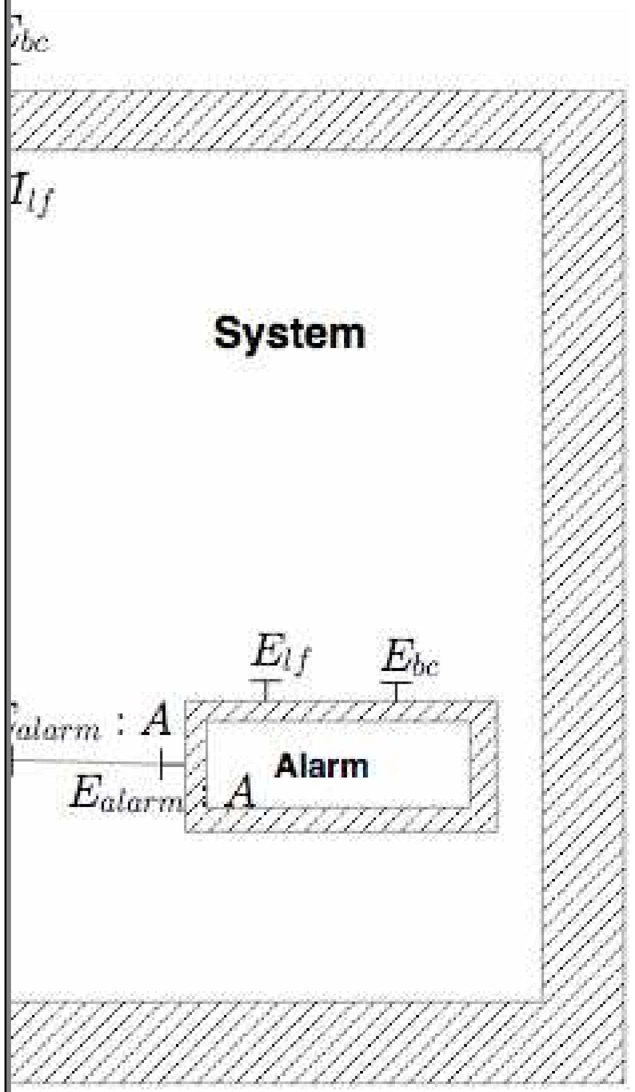


- Functional behaviour is known
  - Given by the user
  - Obtained by static analysis

- Non-functional & asynchronous behaviour is automatically added from the component's ADL
  - Automata within a synchronisation network, named controller

- Component's behaviour is the controller's synchronisation product

System example



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE .... >

<definition name="components.System">

 <component name="BufferSystem"
     definition="components.BufferSystem(3)">
  <interface name="alarm" role="client"
     signature="components.AlarmInterface"/>
 </component>

 <component name="Alarm">
  <interface name="alarm" role="server"
     signature="components.AlarmInterface"/>
 <content class="components.Alarm">
  <behaviour file="AlarmBehav"
     format="FC2Param"/>
 </content>
</component>
 <binding client="BufferSystem.alarm"
     server="Alarm.alarm"/>
</definition>
```

Building the Models: Topology

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE .... >
```

```
nts.BufferSystem">
```

```
<component name="Buffer"
```

```
 <component name="Buffer"
 <interface name="get" role="server"
     signature="components.GetInterface"/>
 <interface name="put" role="server"
     signature="components.PutInterface"/>
  <interface name="alarm" role="client"
     signature="components.AlmInterface"/>
 <content class="components.Alarm">
  <behaviour file="AlarmBehav"
```

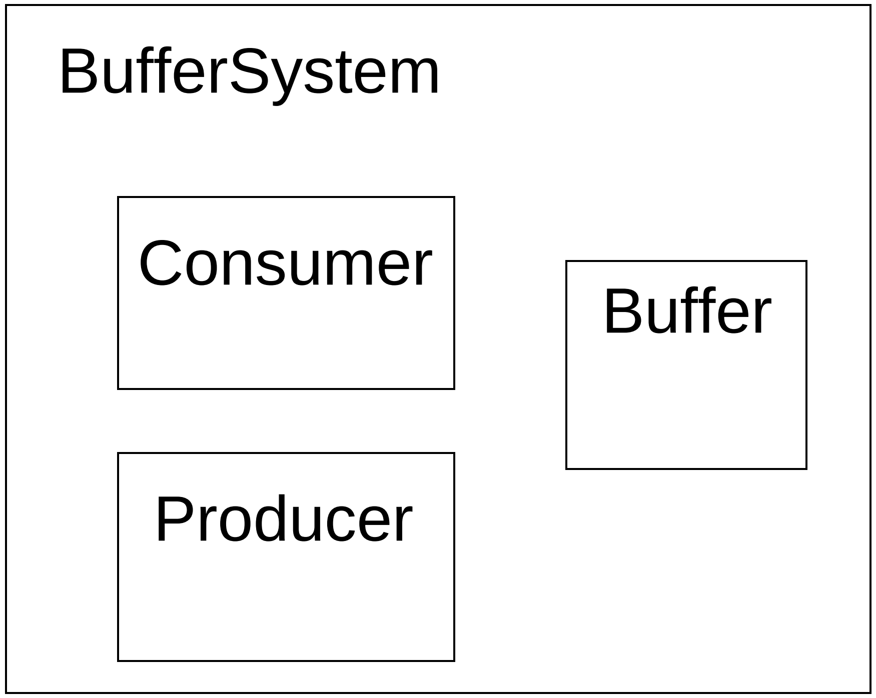
```
<component name="Consumer"
```

```
 <component name="Consumer"
 <interface name="buf" role="client"
     signature="components.GetInterface"/>
 <content class="components.Consumer">
  <behaviour file="ConsBehav"
```

```
<component name="Producer"
```

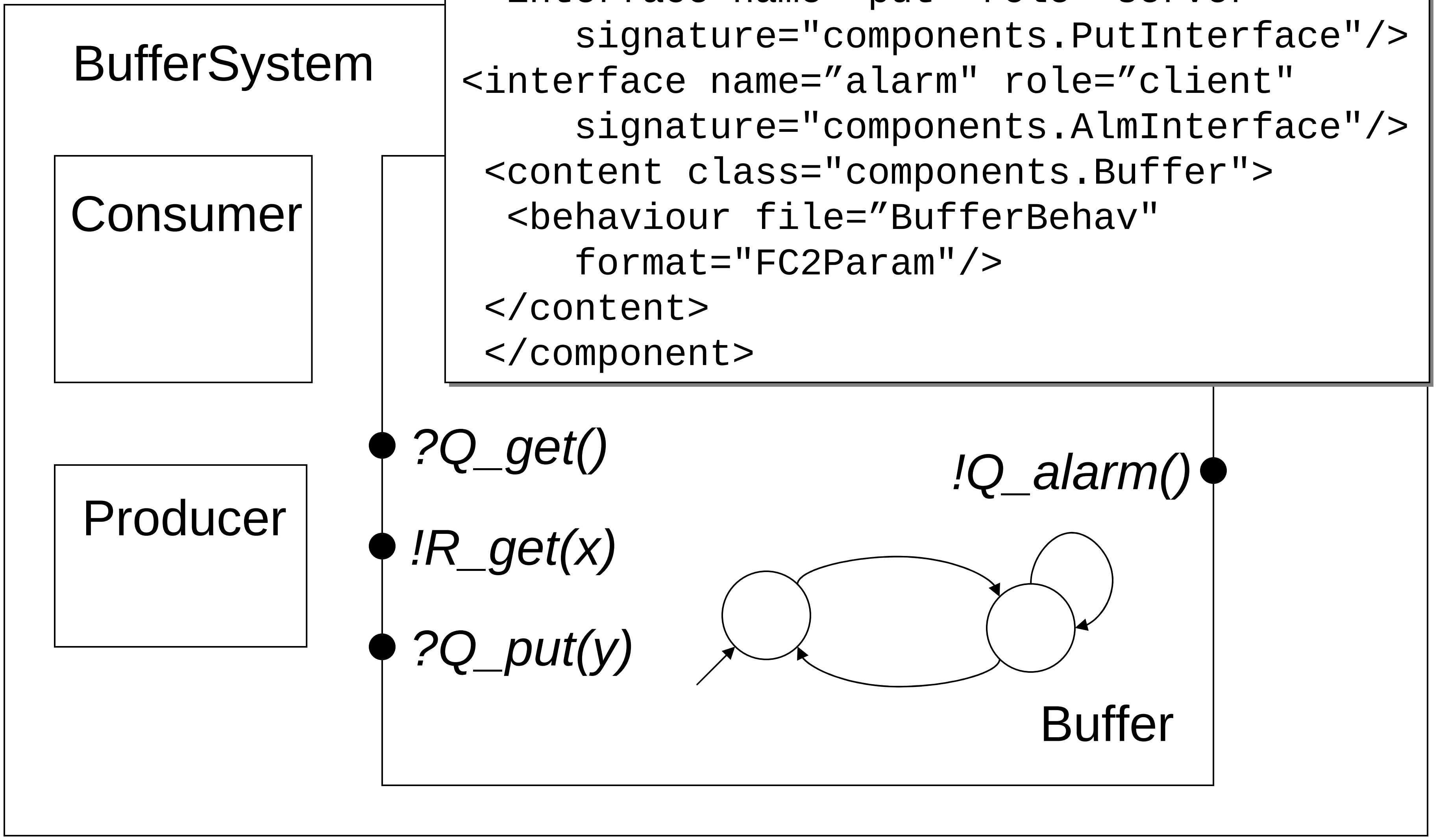
```
 <component name="Producer"
 <interface name="buf" role="client"
     signature="components.PutInterface"/>
 <content class="components.Consumer">
  <behaviour file="ProdBehav"
     format="FC2Param"/>
 </content>
 </component>

<binding client="Producer.buf" server="Buffer.put"/>
<binding client="Consumer.buf" server="Buffer.get"/>
<binding client="Buffer.alarm" erver="alarm"/>
</definition>
```

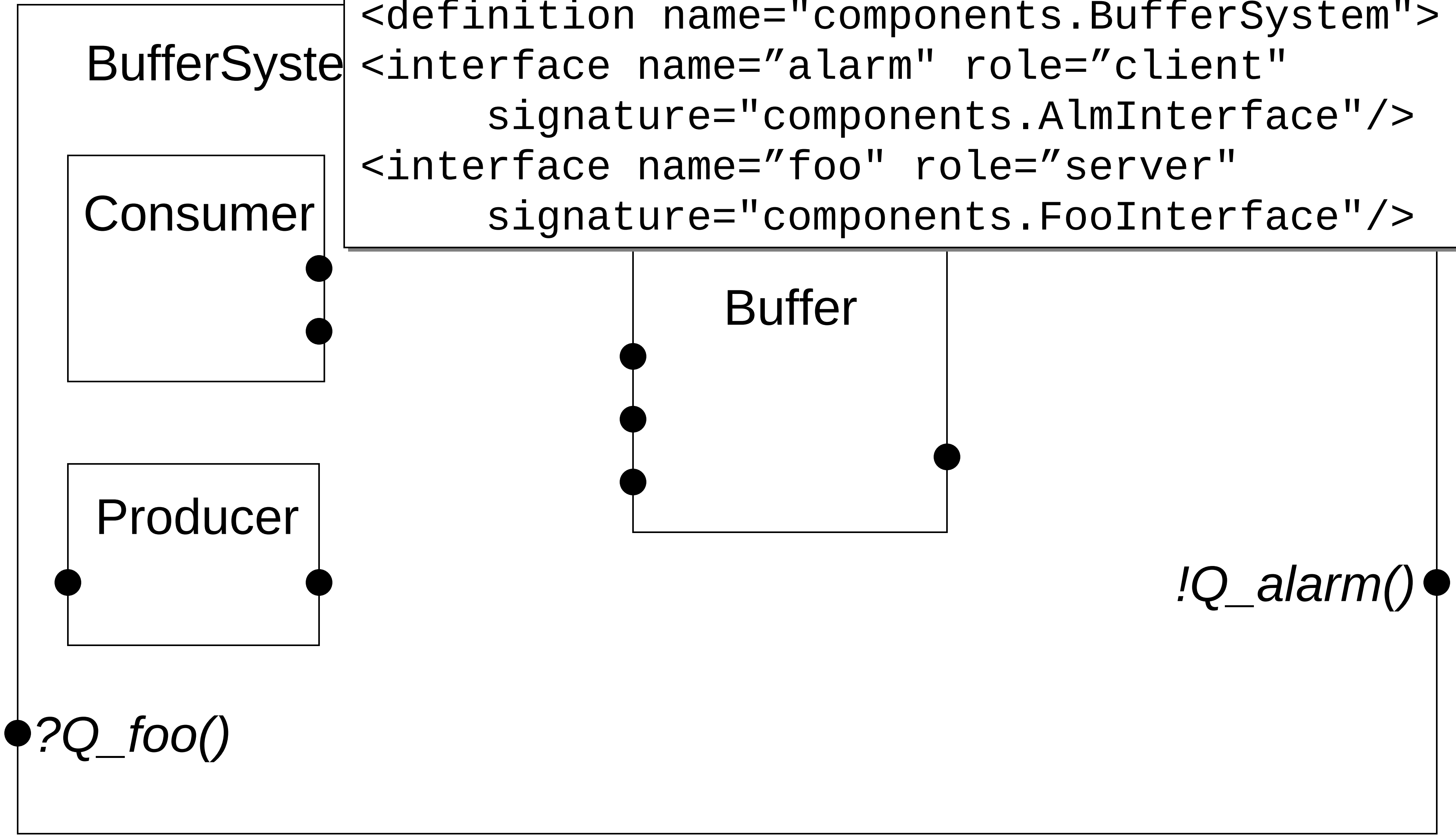


Building

```
<component name=”Buffer"
 <interface name=”get" role=”server"
     signature="components.GetInterface"/>
 <interface name=”put" role=”server"
     signature="components.PutInterface"/>
<interface name=”alarm" role=”client"
     signature="components.AlmInterface"/>
 <content class="components.Buffer">
  <behaviour file=”BufferBehav"
     format="FC2Param"/>
 </content>
 </component>
```

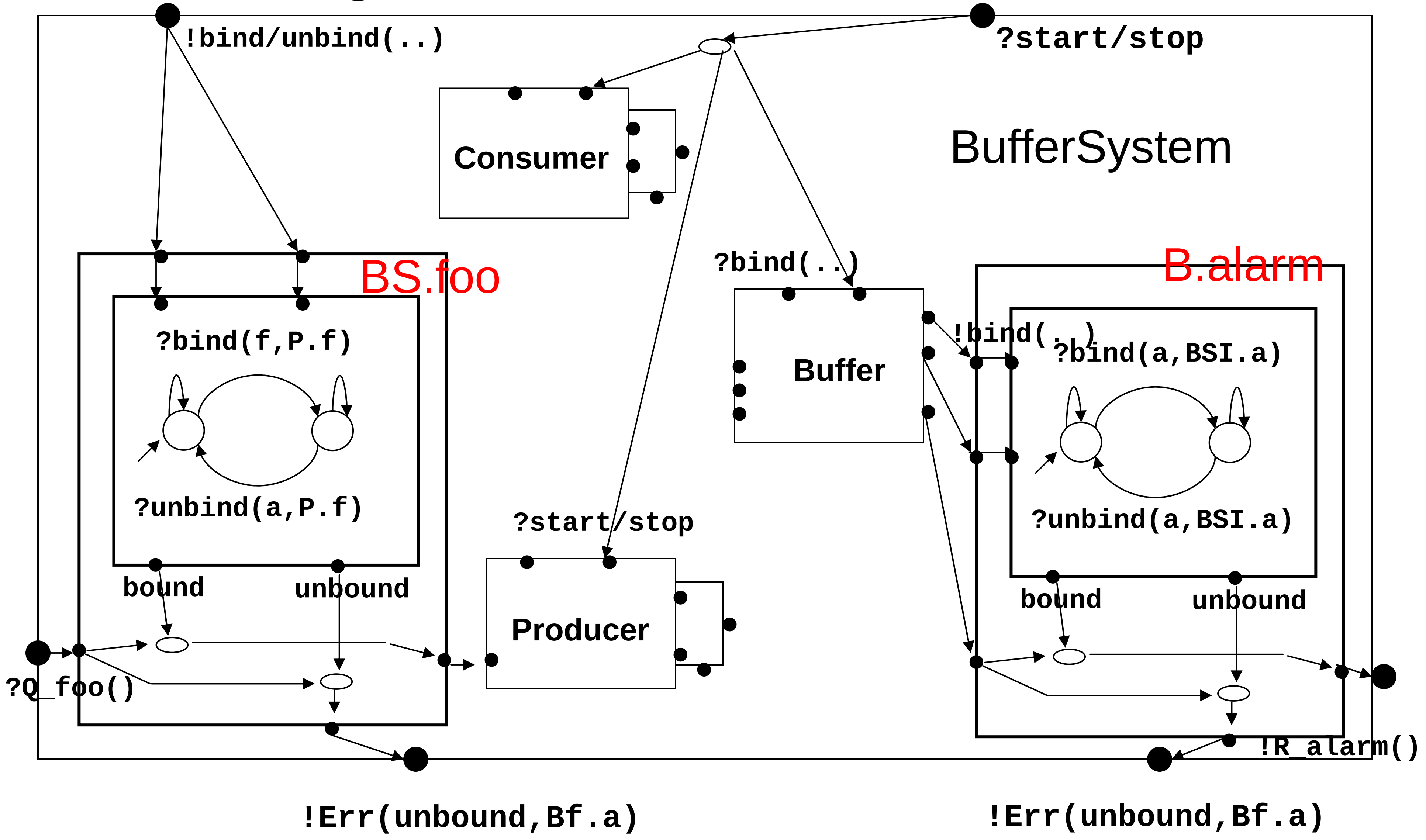


Building the Models: Topology



```
<definition name="components.BufferSystem">
<interface name=”alarm" role=”client"
     signature="components.AlmInterface"/>
<interface name=”foo" role=”server"
     signature="components.FooInterface"/>
```

Building the Models: Non-Functional Behaviour

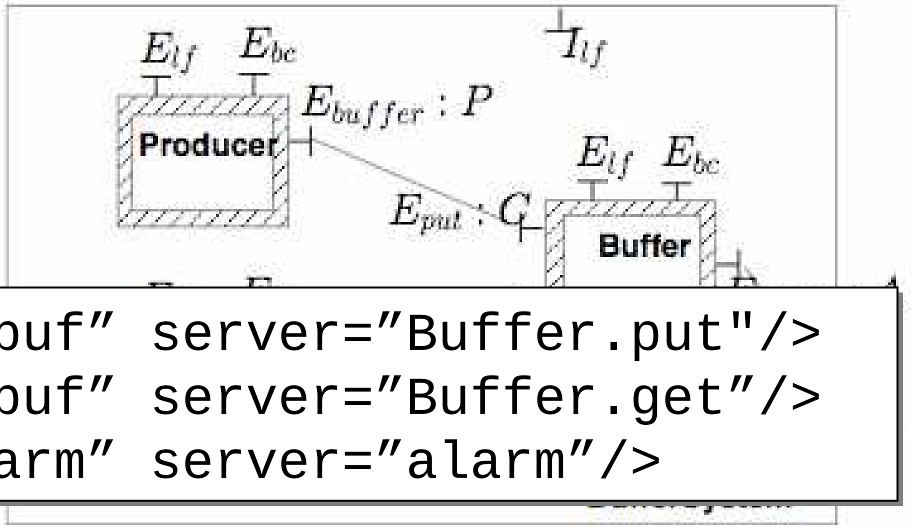


Building the Models: asynchronous behaviour

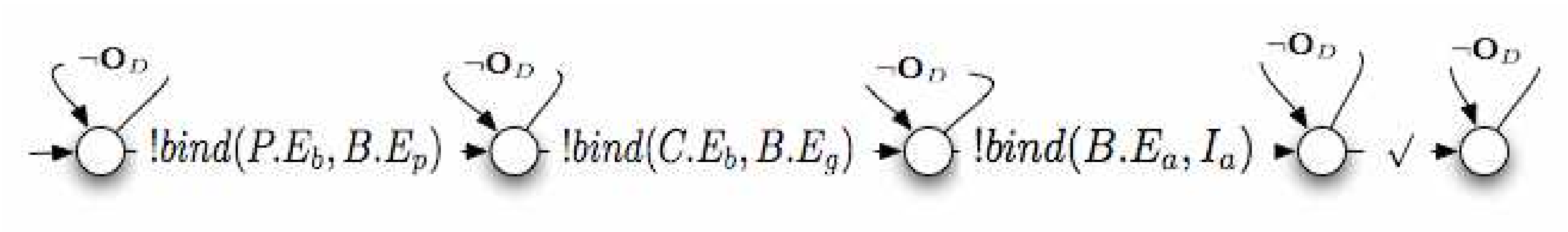
Static Automaton

- Deployment

Automaton



```
<binding client=”Producer.buf” server=”Buffer.put"/>
<binding client=”Consumer.buf” server=”Buffer.get”/>
<binding client=”Buffer.alarm” server=”alarm”/>
```



Static automaton = (Controller || Deployment)

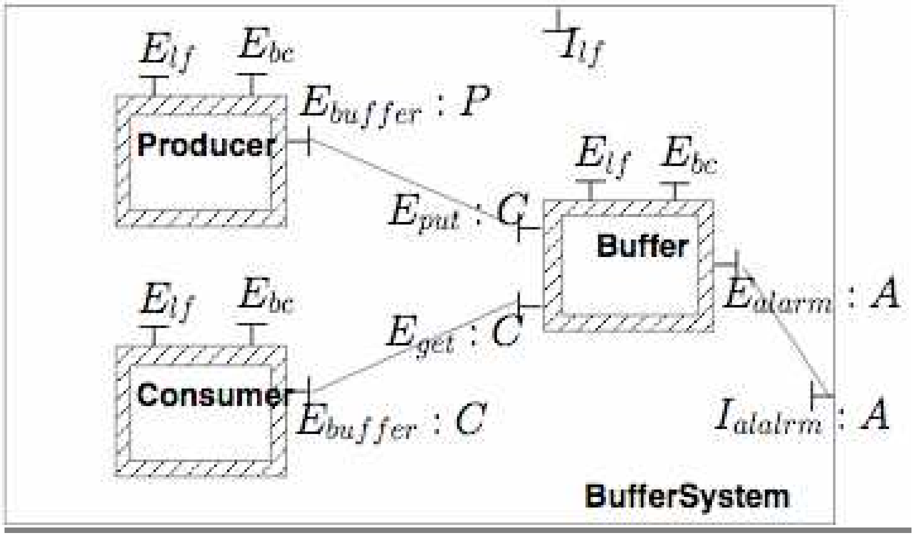
+ *hiding & minimisation*

Properties Verification

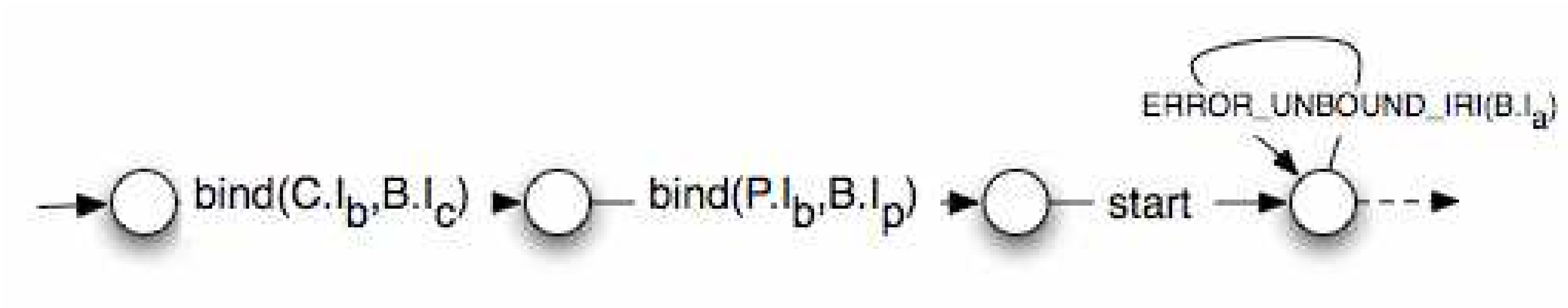
(ACTL)

- Error absence

$\mathbf{AG}_{tru}$



e.g. to start Buffer without linking alarm



Properties Verification

(regular μ-calculus)

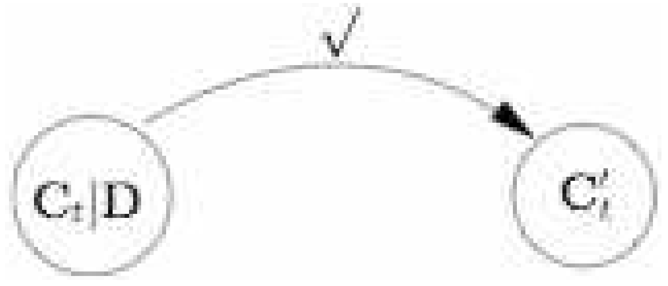
- Functional behaviour (on the static automaton)
  - Get from the buffer eventually gives an answer

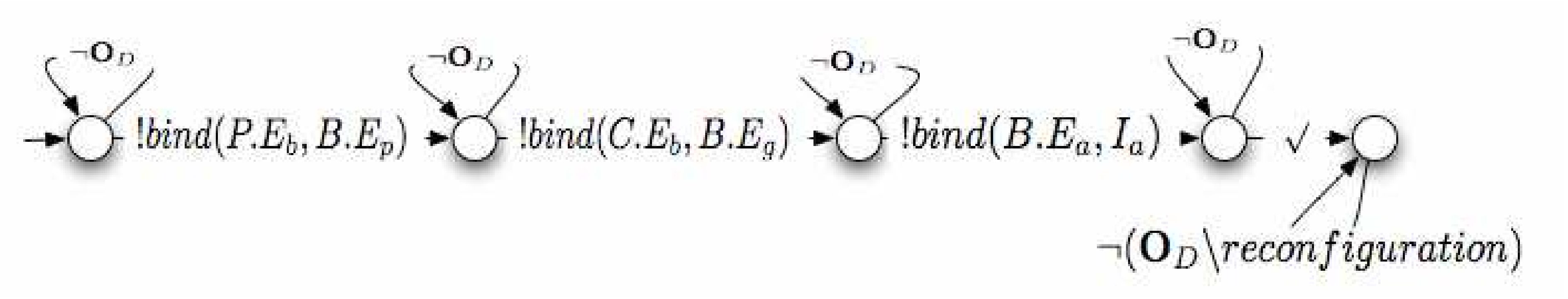
$$[\ \text{true}^*.\text{get\_req}()\ ]\ \mu X.\ (<\ \text{true}\ >\ \text{true} \wedge [\neg \text{get\_rep}()\ ]\ X\ )$$

Properties Verification

(regular μ-calculus)

- Functional under reconfiguration
  - reconfiguration actions are allowed after deployment





Properties Verification

(regular μ-calculus)

- Functional under reconfiguration
  - Future update (once the method served) independent of life-cycle or bindings reconfigurations
  - E.g:

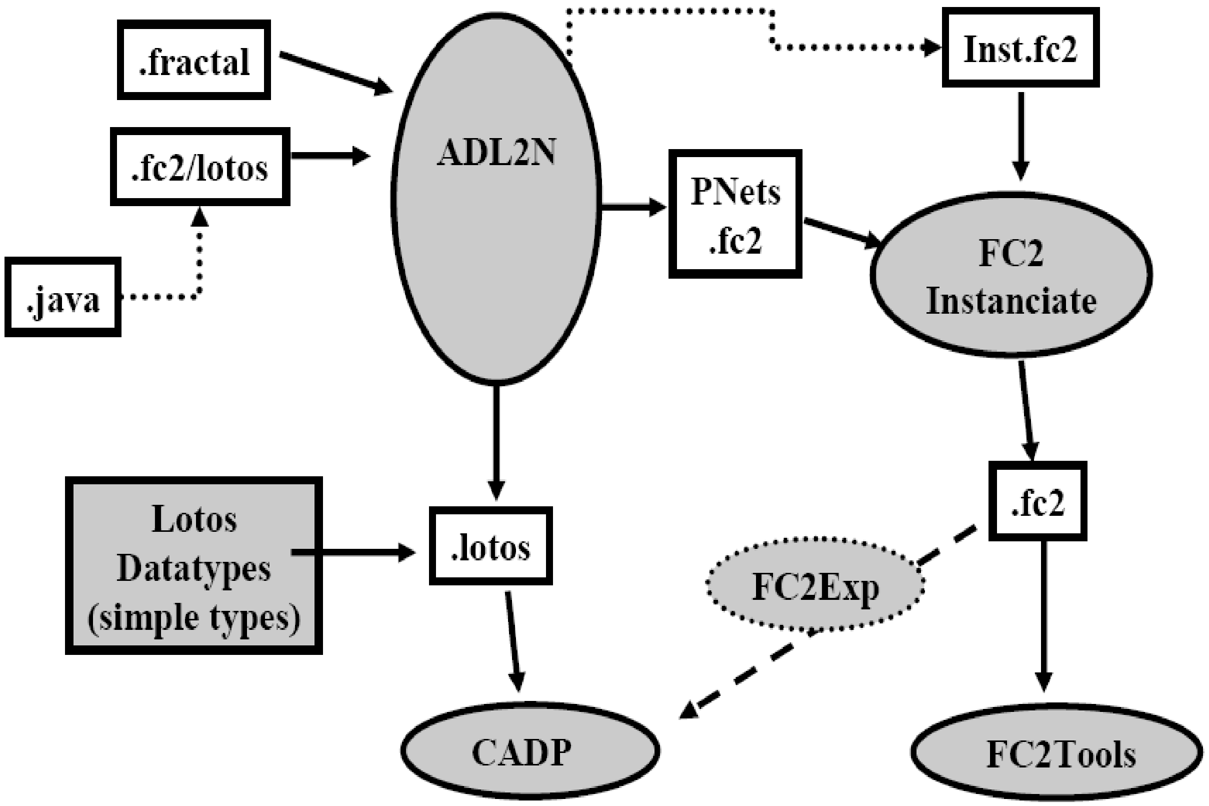
$$[\ \mathrm{true}^*.\mathrm{get\_req}()\ ]\ \mu X.\ (<\ \mathrm{true}\ >\ \mathrm{true} \wedge [\neg \mathrm{get\_rep}()\ ]\ X\ )$$

  - Enabling:

$$?unbind(C.E_b, B.E_g)$$
$$?stop(C)$$

Vercors Platform

- Tool
  - Co
  - Mo
  - Int on tools (av



Supported by FIACRE

An ACI-Security action of the French research ministry

Tools: Pragmatics

Avoiding state explosion

1. Distributed model generation (distributor, CADP)
2. Reduced controllers based on deployment
3. On-the-fly mixed with compositional hiding and minimisation

tion engines.

etworks of

e format.

ers.

More References

– Reference book:

Robin Milner: *Communication and Concurrency*

Prentice Hall, 1989.

– Research: Methods and Toolset for distributed applications and distributed components:

www-sop.inria.fr/oasis/Vercors