Verification of Distributed Components : A Case - study

A. Cansado, L. Henrio, E. Madelaine

OASIS Team,

INRIA -- CNRS - I3S -- Univ. of Nice Sophia-Antipolis

Fractal workshop, Nantes, 3 july 2006

Agenda

- Context
  - Behaviour Specifications for Components
- The ADL and the Specifications
  - A Proposal for Fractal ADL V3
- The AirPort Case-study
  - Presentation, and Zoom-in
  - Specification and Verification with the Vercors platform
- Conclusion & Perspectives

Specification of Software Components

Aim :

Build reliable components from the composition of smaller pieces by the use of their formal specifications.

Component paradigm : only observe activity at interfaces.

Behavioural properties:

Deadlock freeness, progress/termination, safety and liveness.

Applications :

- Build complex systems from off-the-shelf components
- Check behavioural **compatibility** between sub-components
- Check correctness of component **deployment** and **behaviour**
- Check correctness of the **transformation** inside a running application.

Specification of Software Components

Behaviour Specifications :

Interface Signatures are not Sufficient, we need formal Behaviour Specifications of Components for:

=> detection of composition errors (deadlocks, progress, termination)

=> correctness of deployment and component update

=> compliance

Contributions :

- **Behavior Protocols** (SOFA, Fractal) :
- **Parameterized Networks** (ProActive, Fractal) :

Agenda

- Context
  - Behaviour Specifications for Components
- The ADL and the Specifications
  - A Proposal for Fractal ADL V3 (Sofa and Oasis teams)
- The AirPort Case-study
  - Presentation, and Zoom-in
  - Specification and Verification with the Vercors platform
  - Group communication
- Conclusion & Perspectives

Extension of Fractal ADL : A Proposal
(INRIA-Oasis, Charles Univ. – Sofa)

- Integration into the standard is important :

  To foster the usage of behaviour specification, and of formal verification of component composition

  To enable comparison of approaches on common examples

- Minimal Changes and Openness :

  The Behaviour specification itself can be big, and left to an external Parser.

  Let the syntax OPEN to other formalisms.

Extension of Fractal ADL : A Proposal

A **behavior** element

nested inside component, interface, or binding elements

with :

A **language** attribute,

with current possible values **FC2**, **Lotos, behavior-protocol**

A **file** or a **value** attribute,

Extension of Fractal ADL : A Proposal

Examples:

```
<component name="Alarm">
  <interface … />
  <content class="Alarm"/>
  <controller desc="primitive"/>
  <behaviour file="Alarm.lotos"
              language="Lotos"/>
</component>
```

```
<component name="IFirewall">
    <interface name="IFirewall" role = "server" />
    <content class="IFirewallImpl"/>
    <controller desc="primitive"/>
    <behaviour language="behavior-protocol"/>
        value = " ? IF.Enable* | ? IF.Disable* "/>
</component>
```

Specialized parsers, often already existing,

- for behaviour languages,
- and for additional environment information (linking the ADL and interface signature elements with the behaviour specification)

Agenda

- Context
  - Behaviour Specifications for Components
- The ADL and the Specifications
  - A Proposal for Fractal ADL V3
- The AirPort Case-study
  - Presentation, and Zoom-in
  - Specification and Verification with the Vercors platform
  - Group Communication
- Conclusion & Perspectives

The AIRPORT Wifi Network Case-study

Origin : France-Telecom / Sofa (Charles Univ. Prague)
Component Reliability Extensions for the Fractal Model

Description : http://kraken.cs.cas.cz/ft/public

Bibliography :

FACS'05: ***Model Checking of Component Behavior Specification: A Real Life Example***

Internet and Databases

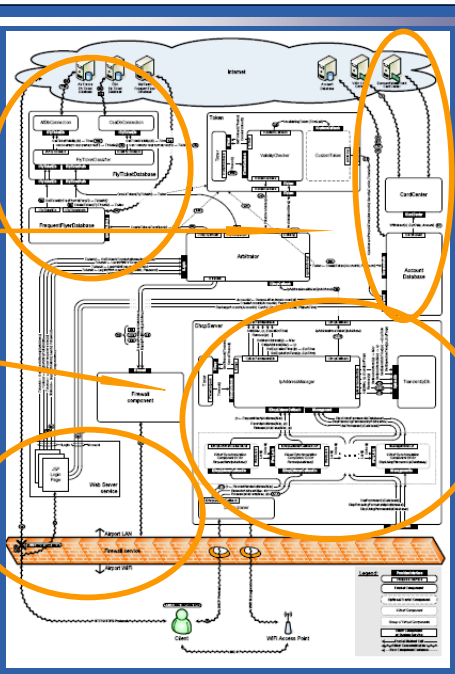
Ticket, FrequentFlyer, and Card management

DHCP server

Firewall and

Airport web server

Users



Zooming in …

This presentation

Login session

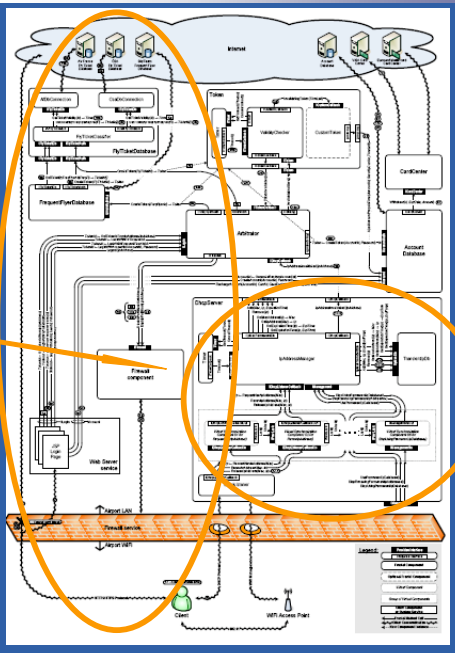
Firewall+Web server

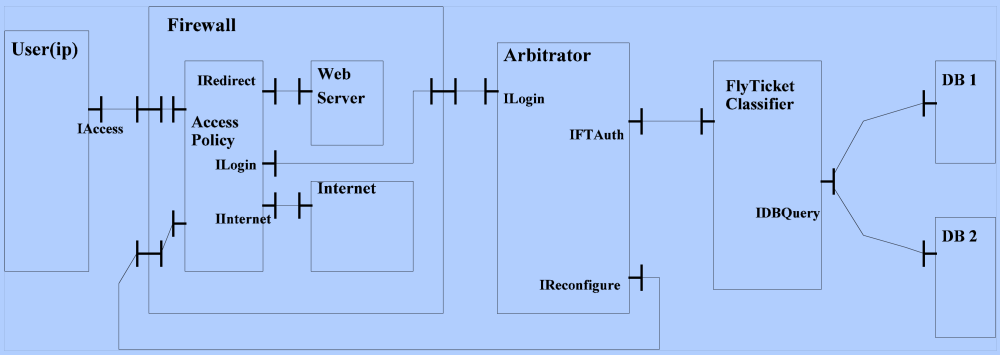
FACS'05 study (SOFA)

Arbitrator / Token dispenser

Ticket Classifier

Frequent Flyer Classifier



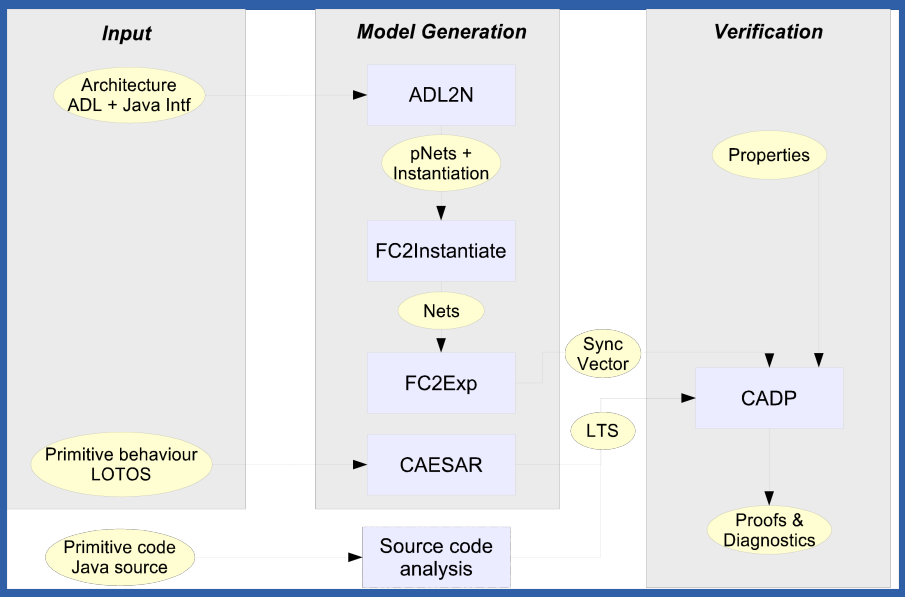


Login to the airport network : simple scenario

Choices :

Specify User and Firewall as components

Abstract away from time constraint (Token lease)



Supported by FIACRE, An ACI-Security Action of the French research ministry

Model Generation : the ADL2N tool

Semantic formalism :

pNets = parameterized synchronisation networks (**Forte'04**)

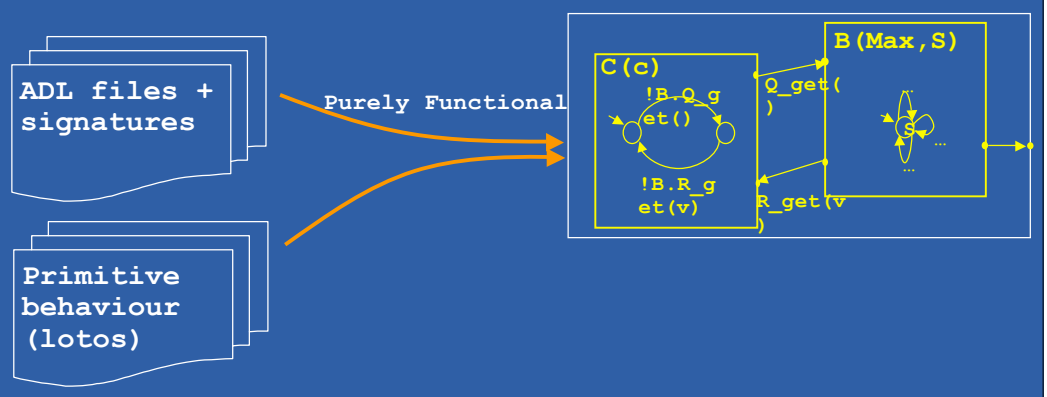
Composite Components : ADL -> pNets

Generation of pNets from the Architectural Description + Interface signatures

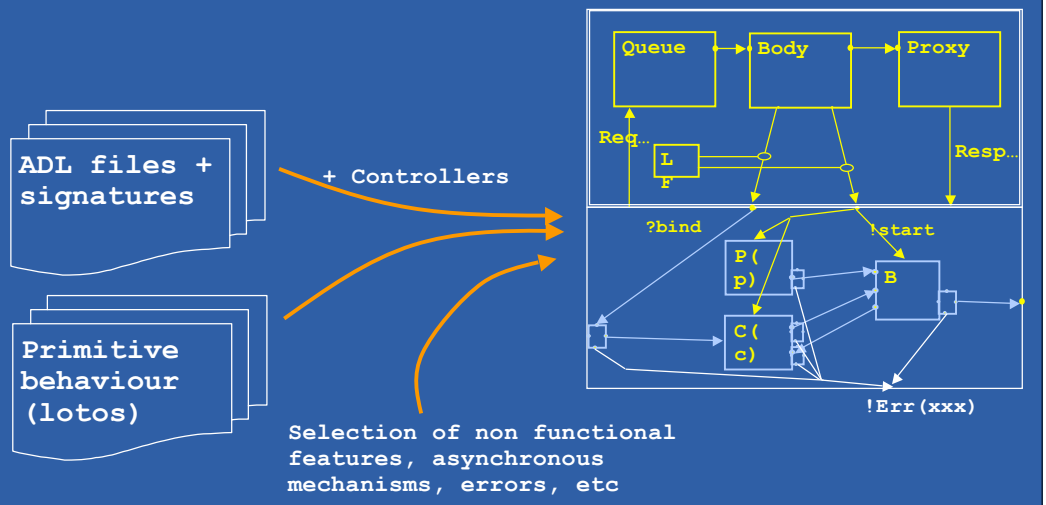
Spin'05 : Behavioural Models for Hierarchical Components

Facs'05 : Behavioural Models for Distributed Components

Model Generation : the ADL2N tool



Model Generation : the ADL2N tool



Building the global model

Simple scenario, purely functional.

Branching minimization, keeping all upper level events visible.

Instantiation :

1 single user

3 web pages, 2 tickets, 2 databases

Sizes :

global system = 2152 st / 6553 trs minimized = 57 st / 114 trs

biggest primitive component = 5266 st / 27300 trs

Mastering the complexity :

- Smaller representations (i.e. partial orders, symmetries)
- Reduce the number of visible events
- Use distributed verification tools
- **Reason at component level**

Proving Properties

- Press-button verification :
  - Finding Deadlocks
  - Absence of usual errors :
    - J. Adamek “Composition Errors”
    - ProActive “Deployment Errors”
- Logical languages :
    - CADP : regular μ-calculus
    - Specification patterns
- Custom scenarios specifying the execution environment
- Equivalence / Compliance with a specification

Proving Properties

- Deadlock : our initial specification has one.

Diagnostic :

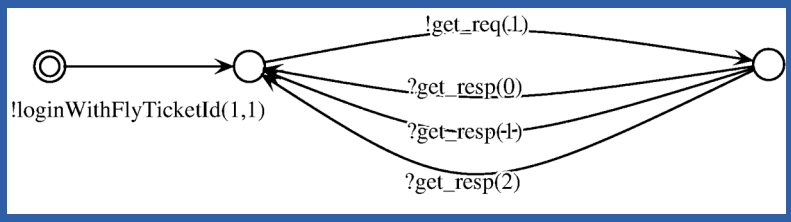
<initial state>
""loginWithFlyTicketId(IAccess)(0,1,1)""
""loginWithFlyTicketId(ILogin)(0,1,1)""
""loginWithFlyTicketId(IAccess)(0,1,1)""
""CreateToken_req(IFTAuth)(1,1)""
""GetFlyTicketValidity_req(IFTAuth)(1,1)""
""GetFlyTicketValidity_resp(IFTAuth)(1,1)""
""CreateToken_resp(IFTAuth)(1)""
<deadlock>

Asks twice for loggin

The Arbitrator (synchronous and monothreaded) gets the first answer, but blocks on the second request.

Custom Scenario / Environment

- **Restrict the possible behaviours of our User component :**



(Specified in LOTOS and compiled with CAESAR (CADP))

=> Result : no deadlock.

Proving Properties

- Functional property:

  If user is not logged in, it will always be redirected

  [not('login(0,1)')*.'get_resp(0,1)'] false

  If the user asks for loggin, will he inevitably get it ?

  [""" You don't like μ-calculus ?
  Use specification patterns…

Agenda

- Context
  - Behaviour Specifications for Components
- The ADL and the Specifications
  - A Proposal for Fractal ADL V3
- The AirPort Case-study
  - Presentation, and Zoom-in
  - Specification and Verification with the Vercors platform
  - Group Communication
- Conclusion & Perspectives

Second Scenario

Instantiation domains:

- Each ticket has 2 values
- Each DB sends 0 or 1 ticket
- The Query returns 0 to 2 tickets

111634 states
202380 trans.
39 labels

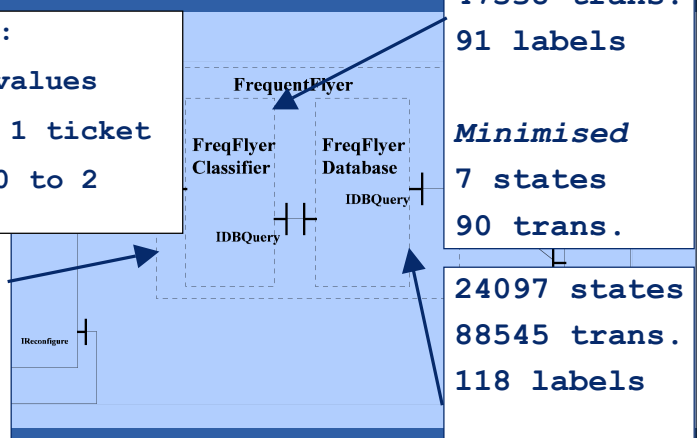
Minimised
12 states
92 trans.

24634 states
47538 trans.
91 labels

Minimised
7 states
90 trans.

24097 states
88545 trans.
118 labels

Minimised
98 states
360 trans.



rcast interface to severa
ned inside a composite co

Proposal for ADL Extension: GRID'S Specifics

Collective Interfaces (Matthieu Morel)

Distributed, parallel components require specific methods for distributing and gathering information.

New attribute :

cardinality = "multicast" (1 to n)
cardinality = "gathercast" (n to 1)

⇒ ProActive implementation
(methods-wise or args-wise policies specified as Fractlets in the component code)

⇒ Model generation for behaviour verification
(methods/args policies specified as abstractions in the ADL2N tool)

Agenda

- Context
  - Behaviour Specifications for Components
- The ADL and the Specifications
  - A Proposal for Fractal ADL V3
- The AirPort Case-study
  - Presentation, and Zoom-in
  - Specification and Verification with the Vercors platform
  - Group Communication
- Conclusion & Perspectives

Current Status

Tool set :

- Code analysis (prototype, partial)
- Model generation (V0.8 available)
- Interactions with model-checking and the CADP verification toolbox (available)

Ongoing work

Model generation :

- Including NF-controllers, asynchronous semantics
- Multicast Interfaces

Expression of Properties :

- Pattern language specific to Grid Application

Perspectives:

- Parameterized components at specification level

Conclusions

- Formalisms for specifying the dynamic behaviour of components
- Tools for building finite models of behaviours from the Architecture Description
- Realistic Use-case

Question (to FT & Sofa) :

Available for comparisons of formalisms, methods, tools ?

Papers, Use-cases and Tools at :

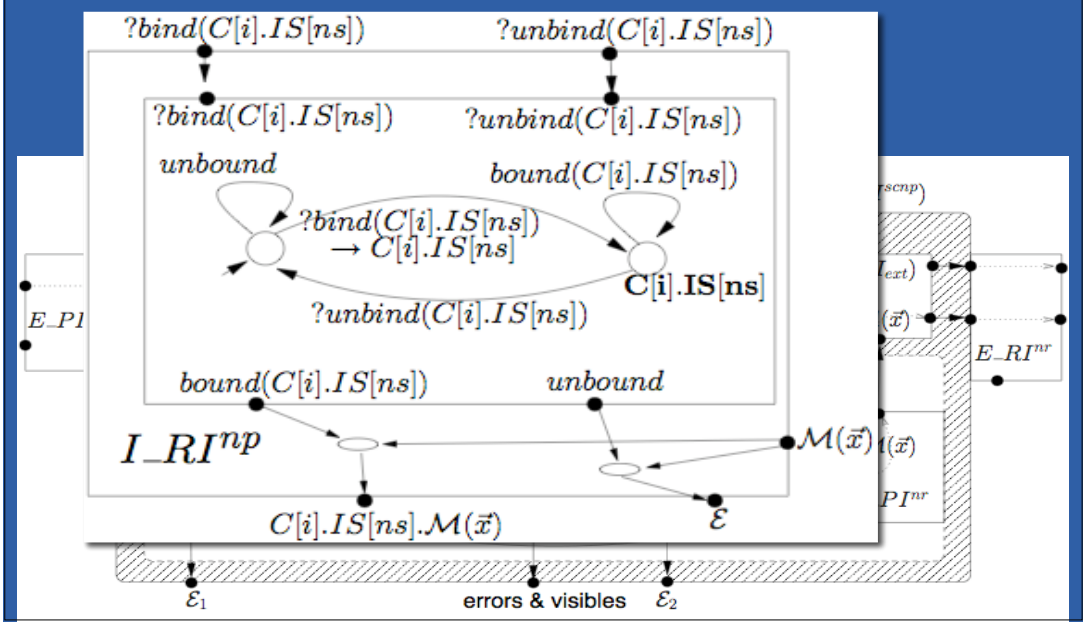
http://www-sop.inria.fr/oasis/Vercors

Thank you.

Papers, Use-cases and Tools at :

http://www-sop.inria.fr/oasis/Vercors

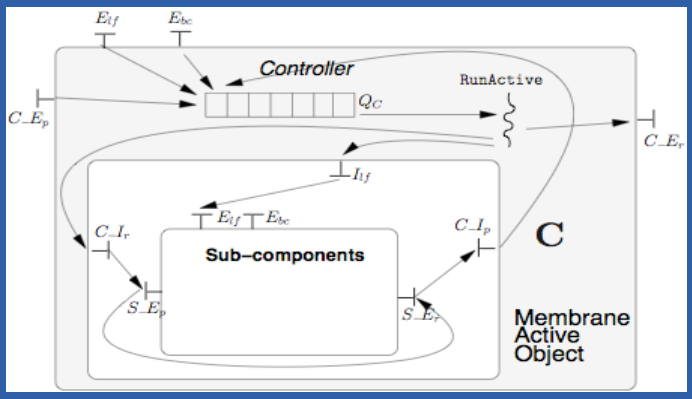
Fractive Behavioural Models



Fractive implementation

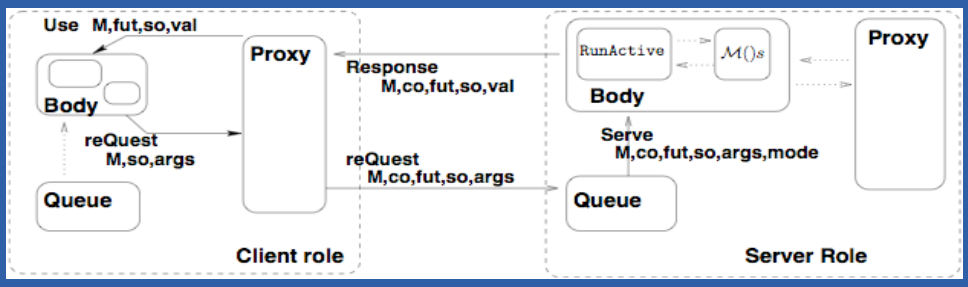
Primitive component => Active object

Composite membrane => Active object



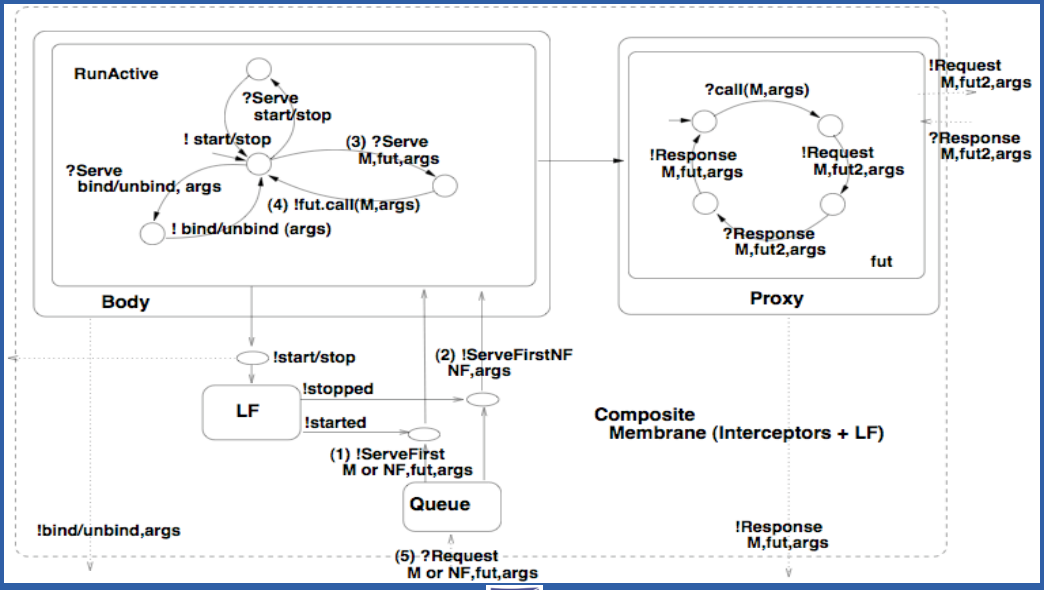
Previous work: ProActive behavioural models

(presented at Forte 2004)



T. Barros, R. Boulifa, E. Madelaine: Parameterized Models for Distributed Java Objects, Forte'2004 Conference, Madrid, Sep. 2004, LNCS 3235, © Springer-Verlag

Fractive composites



3) Membrane for primitive Asynchronous components

= request queues, f

4) … and also for composites components