# Model-checking distributed components: The *Vercors* platform

Tomás Barros
Antonio Cansado
Eric Madelaine

# Agenda

- Introduction
- Theoretical outline
- Case Study
- Platform overview
- Work in progress and Future Work
- Conclusion

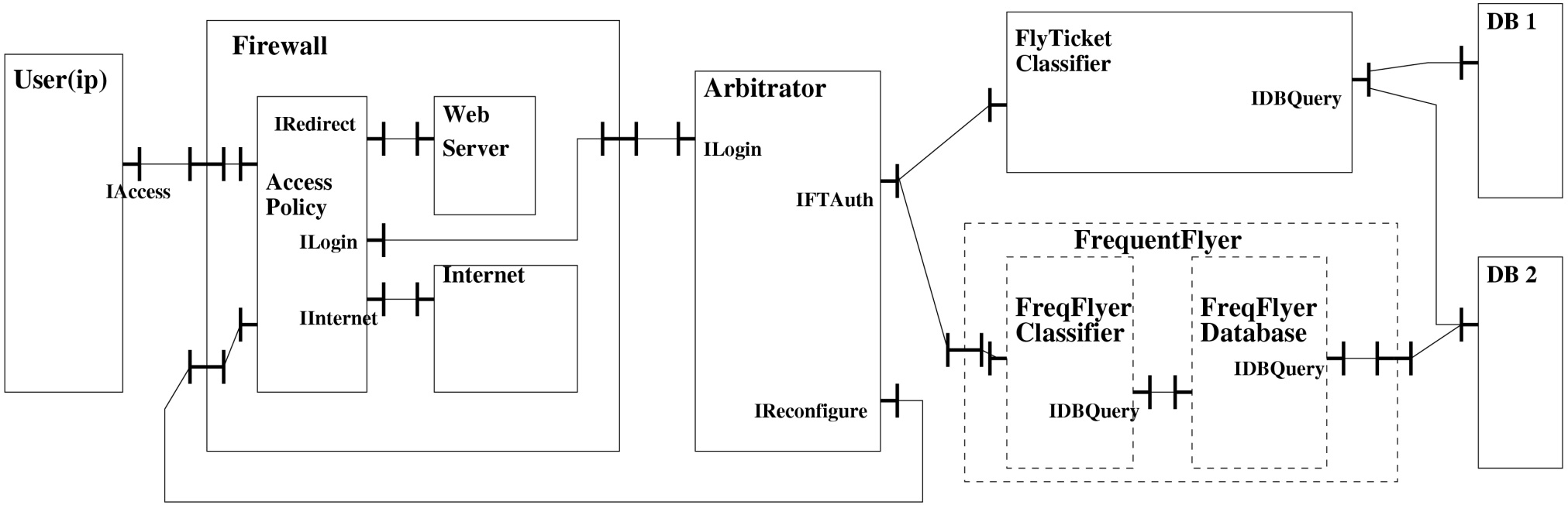
# Agenda

- Introduction
- Theoretical outline
- Case Study
- Platform overview
- Work in progress and Future Work
- Conclusion

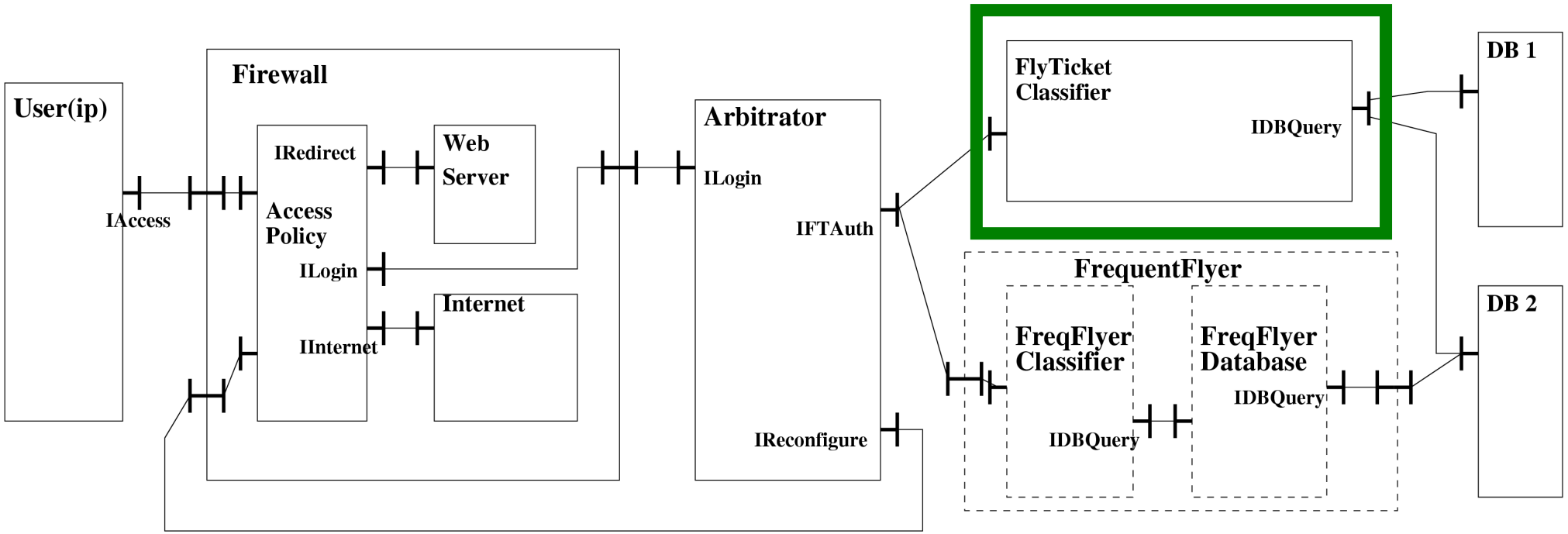
# Introduction

- Component programming as a mean to split software
  - Only observable through its interfaces
- Fractal component model
  - Hierarchical components
- Distributed components: Fractive
  - ProActive's implementation of Fractal

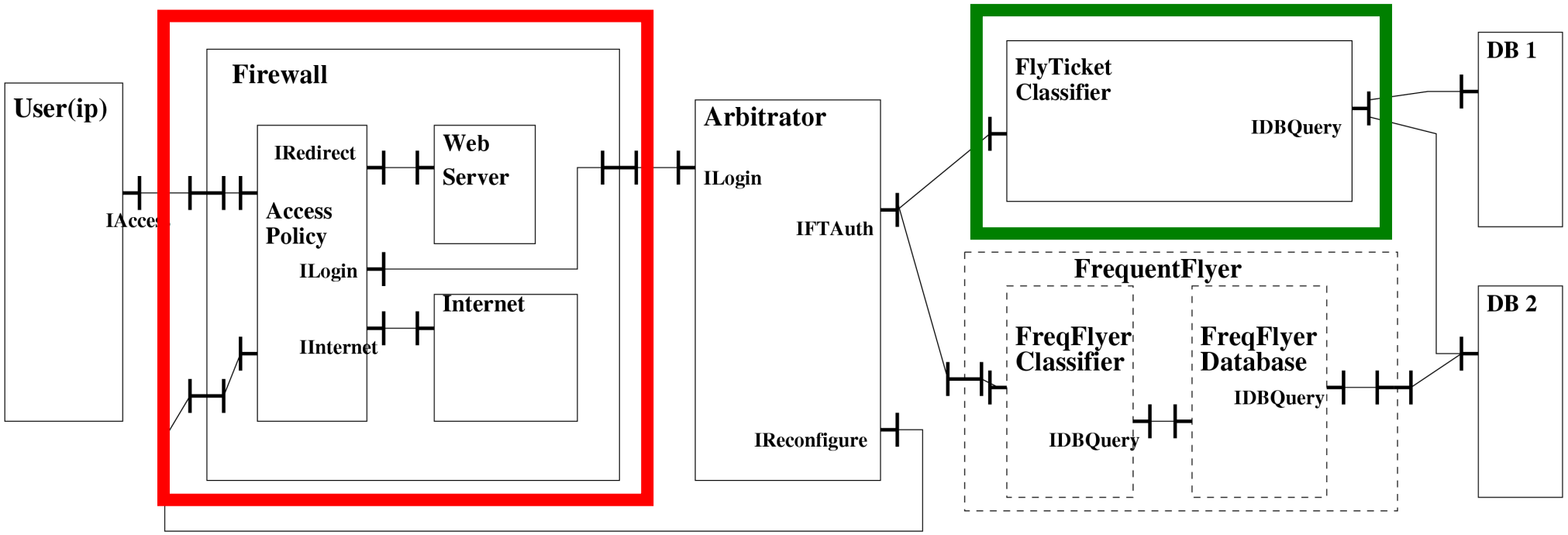
# Fractal component model



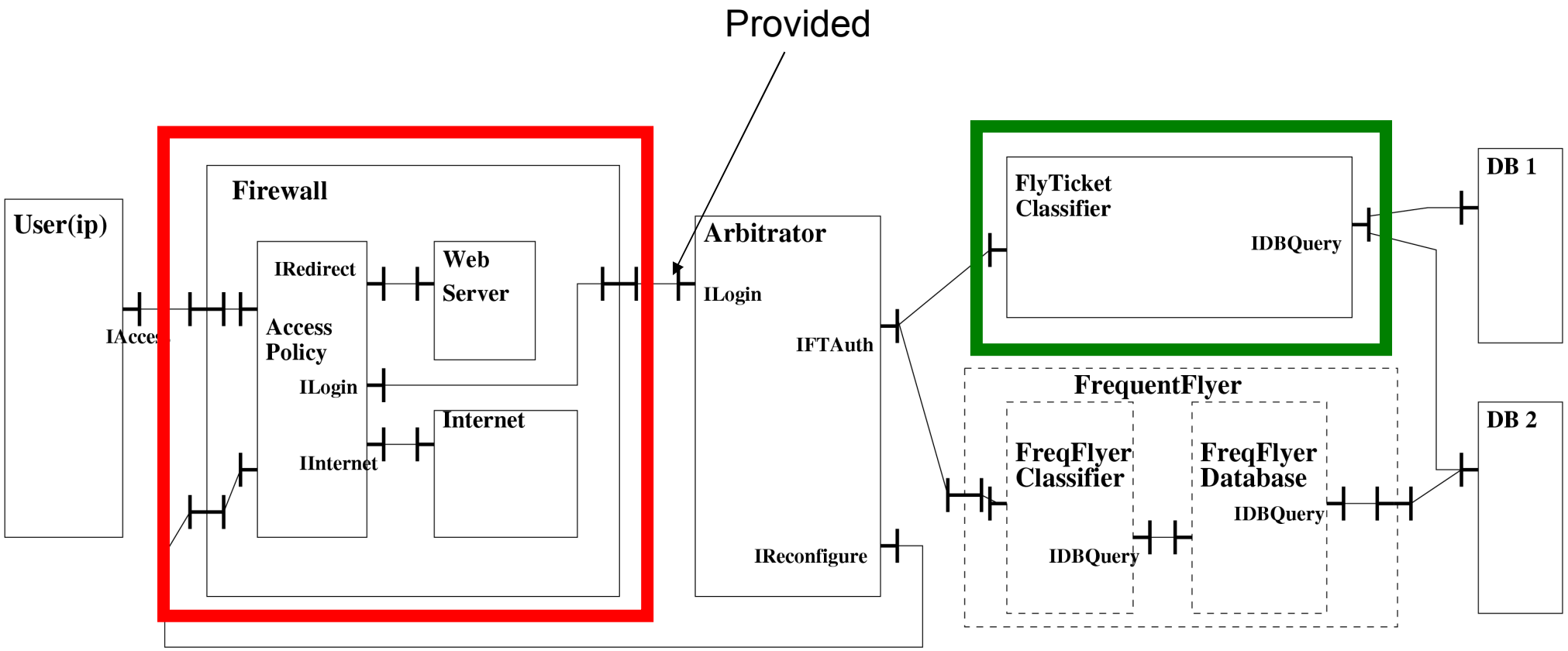
# Fractal component model



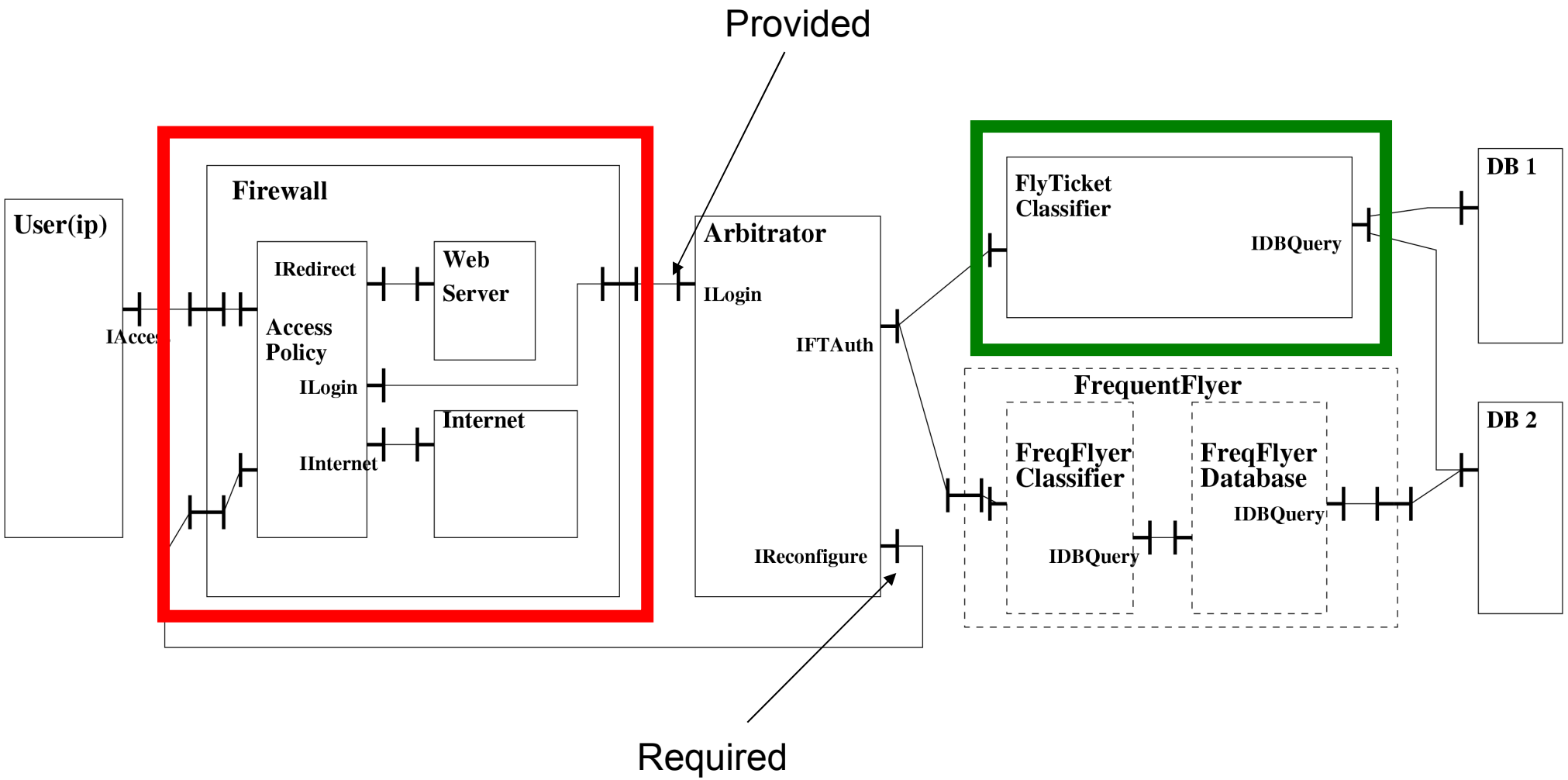
# Fractal component model



# Fractal component model



# Fractal component model



# Introduction

- Static typing of bound interfaces is not enough
  - Dynamic behavioural compatibility is mandatory to ensure a correct assembly of off-the-shelf components
- Formal methods
  - Strong mathematical background
  - Gap between formal specifications and implementations
- *Provide a framework for non-specialists*

# Agenda

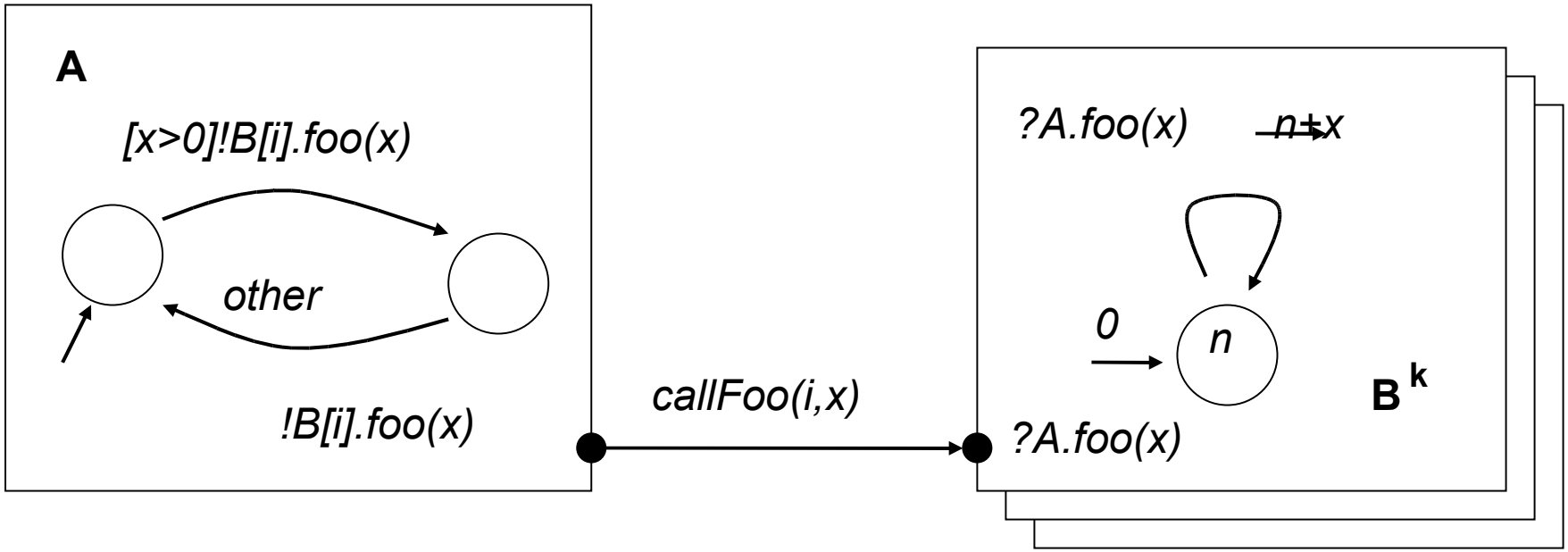
- Introduction
- Theoretical outline
- Case Study
- Platform overview
- Work in progress and Future Work
- Conclusion

# Basics

- Bisimulation theories
  - Congruence properties
- Semantic model: pNets
- Previously defined models
  - Functional behaviour
  - Non-functional behaviour

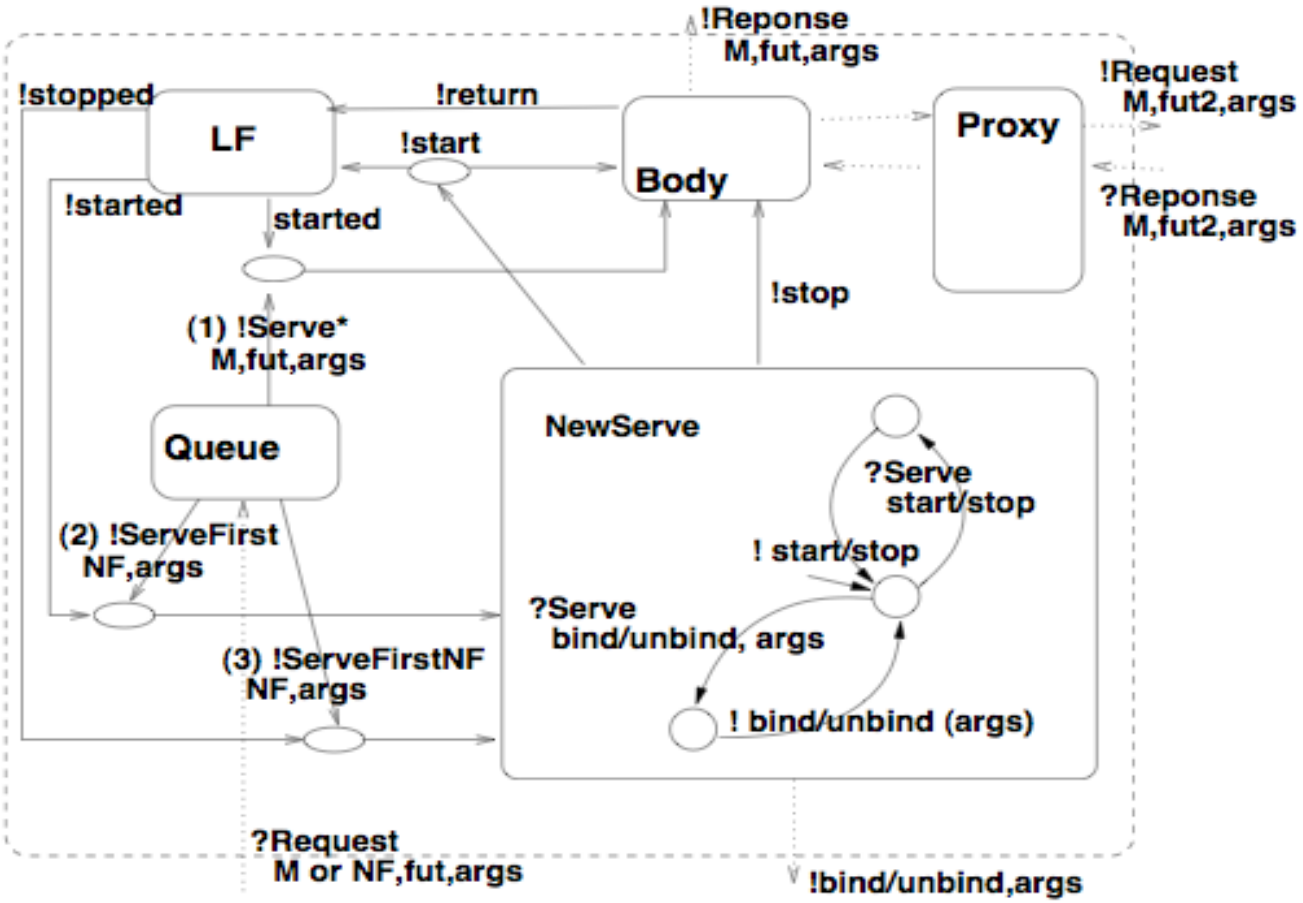
# Basics

- Semantic model: pNets

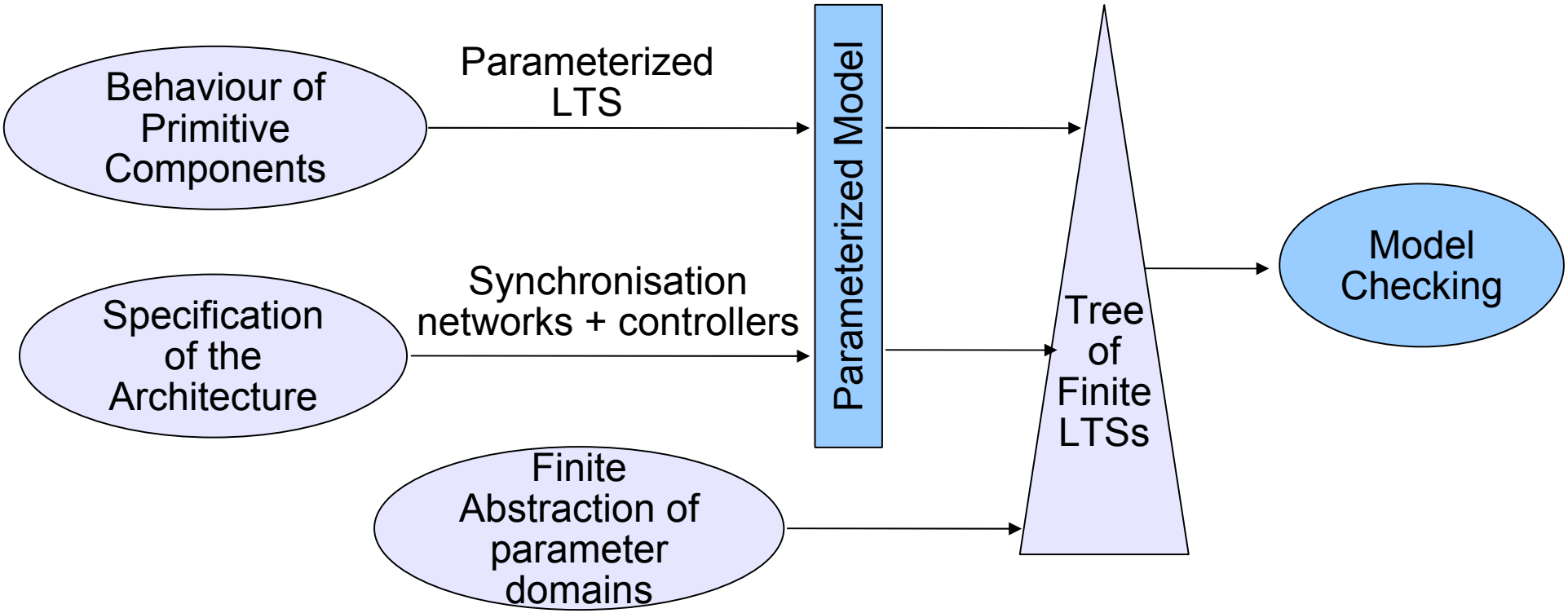


# Basics

- Previously defined models



# Theoretical principals

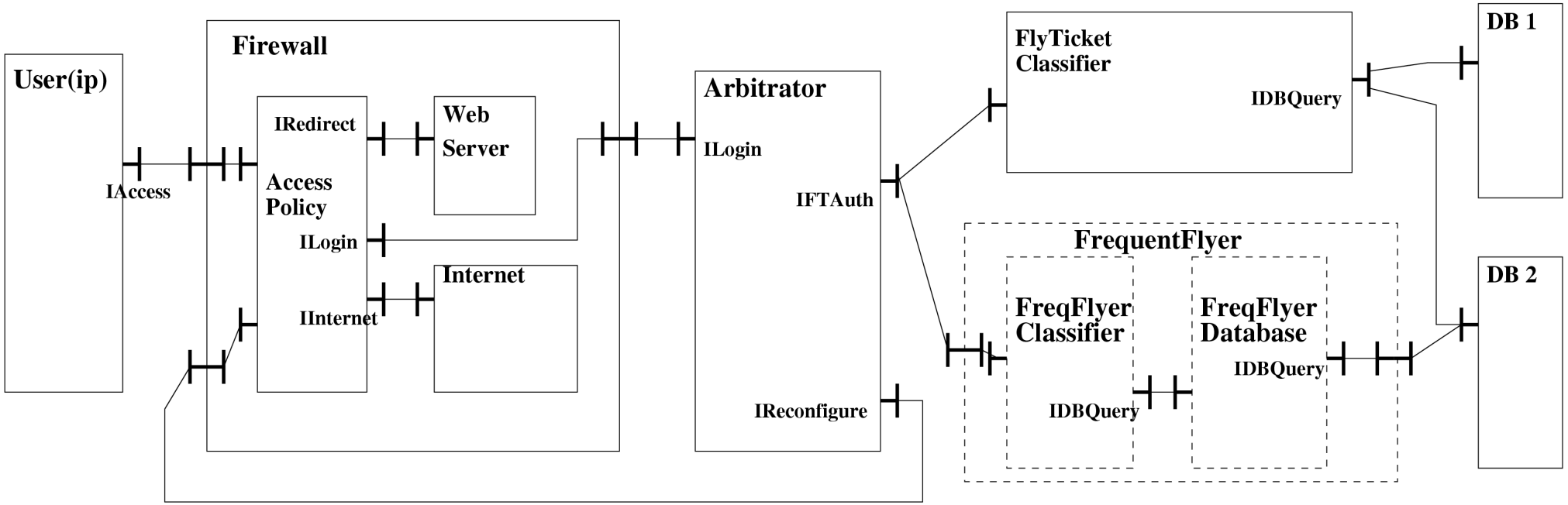


# Behaviour of Primitives

- Functional behaviour
  - Given by the user
  - Static source code analysis
  - Currently supported languages: FC2 and LOTOS
- Usage
  - Parameterized LTS encoding the behaviour specification

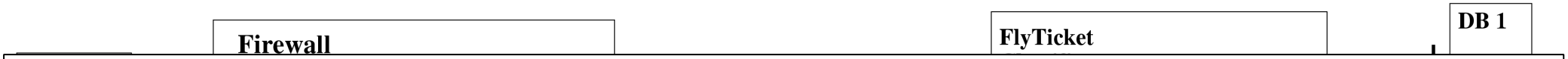
# Architecture Specification

- Architecture Description Language (ADL)



# Architecture Specification

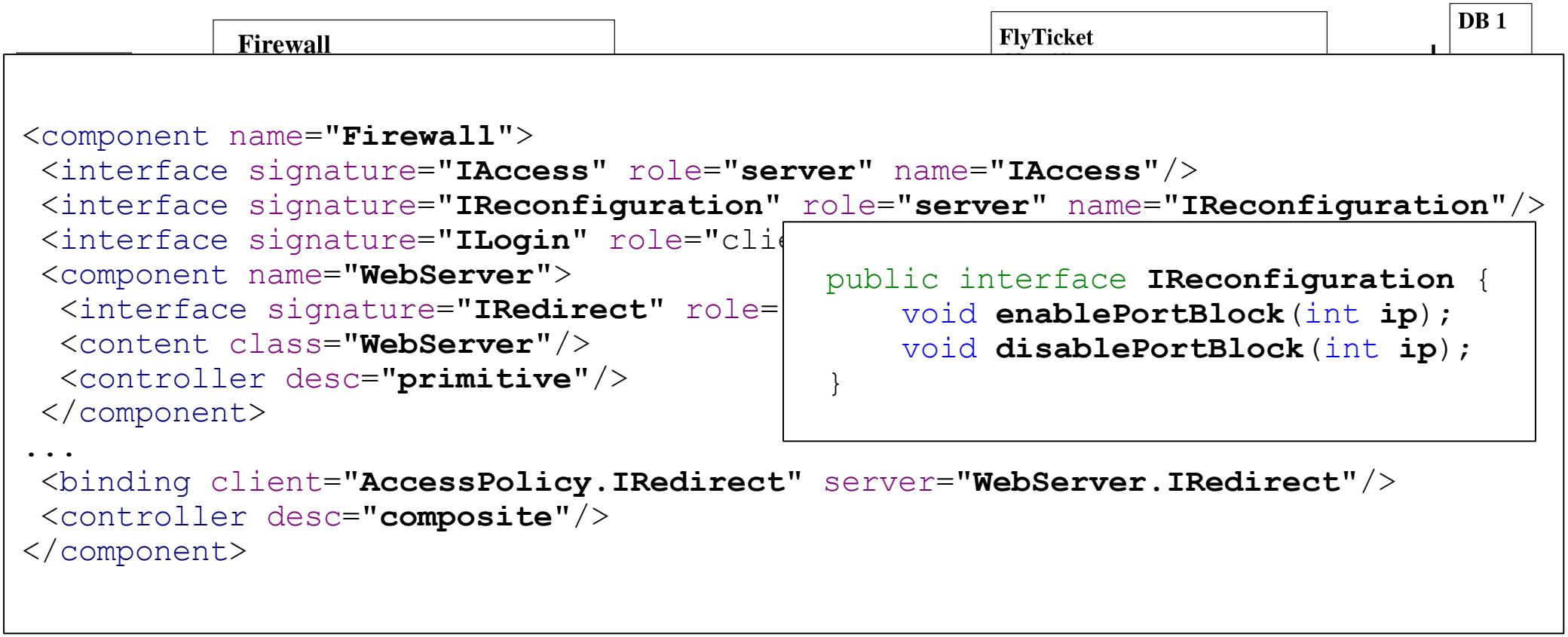
- Architecture Description Language (ADL)




```
<component name="Firewall">
 <interface signature="IAccess" role="server" name="IAccess"/>
 <interface signature="IReconfiguration" role="server" name="IReconfiguration"/>
 <interface signature="ILogin" role="client" name="ILogin"/>
 <component name="WebServer">
  <interface signature="IRedirect" role="server" name="IRedirect"/>
  <content class="WebServer"/>
  <controller desc="primitive"/>
 </component>
...
 <binding client="AccessPolicy.IRedirect" server="WebServer.IRedirect"/>
 <controller desc="composite"/>
</component>
```

# Architecture Specification

- Architecture Description Language (ADL)



# Architecture Specification

- Architecture Description Language (ADL)
- Usage
  - Create Synchronisation networks modelling component interaction
  - Create controllers with NF aspects

# Abstraction

- Simple datatypes (first order countable domains)
- Finite instantiation of a parameterized model is an abstraction in the sense of Cleaveland and Riely
- Partitions with finite number of distinguished concrete values, plus one or more for others

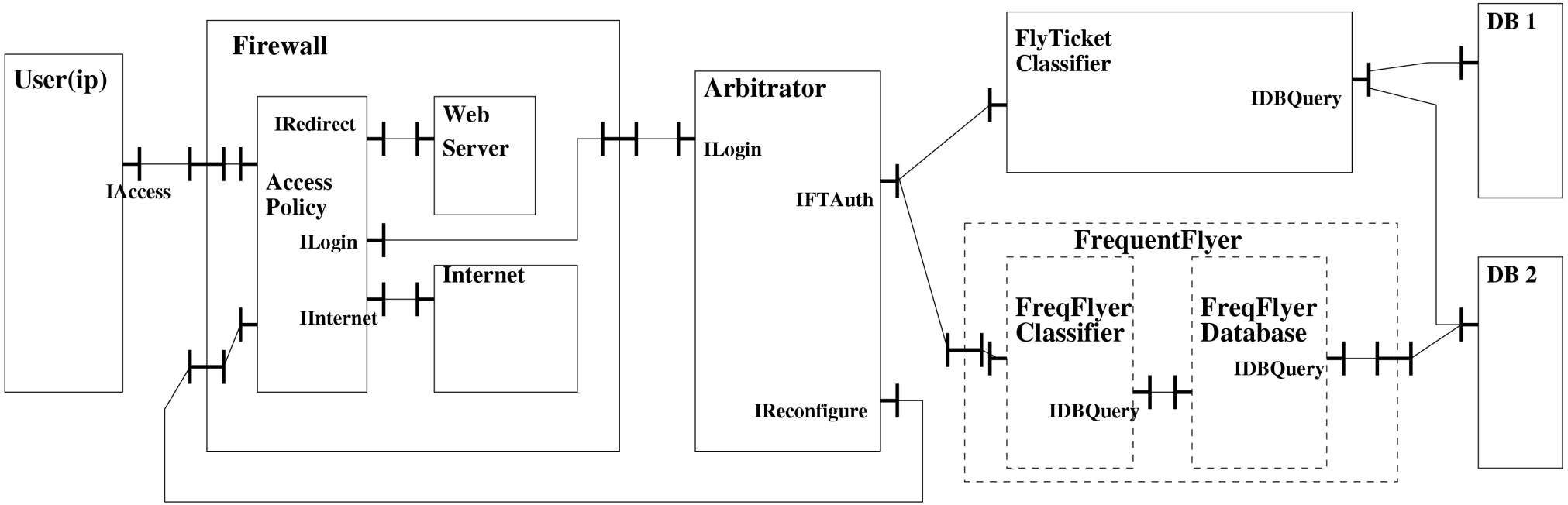
# Agenda

- Introduction
- Theoretical outline
- Case Study
- Platform overview
- Work in progress and Future Work
- Conclusion

# Case Study

- WiFi Internet access
- France Telecom and SOFA team in Prague
- Proposed as a standard Fractal component example

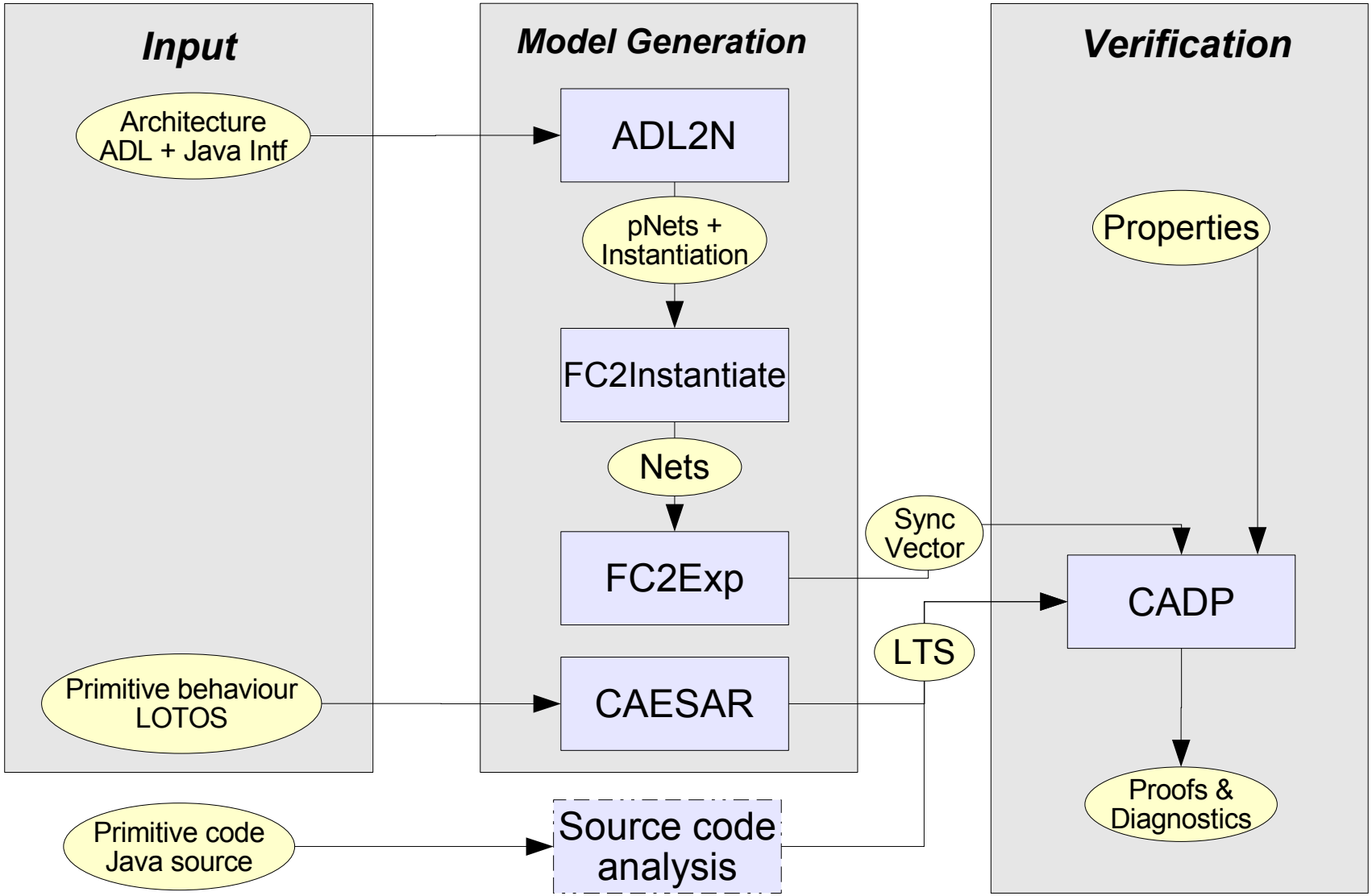
# Case Study


User(ip)
IAccess
Firewall
IRedirect
Access
Policy
ILogin
IInternet
Web
Server
Internet
Arbitrator
ILogin
IFTAuth
IReconfigure
FlyTicket
Classifier
IDBQuery
FrequentFlyer
FreqFlyer
Classifier
IDBQuery
FreqFlyer
Database
IDBQuery
DB 1
DB 2

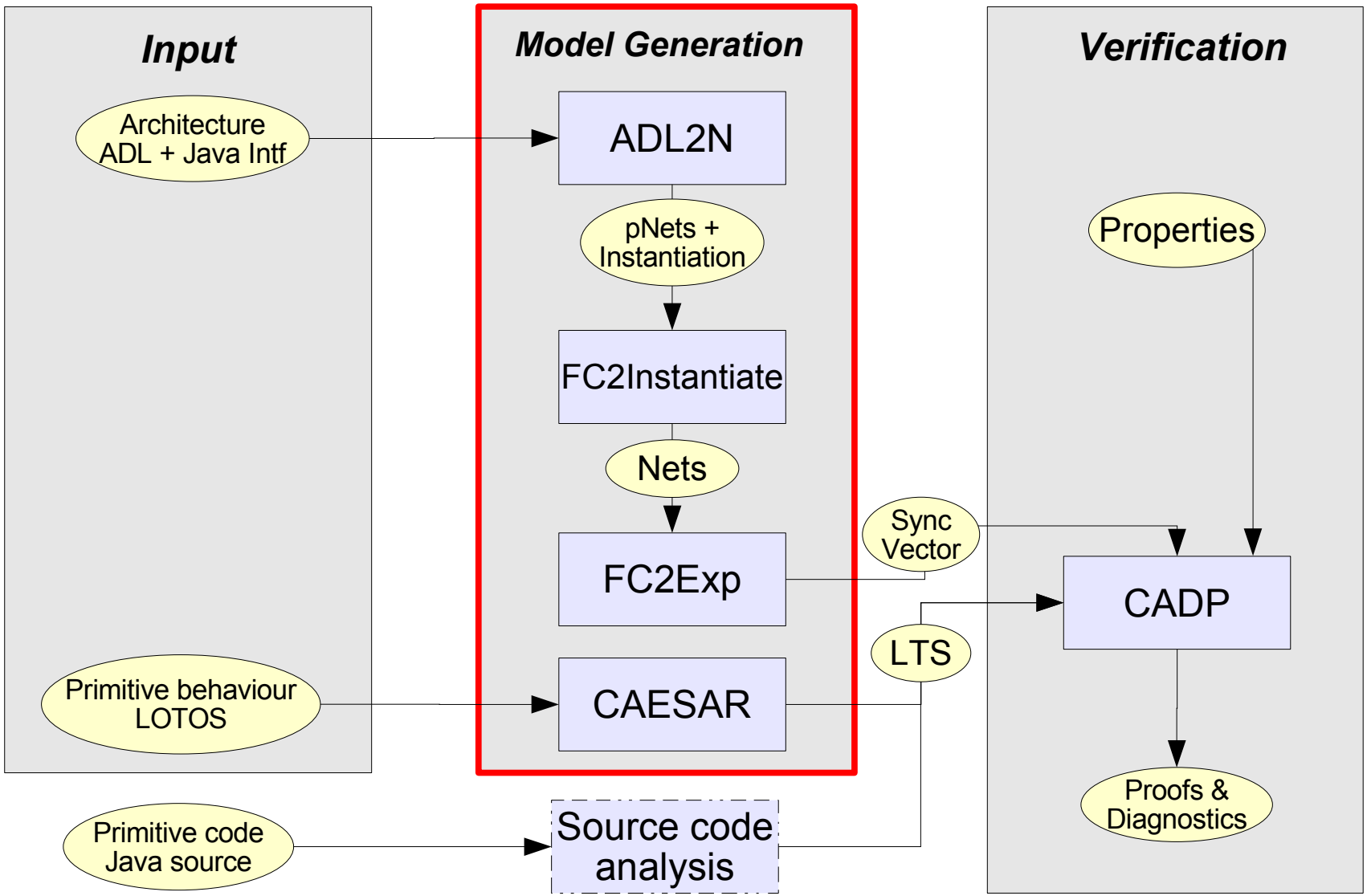
# Agenda

- Introduction
- Theoretical outline
- Case Study
- Platform overview
- Work in progress and Future Work
- Conclusion

# Vercors Platform

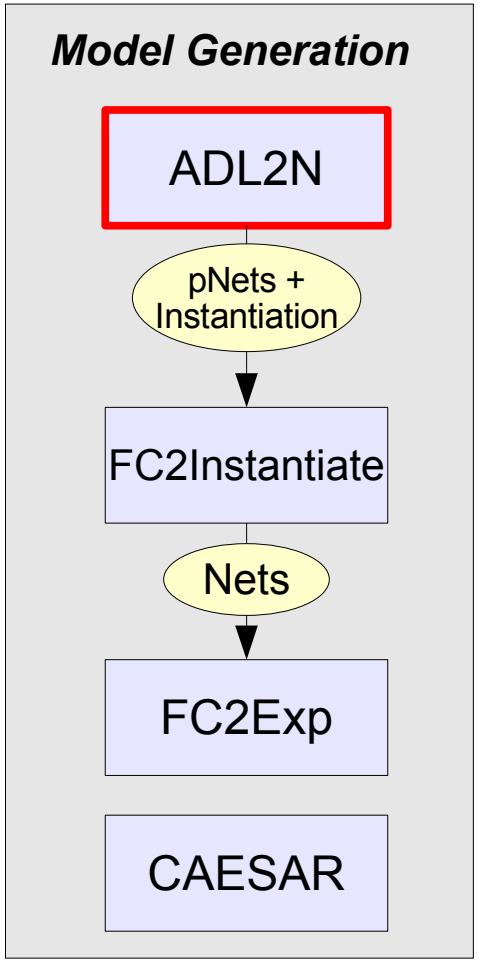


# Vercors Platform

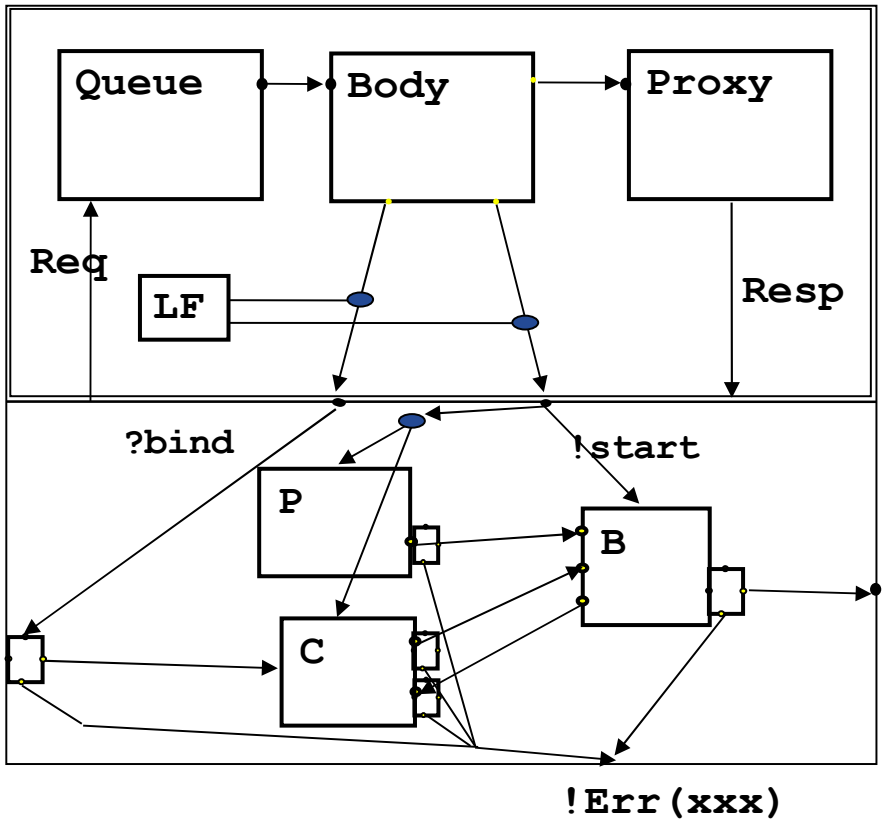


# Model creation: ADL2N

- Inputs
  - ADL (including Java interfaces)
  - Abstraction
  - Observable actions
- Outputs
  - Parameterized FC2 with synchronisations
  - Instantiation file

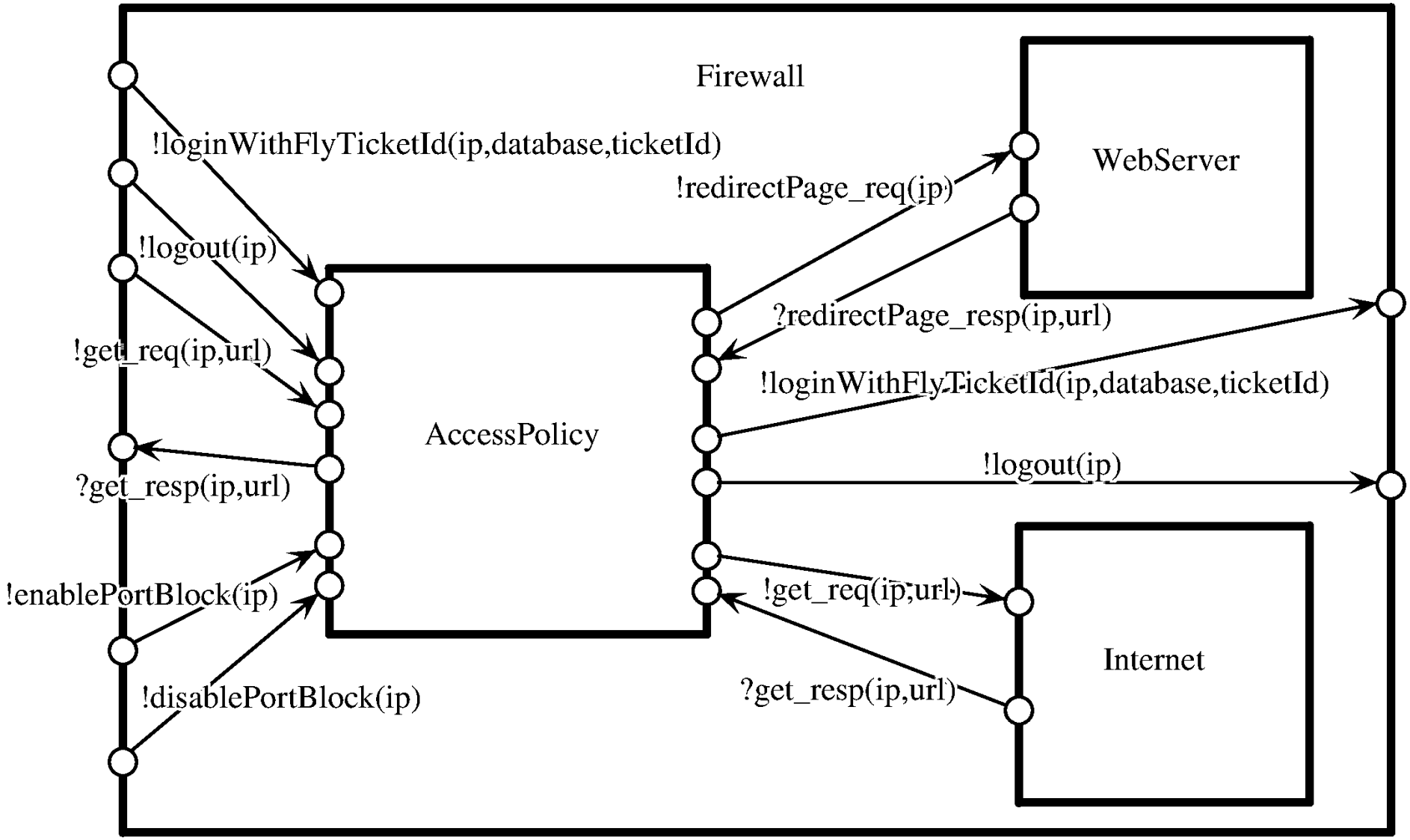


# ADL2N

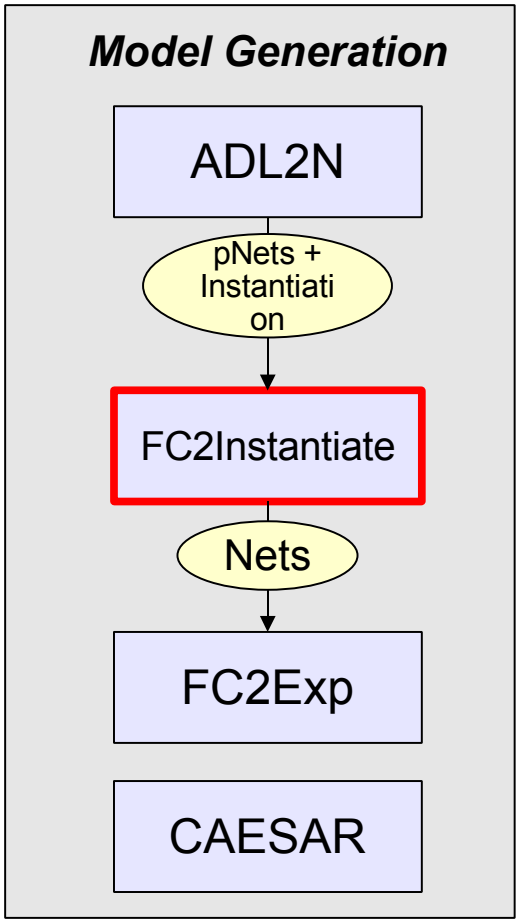

Selection of non functional features, asynchronous mechanisms, errors, etc

# pNet obtained by ADL analysis



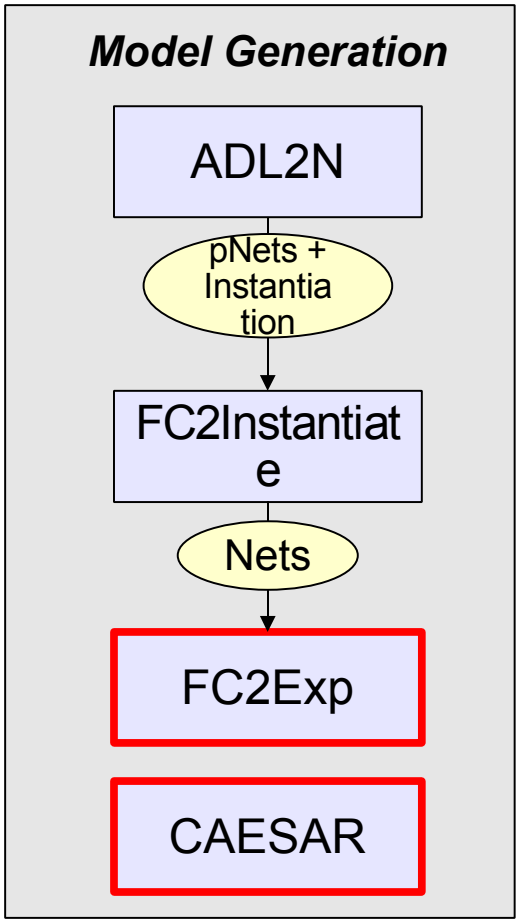
# Finite model

- Instantiate pNets towards a finite LTS, using the user-given abstraction in ADL2N
  - Automatic tool: FC2Instantiate
  - Generates a finite system of communicating automata



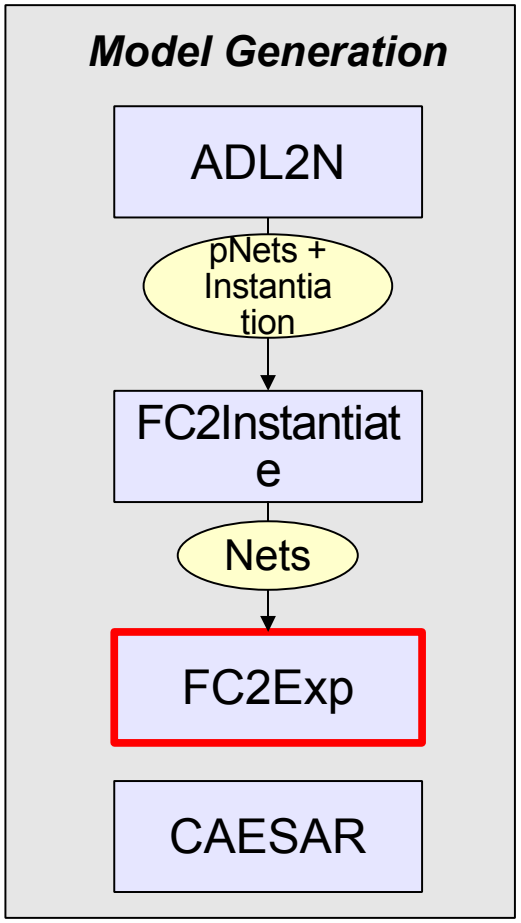
# Interoperability with Verification Tools

- Model checker: *CADP*
  - Based on process algebras
  - LOTOS compiler
  - On-the-fly capabilities
  - Distributed space-generation



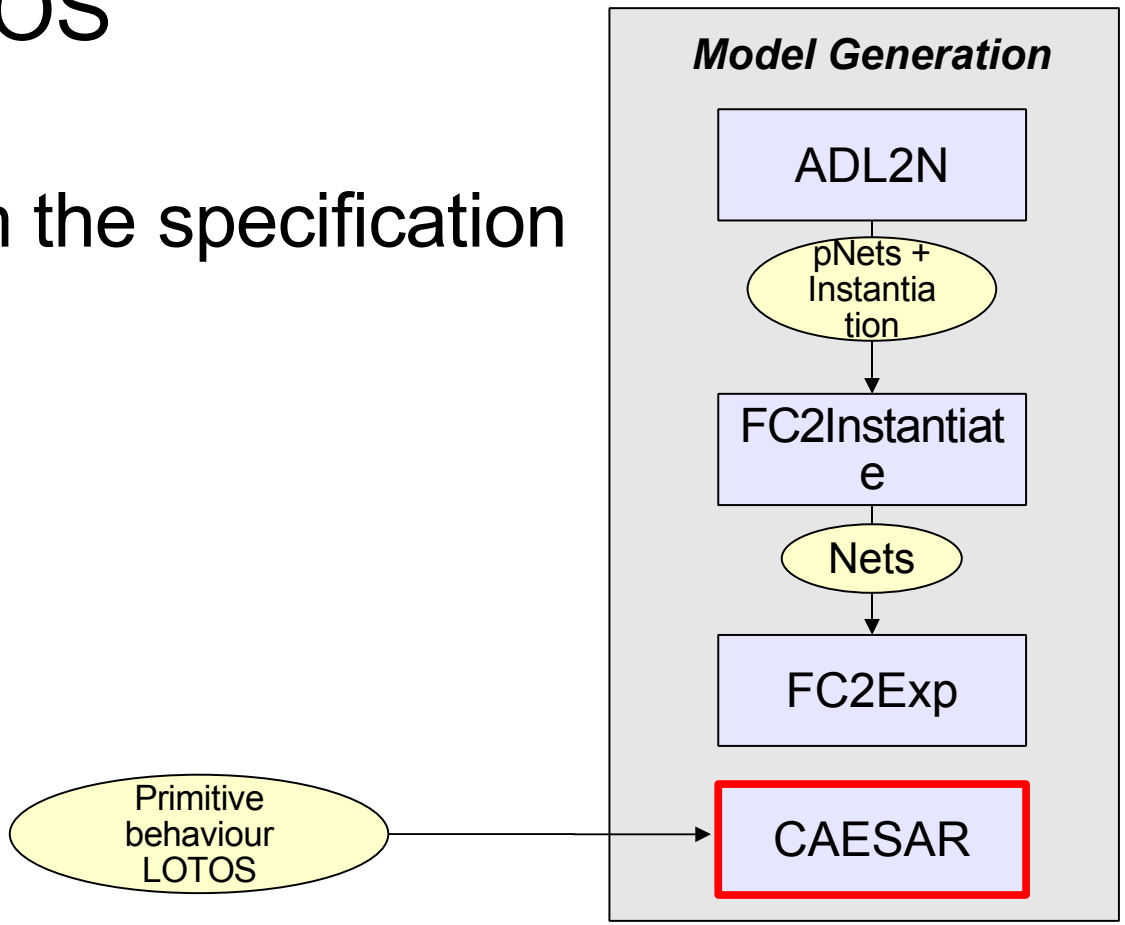
# Interoperability with Verification Tools

- FC2 format is translated into EXP and BCG
  - Automatic tool: FC2EXP



# Interoperability with Verification Tools

- Compilation of LOTOS specifications
  - Build finite LTS from the specification



# Functional specification

```
specification INTERNET [IINTERNET] : noexit

library GENERAL,IINTERNET endlib

behaviour INTERNETBODY[IINTERNET] where

process METHOD_GET[IINTERNET](m:MethodGet) : exit :=
    IINTERNET !getIP(m) !getURL(m);
    exit
endproc

process INTERNETBODY[IINTERNET]: noexit :=
    choice IP:IP []
    choice URL:URL []
        (
        IINTERNET !C(get(IP,URL)) of IInternet;
        METHOD_GET[IINTERNET](get(IP,URL))
        )
    >> INTERNETBODY[IINTERNET]
endproc

endspec
```

# Functional specification

```
specification INTERNET [IINTERNET] : noexit

library GENERAL,IINTERNET endlib

behaviour INTERNETBODY[IINTERNET] where

process METHOD_GET[IINTERNET](m:MethodGet) : exit :=
    IINTERNET !getIP(m) !getURL(m);
    exit
endproc

process INTERNETBODY[IINTERNET]: noexit :=
    choice IP:IP []
    choice URL:URL []
        (
        IINTERNET !C(get(IP,URL)) of IInternet;
        METHOD_GET[IINTERNET](get(IP,URL))
        )
    >> INTERNETBODY[IINTERNET]
endproc

endspec
```

runActivity()

# Functional specification

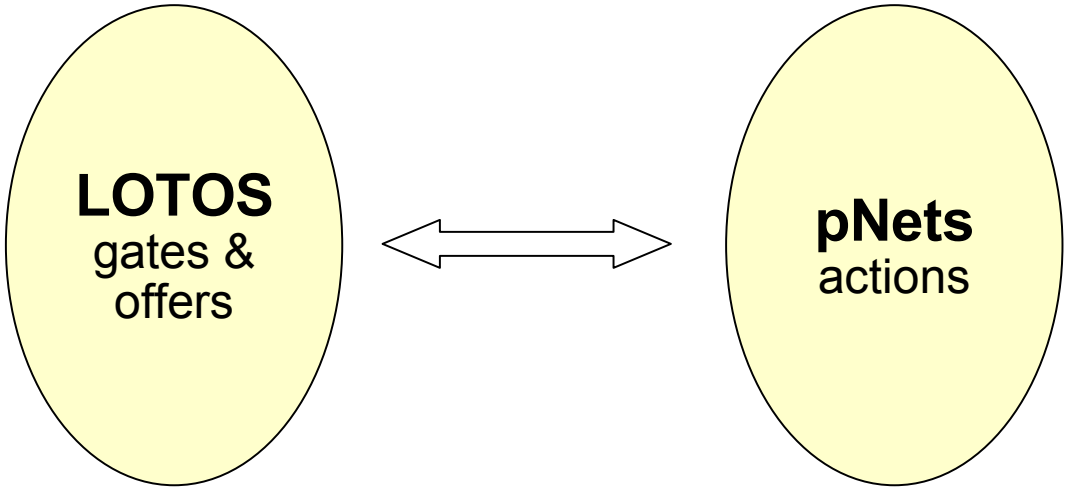
Method specification

runActivity()

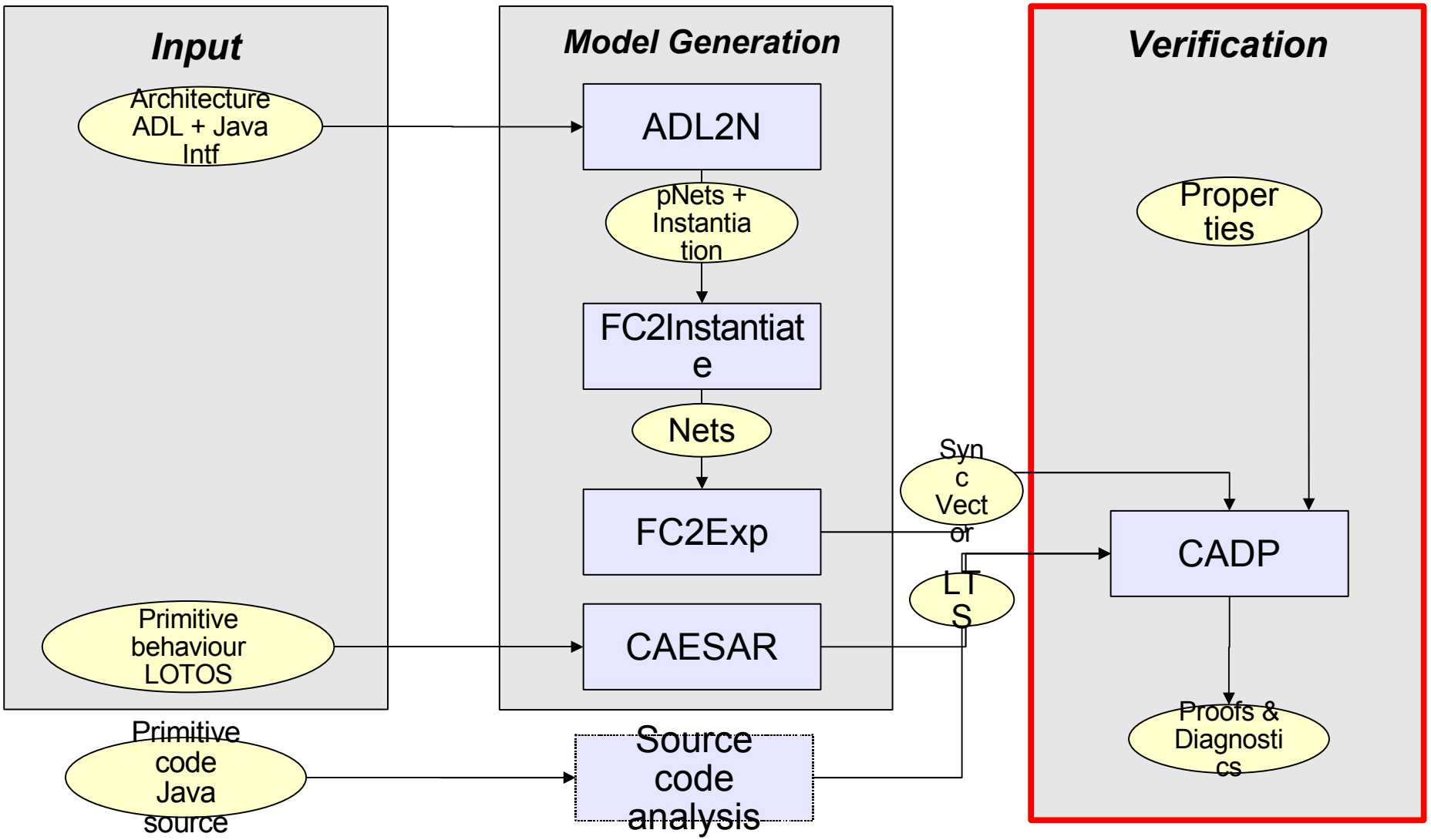
```
specification INTERNET [IINTERNET] : noexit

library GENERAL,IINTERNET endlib

behaviour INTERNETBODY[IINTERNET] where

process METHOD_GET[IINTERNET](m:MethodGet) : exit :=
    IINTERNET !getIP(m) !getURL(m);
    exit
endproc

process INTERNETBODY[IINTERNET]: noexit :=
    choice IP:IP []
    choice URL:URL []
        (
        IINTERNET !C(get(IP,URL)) of IInternet;
        METHOD_GET[IINTERNET](get(IP,URL))
        )
    >> INTERNETBODY[IINTERNET]
endproc

endspec
```

# Mapping to pNets

```
rename
IINTERNET !C (GET (IP (\([0-9]*\)), URL (\([0-9]*\)))) ->
   ?get_req(\1,\2)
IINTERNET !IP (\([0-9]*\)) !URL (\([0-9]*\)) ->
   !get_resp(\1,\2)
```



# Vercors Platform



# Verification

- Press-button verification
  - Finding Deadlocks
  - Absence of usual errors
- Logical Languages
  - Regular μ-calculus

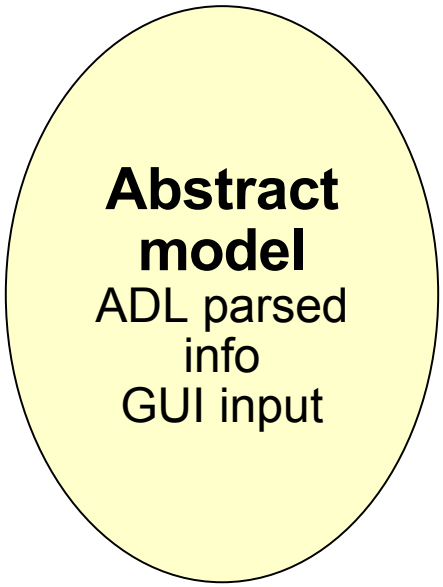
Restrict user scenario encoded by the behaviour specification

# Agenda

- Introduction
- Theoretical outline
- Case Study
- Platform overview
- Work in progress and Future Work
- Conclusion

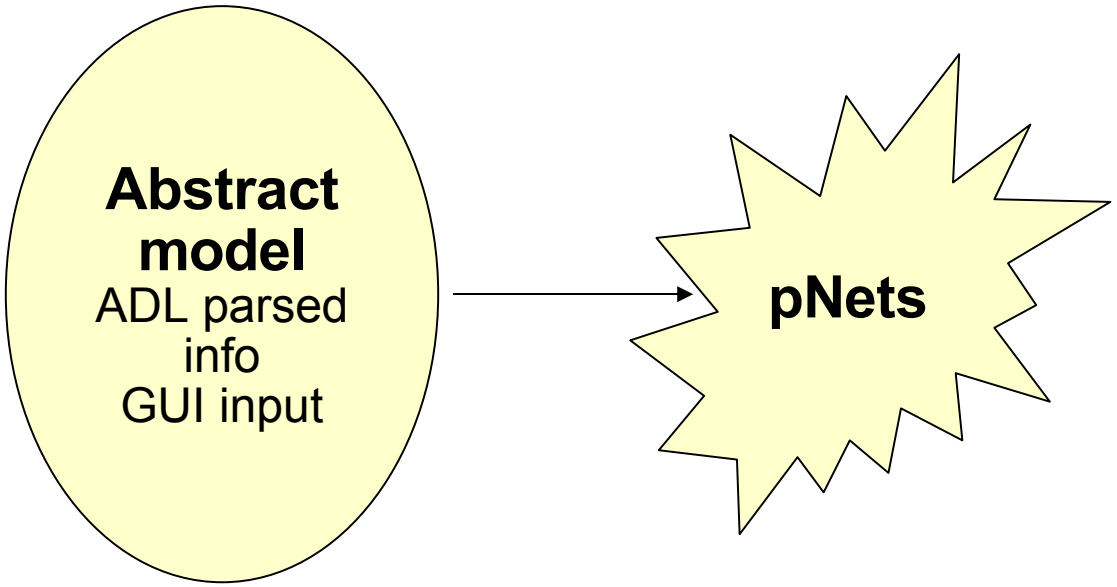
# ADL2N v1

- 2 phase compilation



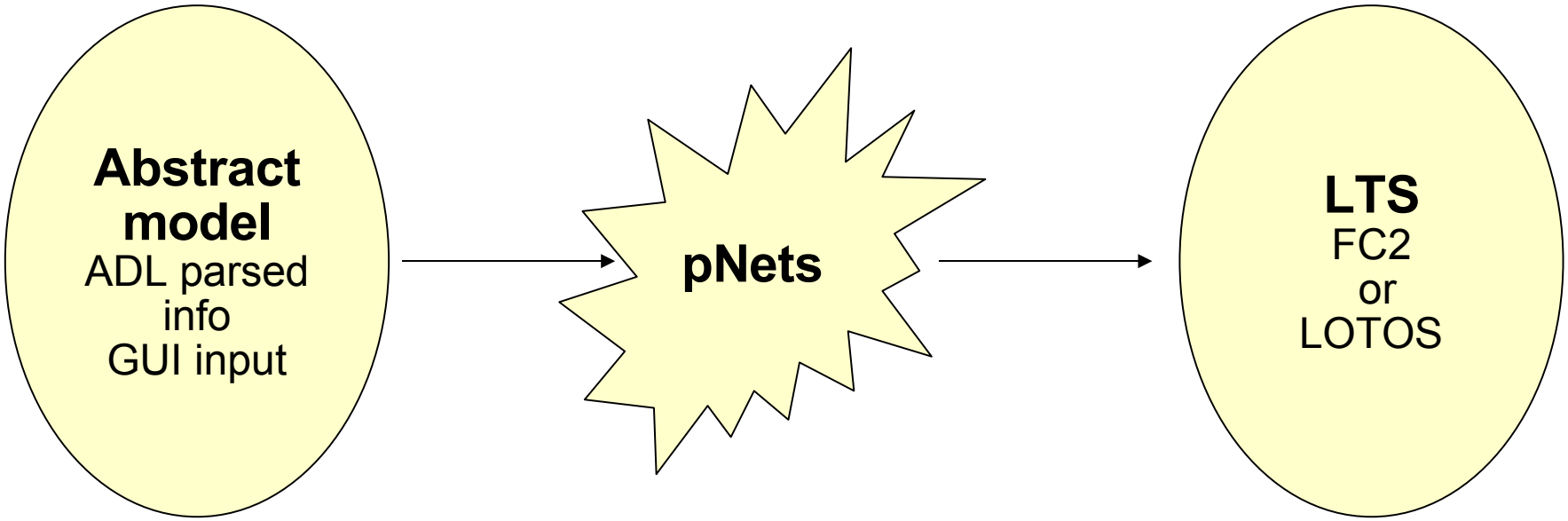
# ADL2N v1

- 2 phase compilation



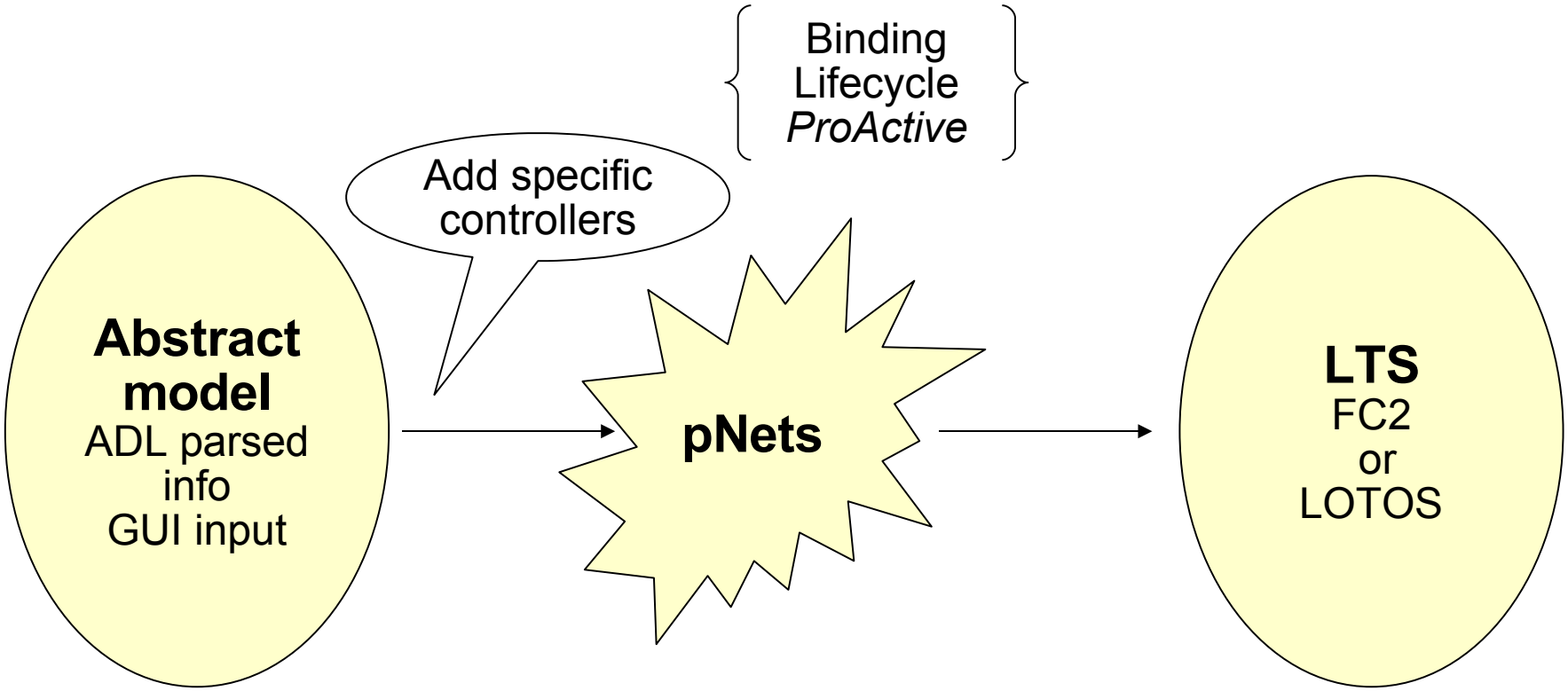
# ADL2N v1

- 2 phase compilation



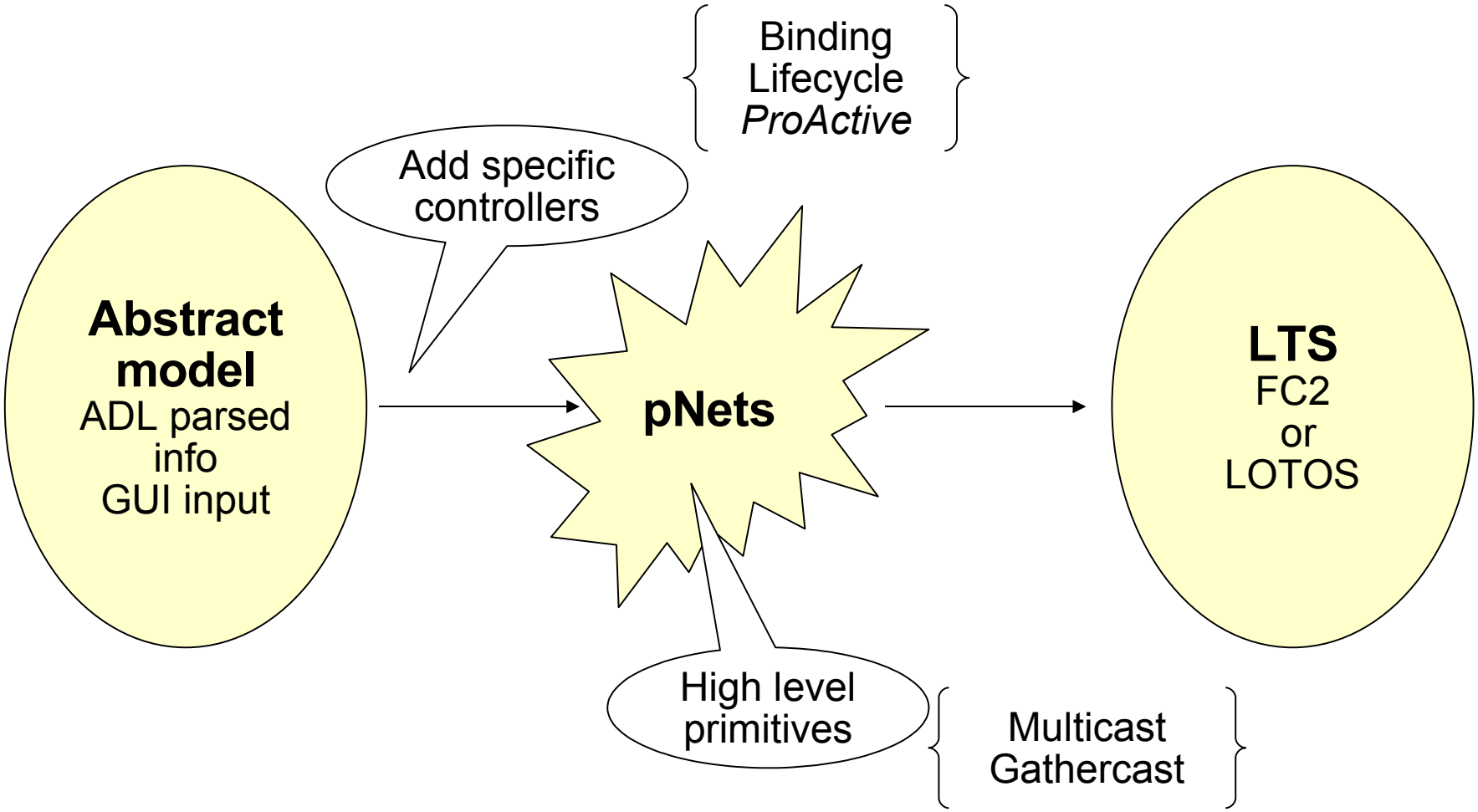
# ADL2N v1

- 2 phase compilation



# ADL2N v1

- 2 phase compilation



# Conclusion

- Current state of tools
  - Functional behaviour
    - Ready and available (ADL2N)
  - Interoperability with CADP
    - Translation from hierarchical FC2 to EXP/BCG is ready and available through FC2EXP
  - NF controllers and ProActive's semantics
    - To be released in ADL2N v1
  - FC2Instantiate
    - Available, but still needs some debugging aid

# Conclusion

- Current state of model
  - Functional and Non-functional distributed components
  - Extensions still needed
    - Exception handling is mandatory
    - Collective interfaces
    - Other features of ProActive

# Future work

- Specification Language for ProActive community