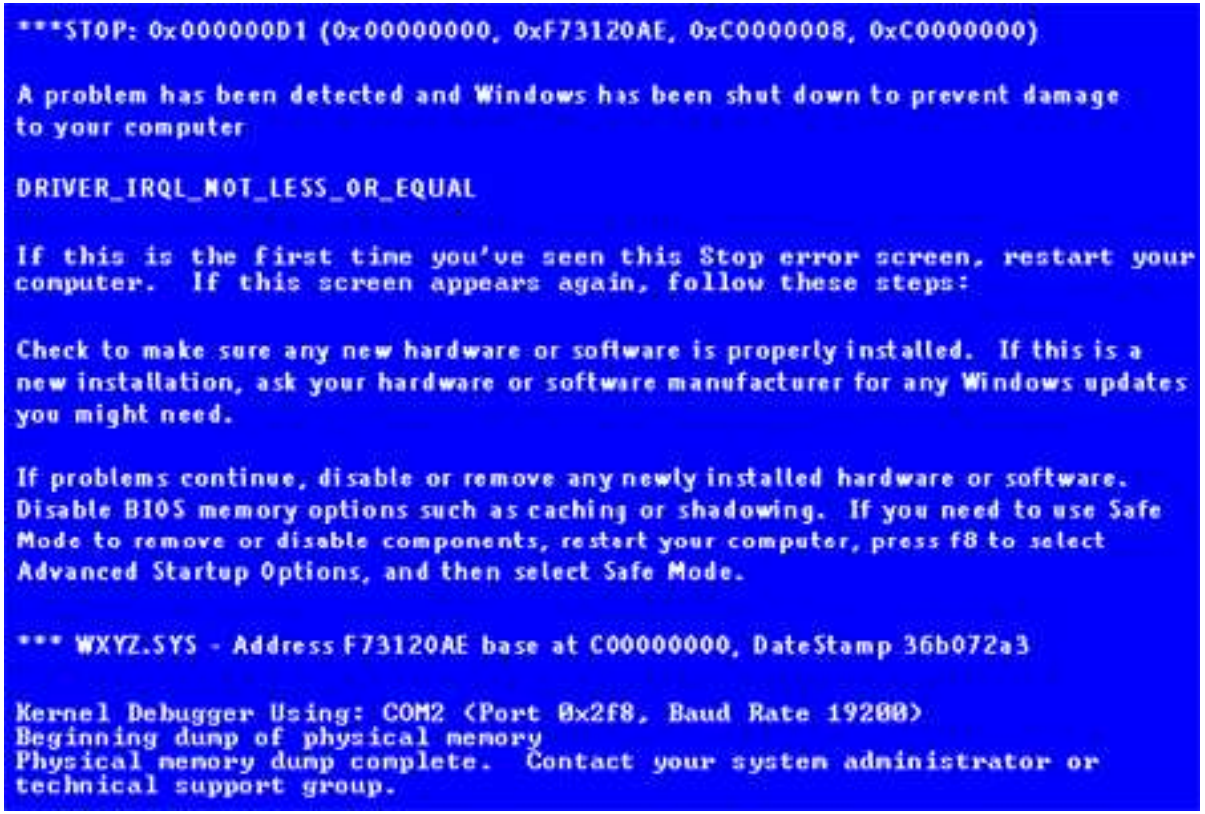
Formal Specification and Verification of Distributed Component Systems

Tomás Barros

Tuesday, October 25th 2005

The problem


***STOP: 0x000000D1 (0x00000000, 0xF73120AE, 0xC0000008, 0xC0000000)
A problem has been detected and Windows has been shut down to prevent damage to your computer
DRIVER_IRQL_NOT_LESS_OR_EQUAL
If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:
Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press f8 to select Advanced Startup Options, and then select Safe Mode.
*** WXYZ.SYS - Address F73120AE base at C00000000, DateStamp 36b072a3
Kernel Debugger Using: COM2 (Port 0x2f8, Baud Rate 19200)
Beginning dump of physical memory
Physical memory dump complete. Contact your system administrator or technical support group.

The facts

"People who write software are human first and programmers only second - in short, they make mistakes, lots of them...

Software is everywhere, not just in computers but in household appliances, cars, aeroplanes, lifts, telephones, toys and countless other pieces of machinery. In a society dependent on software, the consequences of programming errors ("bugs") are becoming increasingly significant..."

Building a better bug-trap. The Economist, June 19th 2003

The need for reliable systems

- June 4, 1996. Ariane 5 rocket exploded 40 seconds after its lift-off.
  - $500 million
  - The error: A casting from a 64 bit float to a 16 bit integer
- Pentium-II floating-point division bug.
  - $475 million

The need for reliable systems

- February 25, 1991, Gulf War. A Patriot Missile failed to intercept an incoming Iraqi Scud missile.
  - 28 american soldier killed
  - The error: Not enough significant decimals
- Between 1985 and 1987, a software flaw in the Therac-25 radiation therapy machine
  - 6 patient deaths and several injured

The solutions

- Development Techniques & standards
  - CMMI, ISO standards, Object patterns
- Testing & Debug
  - Bug detection
- Formal Methods
  - Bug absence

Formal Methods

- Describe software using a sound mathematical basis
- It provides precise notions to model, develop and reason about computer systems. It reveals ambiguity, incompleteness and inconsistency
- Roughly speaking, there are two approaches:
  - Inductive methods: *system specification* (in some logic) $\Rightarrow$ *desired property* (theorem), theorem proving.
  - Model-based methods: *system specification* (behavioural model) $\models$ *desired property* (temporal logic), model-checking.

Focus: Distributed Systems

- Concurrent processes
  - Spawn on several machines
- No shared memory or global time
  - Difficult to identify the system states
- Asynchronous communications

best suitable formalism?

Process Algebras

Observe the actions that a process may execute.

Process is an expression with operators, ex (CCS): $P = \alpha.P + E$

Semantics are given through operational rules, ex (CCS)::

$$\frac{}{\alpha.E \xrightarrow{\alpha} E} \qquad\qquad \frac{E_0 \xrightarrow{\alpha} E_0'}{E_0|E_1 \xrightarrow{\alpha} E_o'|E_1}$$

Defines an equivalence relation (bisimulation)

$\Rightarrow$ reduction & composition $\Rightarrow$ Scalability

Description Language[1]

- Parameterized Networks (**pNets**) of Communicating Automata (**pLTS**)
  - Guarded actions with parameters
  - Parameters encoding value passing and process references
  - Parameters encoding a family of processes
  - Parameterized synchronisation vectors

$$G \leftarrow < *, *, \alpha_3(k_3), *, \alpha_4(k_4), * >$$

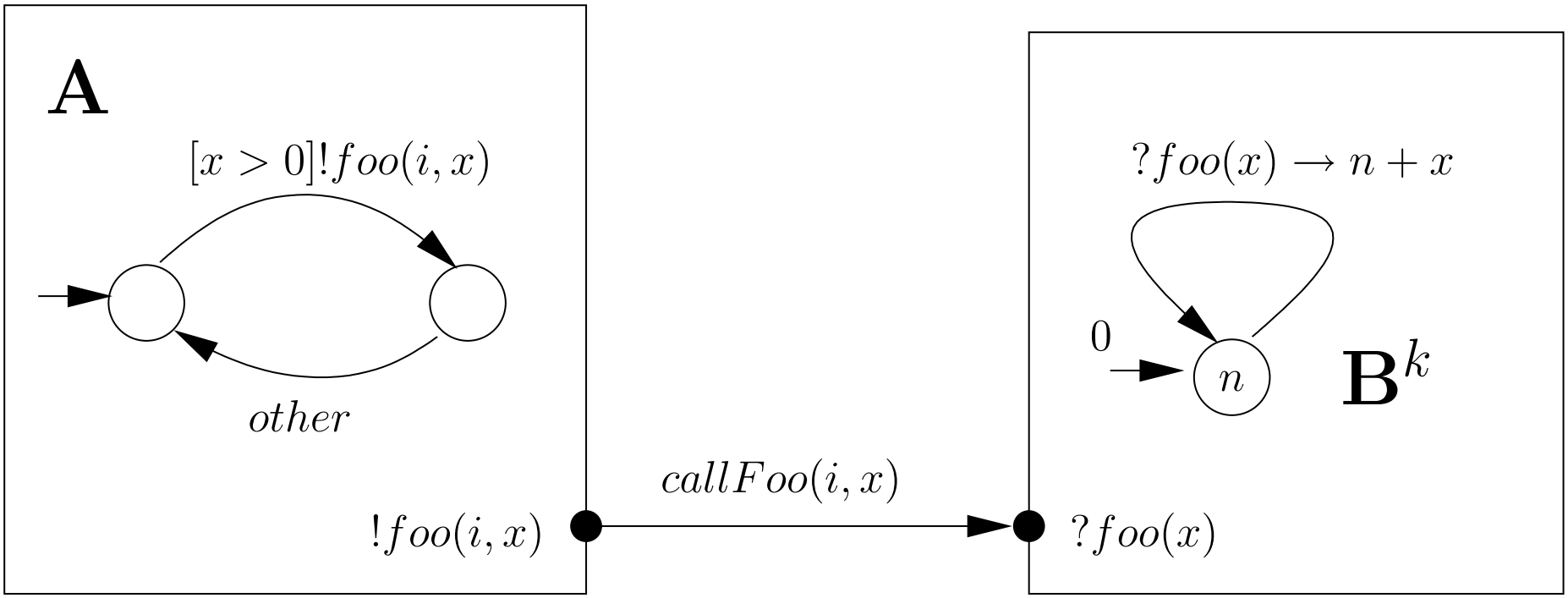
- Instantiation: for a finite abstraction of the parameters domain D_v:

$$\left.\begin{array}{l} pLTS \times D_v \\ pNet \times D_v \end{array}\right. \xrightarrow{\text{Instantiation}} \left.\begin{array}{c} LTS \\ Net \end{array}\right\} \text{[2]Global Behaviour (LTS)}$$

[1] Work published in three international conferences [2, 1], [6] (doctoral symposium) and a research report [7]

[2] Synchronisation Product

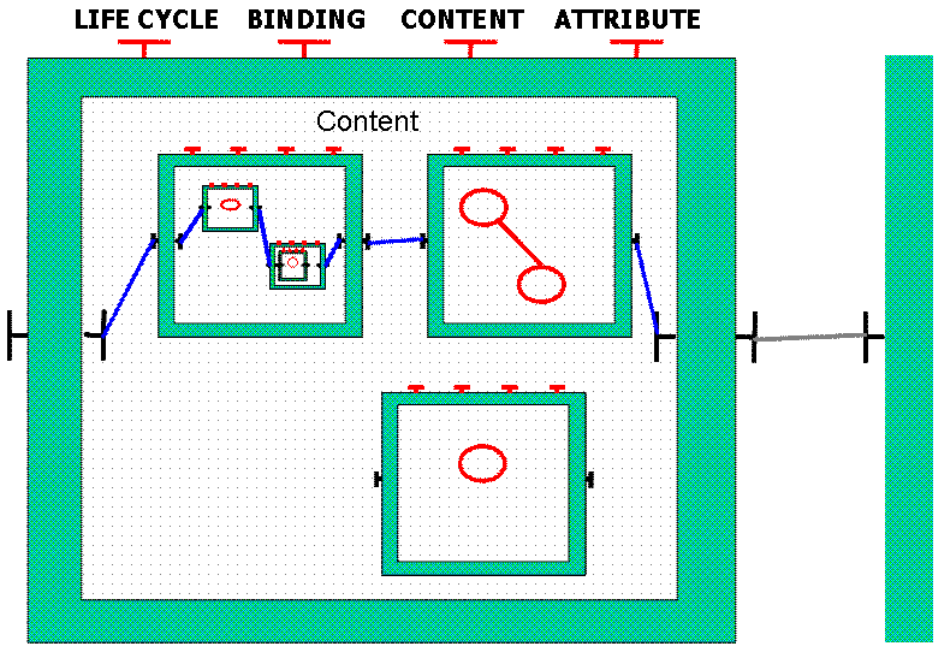
Description Language[3]: graphical view



[3]Suitable for both system specification and as target language for data source analysis

Fractal components

(dealing with complex systems)



- Encapsulation
  - Back-box view, functional & management
- Composition
  - Hierarchical sub-systems
- Description
  - ADL, deployment

Correct behaviour

The challenge is to build a formal framework which ensures that applications built from existing components are **safe**, in the sense that their parts fit together appropriately and behave together smoothly.

Each component must be adequate for its assigned role within the system, and the update or replacement of a component should not cause the rest of the system deadlock or fail.

Existing approaches

- Wright
  - Connectors specified using CSP
  - Compatibility relation (modify CSP refinement)
- Sofa
  - Frame (spec) vs. Architecture (implementation) compliance relation based on traces
  - Hierarchical construction through parallel composition detection of errors: bad activity, no activity and divergence

Existing approaches assumption

> The system has been correctly deployed in its initial construction, in particular all the necessary paths between functional interfaces have been established and the system has been started in a coherent state.

However, components programing allows, through the non-functional capabilities, to control the execution of a component and its dynamic evolution: plugging and unplugging components dynamically provides adaptability and maintenance. Therefore this **inter-play between functional and non-functional aspects influences the behaviour of the system**, even if we look at the components from a pure functional point of view.

Our Approach

(focused on distributed hierarchical components)

We automatically build the system behavioural model including these functional $\leftrightarrow$ non-functional inter-play[4]

- Functional behaviour of primitive components is known
  - Explicit by the user
  - Static Analysis
- Non functional behaviour and asynchronous communications are automatically incorporate from system description (ADL)

[4] Work published in two international conferences [3, 4] and a research report [5]

User's input samples

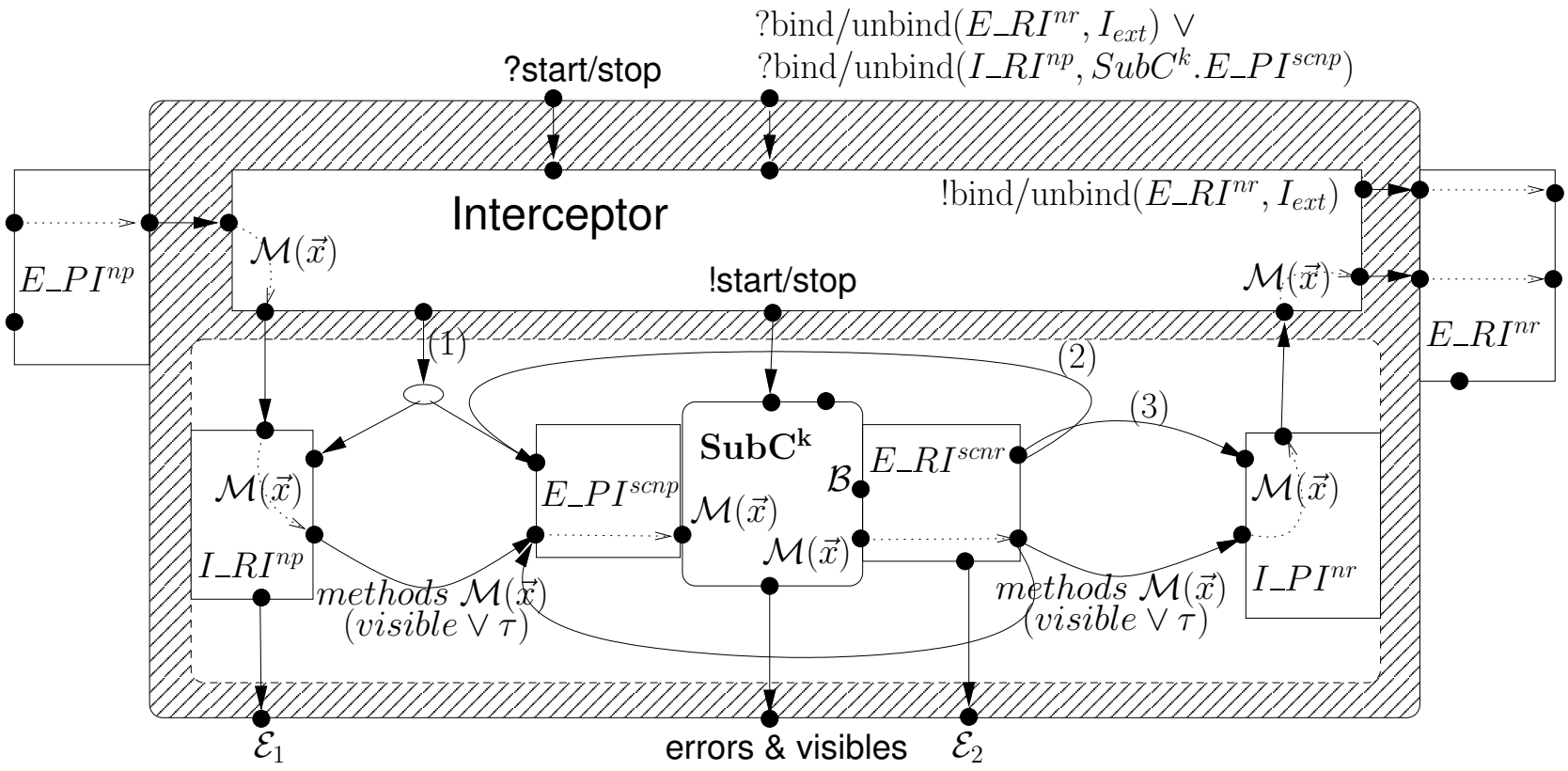
System.fractal

```
<?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">
      ...
      <content class="components.Alarm">
         <behaviour file="AlarmBehav"
                format="FC2Param"/>
      </content>
   </component>

   <binding client="BufferSystem.alarm"
        server="Alarm.alarm"/>
</definition>
```

Buffer.lotos

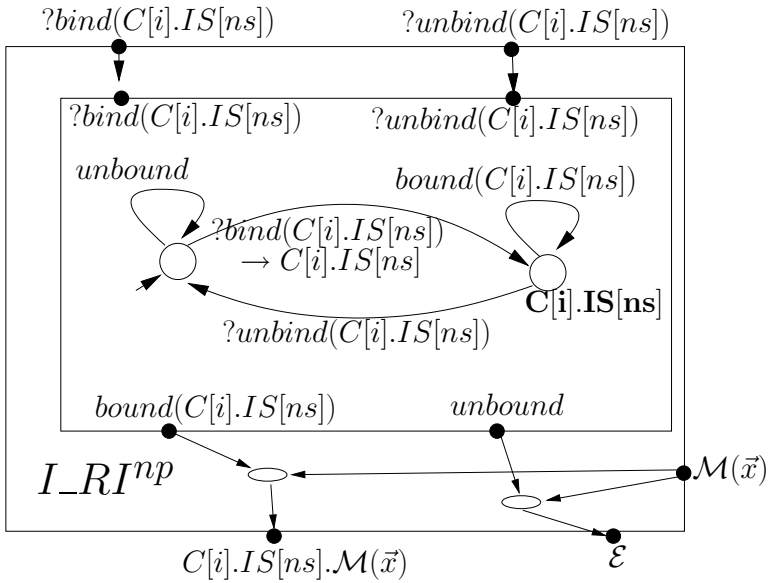
```
process BUFFER[NOACTIVE, SERVE_GET, GET_REP,
    PUT, ALARM](stock:Nat, bound:Nat): exit :=

  PUT ?X:Nat [X <= (bound - stock)];
     BUFFER[NOACTIVE, SERVE_GET, GET_REP,
        PUT, ALARM] (stock+X,bound)
  []
  [stock > 0] -> SERVE_GET?C:Cons; GET_REP!C;
     BUFFER[NOACTIVE, SERVE_GET, GET_REP,
        PUT, ALARM](stock-1,bound)
  []
  [stock == bound] -> ALARM;
     BUFFER[NOACTIVE, SERVE_GET, GET_REP,
        PUT, ALARM](stock,bound)
  []
  NOACTIVE; exit
endproc
```

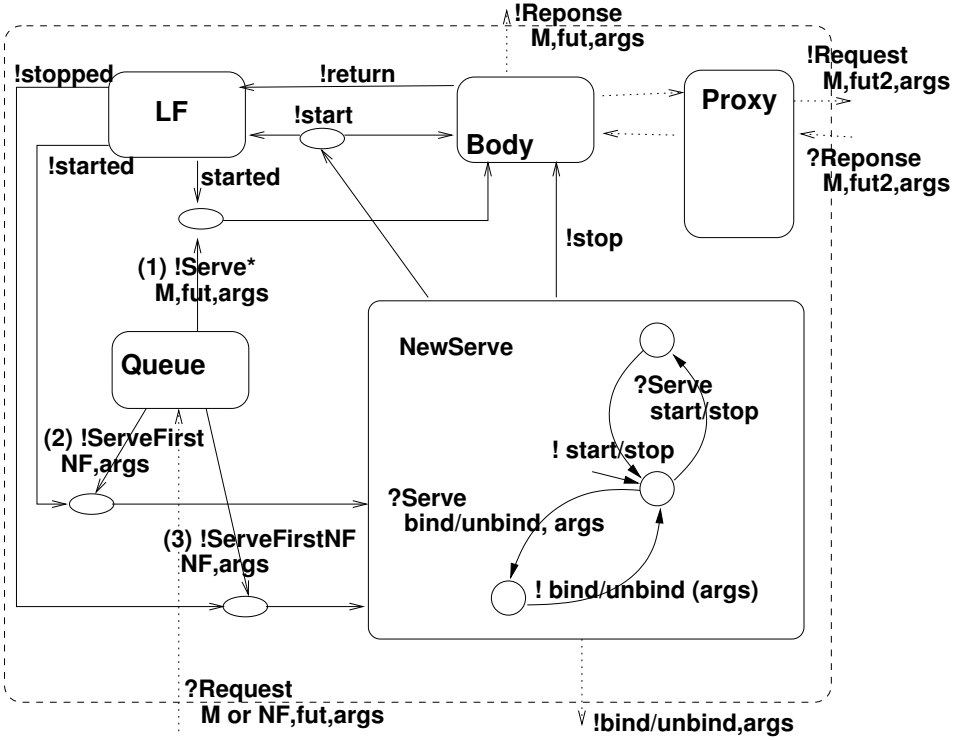
Modelling at any level of the hierarchy



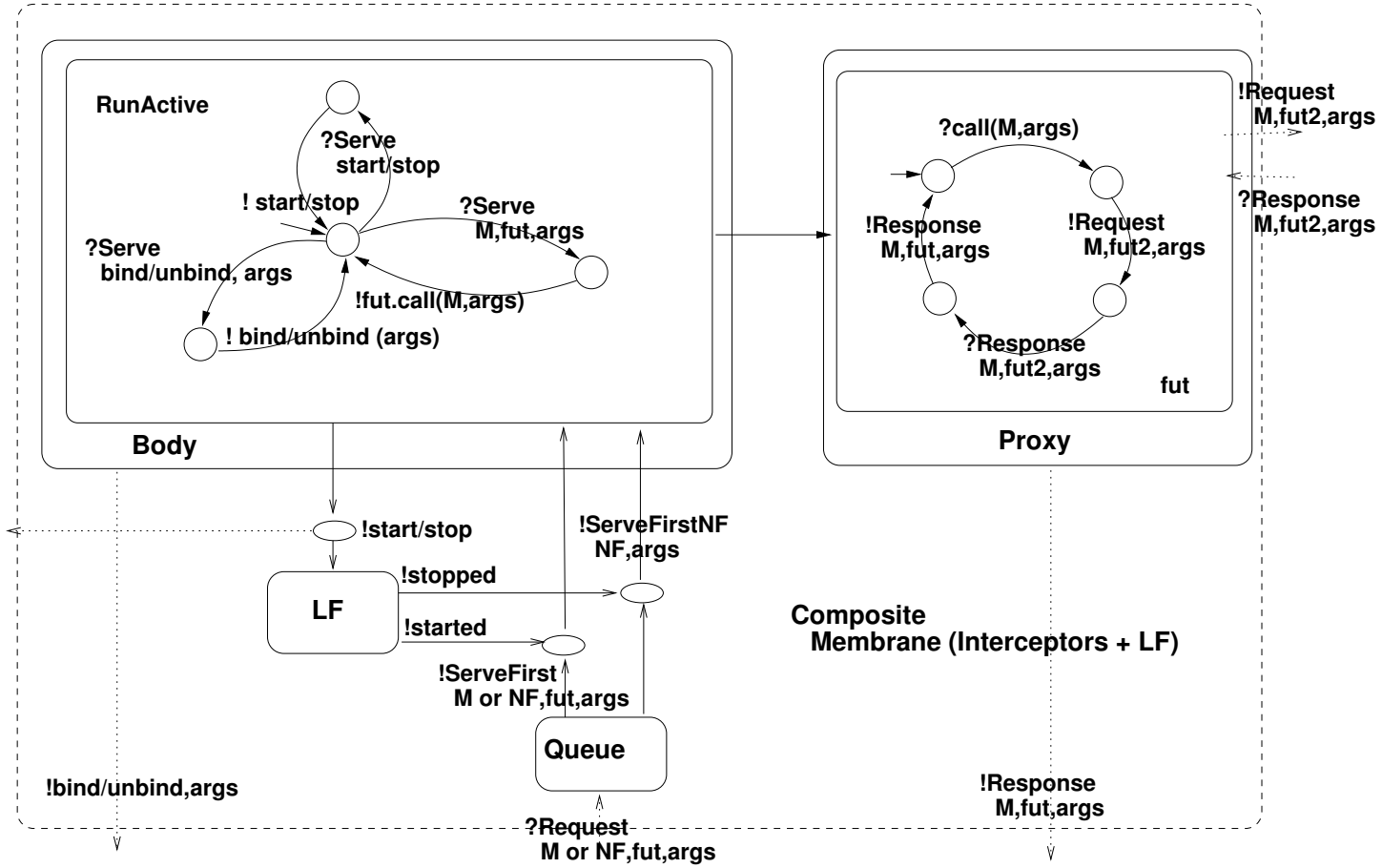
Controller

Interface sample and primitive components





Composites


RunActive
?Serve
start/stop
! start/stop
?Serve
M,fut,args
?Serve
bind/unbind, args
!fut.call(M,args)
! bind/unbind (args)
Body
?call(M,args)
!Request
M,fut2,args
!Response
M,fut,args
!Request
M,fut2,args
?Response
M,fut2,args
?Response
M,fut2,args
fut
Proxy
!Request
M,fut2,args
?Response
M,fut2,args
!start/stop
!ServeFirstNF
NF,args
!stopped
LF
!started
!ServeFirst
M or NF,fut,args
Queue
Composite
Membrane (Interceptors + LF)
!bind/unbind,args
?Request
M or NF,fut,args
!Response
M,fut,args

Deployment

The *Deployment Automaton* is a sequence of management operations.

Static Automaton = (Controller $\|$ Deployment) *(+ hiding & reduction)*

- The static automaton represent the behaviour of the component after deployment
- It encodes the behaviour of this sub-component at the next level of the hierarchy
- It is used to prove temporal properties.

Properties samples

- The deployment is successful and absence of errors (expressed in CTL[5])

$$\mathbf{A}(true_{true}\mathbf{U}_{\surd}\, true) \qquad\qquad \mathbf{AG}_{true}[\mathbf{O}_{E}]\; false$$

- Asynchronous management request are eventually applied (in μ-calculus[6])

$$[true * .!\texttt{start}(\texttt{name})]\mu X.(< true > true \wedge [\neg\texttt{internal\_start}(\texttt{name})]X)$$

[5] propositional logic plus temporal operators **G,X,F** and path operators **A, E**

[6] a fix-point temporal logic

Conclusions

The thesis has lead to

- A new formalism and syntax for describing the behaviour of distributed systems (pNets and pLTSs)
- Several tools to deal with such formalism (FC2Instantiate, FC2EXP, ADL2N (on-going work), PAX)
- An automatic approach for modelling components
- Several illustrations of properties checks

References

[1] I. Attali, T. Barros, and E. Madelaine. Formalisation and proofs of the chilean electronic invoices system. In *XXIV International Conference of the Chilean Computer Science Society (SCCC 2004)*, pages 14–25, Arica, Chili, November 2004. IEEE Computer Society.

[2] T. Barros, R. Boulifa, and E. Madelaine. Parameterized models for distributed java objects. In *Forte'04 conference*, Madrid, 2004. LNCS 3235, Spinger Verlag.

[3] T. Barros, L. Henrio, and E. Madelaine. Behavioural models for hierarchical components. In Patrice Godefroid, editor, *Model Checking Software, 12th International SPIN Workshop*, volume LNCS 3639, pages 154–168, San Francisco, CA, USA, Aug 2005. Springer.

[4] T. Barros, L. Henrio, and E. Madelaine. Verification of distributed hierarchical components. In *International Workshop on Formal Aspects of Component Software (FACS'05)*, Macao, October 2005. Electronic Notes in Theoretical Computer Science (ENTCS).

[5] T. Barros, Henrio Ludovic, and E. Madelaine. Behavioural models for hierarchical components. Technical Report RR-5591, INRIA, June 2005.

[6] T. Barros and E. Madelaine. Formal description and analysis for distributed systems. Technical Report 4-04, University of Kent, Computing Laboratory, April 2004. Doctoral Symposium at IFM'04, Canterbury, Kent, England.

[7] T. Barros and E. Madelaine. Formalisation and proofs of the chilean electronic invoices system. Technical Report RR-5217, INRIA, june 2004.

Future work

- Professor position at Diego Portales University
- Setup collaborations with U. Chile, INRIA Sophia & Grenoble
- Present a Fondecyt project
- Further publications on Journals
- Setup a national research grid and interest group
- and many others in mind. . .

Finally

Some words about OSCAR: students, researchers, workshops, publications, etc.

Thanks