

# A Non-Intrusive Component-Based Approach for Deploying Unanticipated Self-Management Behaviour

**Sandro Santos Andrade**  
**Raimundo José de Araújo Macêdo**  
{sandros, macedo}@ufba.br

**Federal University of Bahia - Brazil**  
Department of Computer Science (DCC)  
Distributed Systems Laboratory (LaSiD)  
Multiinstitutional Doctorate in Computer Science (DMCC)



# Motivation

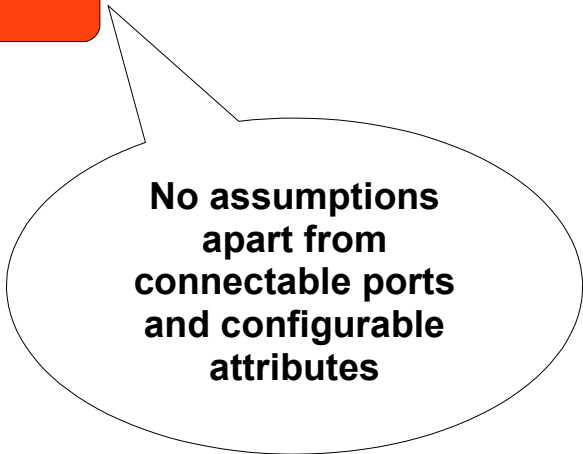
- Architecture-based approaches have proven to be useful for early rationale of non-functional requirements
- Increasing complexity demands the use of fully automatic self-managed software platforms
- Underlying runtime support environment can provide self-management behaviour to already deployed systems (non-intrusiveness)
- Demand for the support of unanticipated changes

# Motivation

- Aspects of dynamic software architectures:
  - Underlying component model
  - Degree of autonomicity
  - Planning and high-level reasoning
  - Runtime support environment
  - Consistency and system integrity

# Motivation

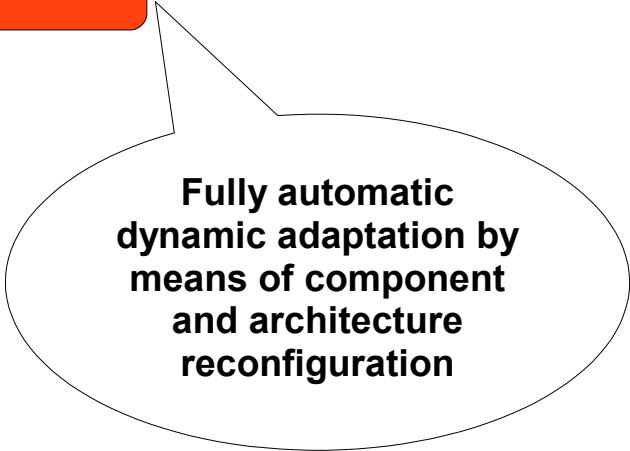
- Aspects of dynamic software architectures:
  - Underlying component model
    - Degree of autonomicity
    - Planning and high-level reasoning
    - Runtime support environment
    - Consistency and system integrity



**No assumptions  
apart from  
connectable ports  
and configurable  
attributes**

# Motivation

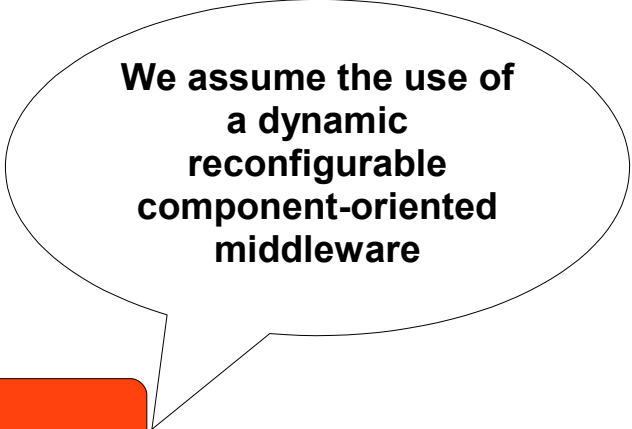
- Aspects of dynamic software architectures:
  - Underlying component model
  - Degree of autonomicity
  - Planning and high-level reasoning
  - Runtime support environment
  - Consistency and system integrity



**Fully automatic  
dynamic adaptation by  
means of component  
and architecture  
reconfiguration**

# Motivation

- Aspects of dynamic software architectures:
  - Underlying component model
  - Degree of autonomicity
  - Planning and high-level reasoning
  - **Runtime support environment**
  - Consistency and system integrity



We assume the use of  
a dynamic  
reconfigurable  
component-oriented  
middleware

# Motivation

- Aspects of dynamic software architectures:
  - Underlying component model
  - Degree of autonomicity
  - Planning and high-level reasoning
  - Runtime support environment
  - Consistency and system integrity



We rely on basic  
middleware  
guarantees

# Motivation

- Aspects of dynamic software architectures:
  - Underlying component model
  - Degree of autonomicity
  - **Planning and high-level reasoning**
  - Runtime support environment
  - Consistency and system integrity



Not addressed  
at all

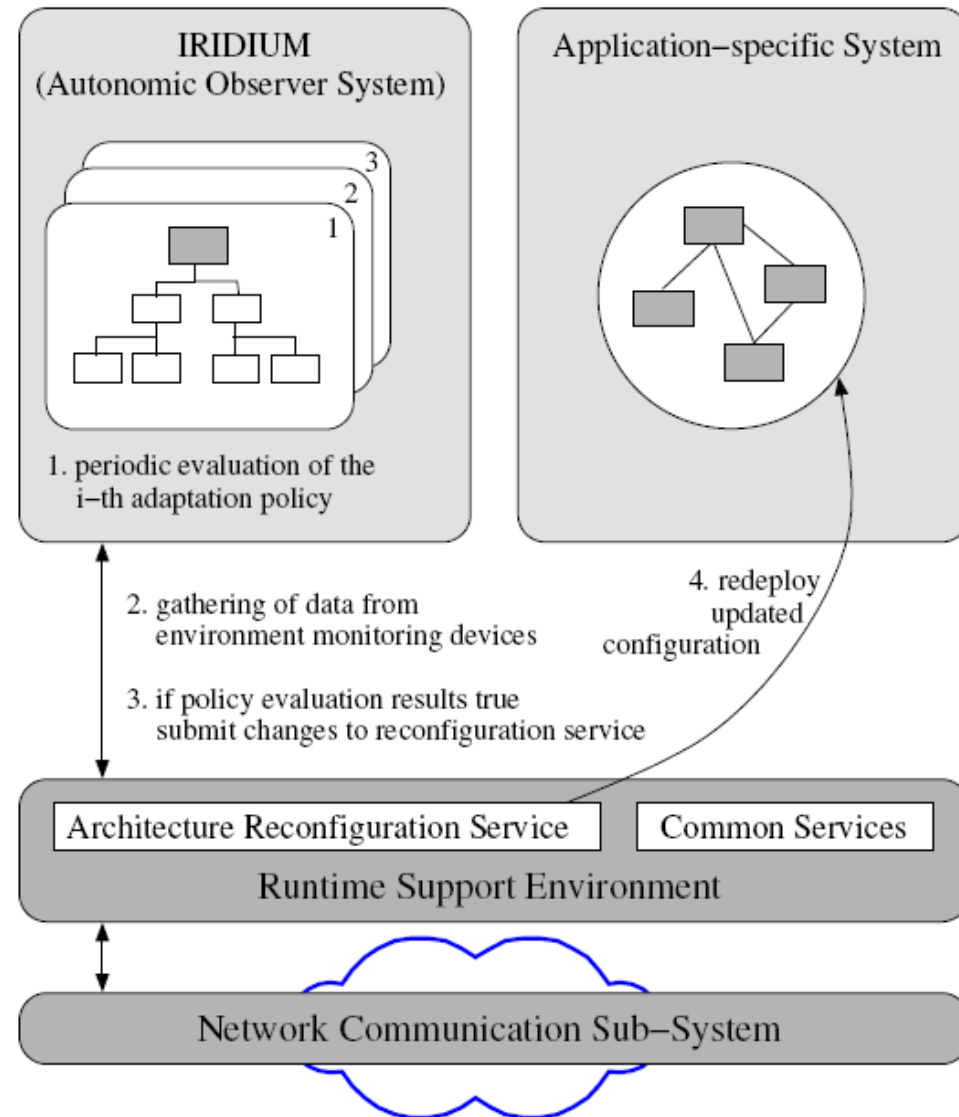


# The Proposed Framework

- Our goal:
  - To provide self-management behaviour in already deployed component-based systems, in a non-intrusive approach
- Characteristics:
  - Reusable core of component-based services for autonomic control loops
  - Building blocks for definition of adaptation policies and architectural changes
  - Hooks (hot-spots) for the connection of developer-supplied components for environment monitoring
  - The adaptation model itself is designed and deployed as a component configuration, which in turn can undergo adaptation

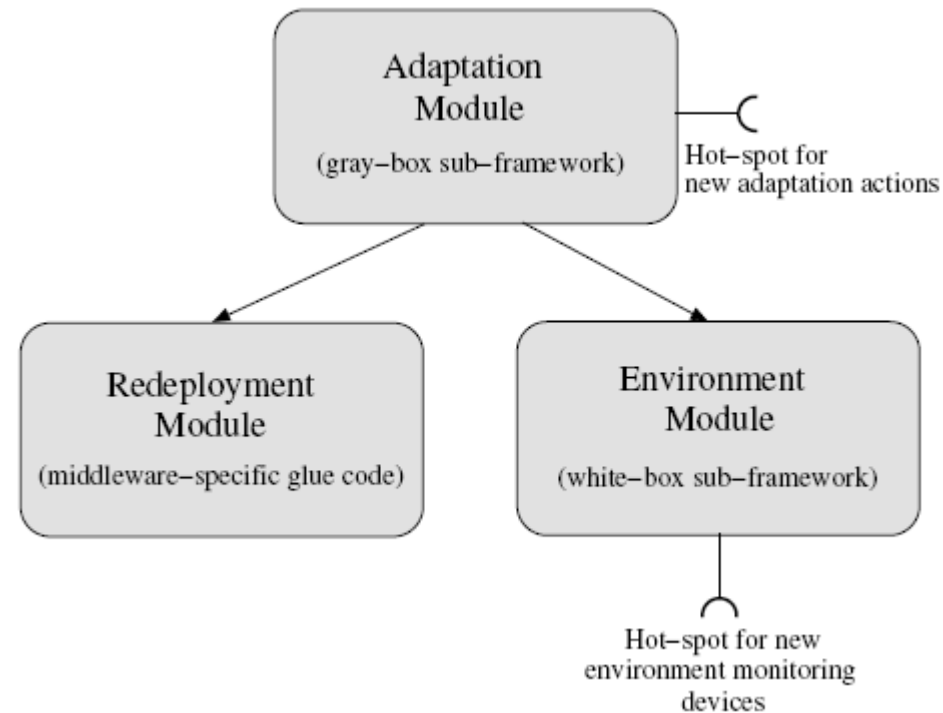
# The Proposed Framework

- Our approach:



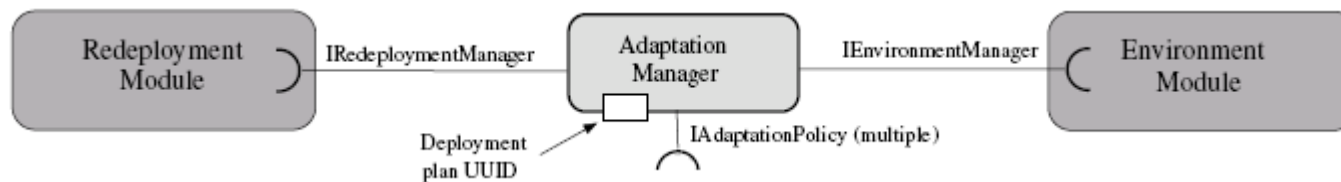
# The Proposed Framework

- Framework architecture:



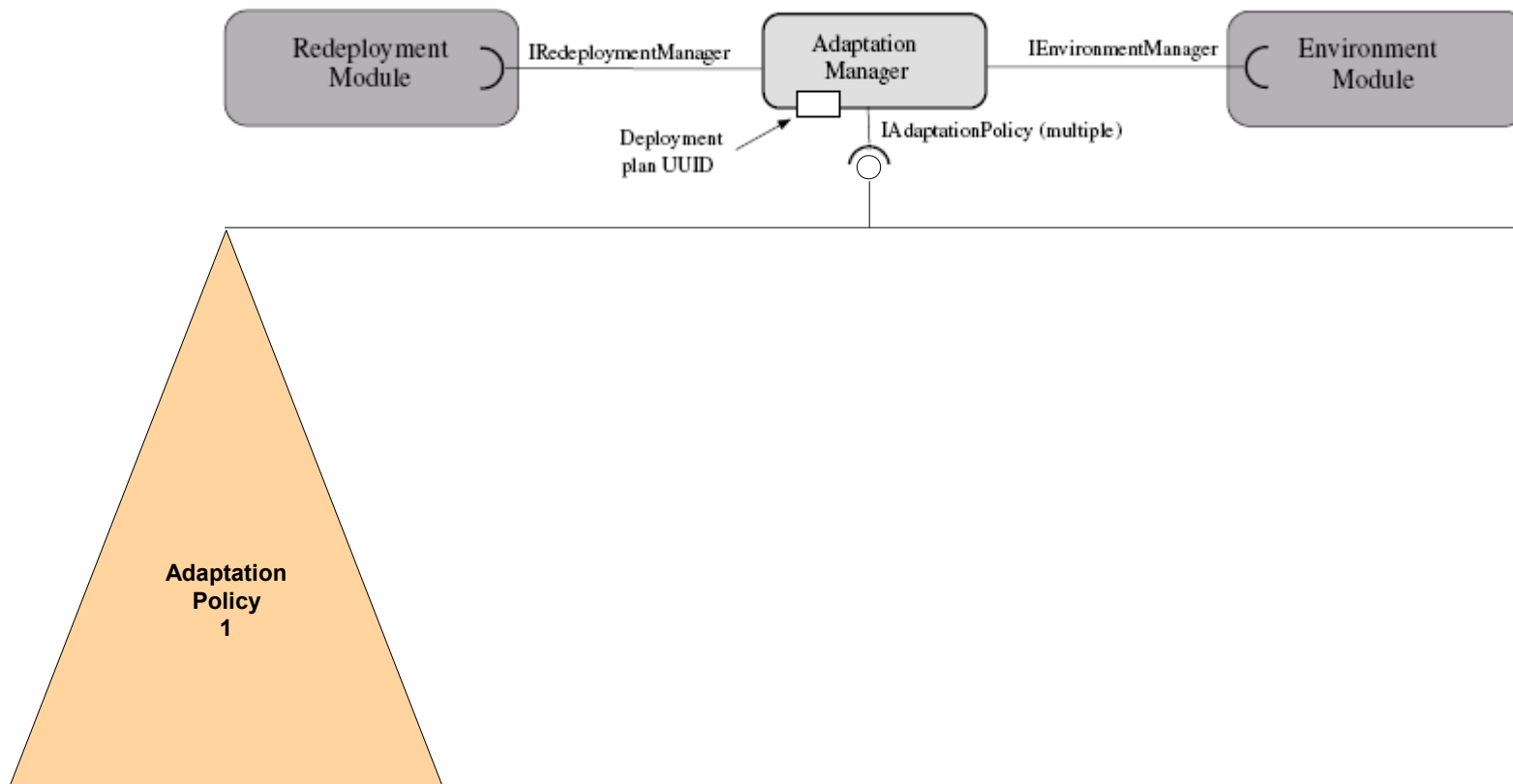
# The Proposed Framework

## 1 Adaptation module:



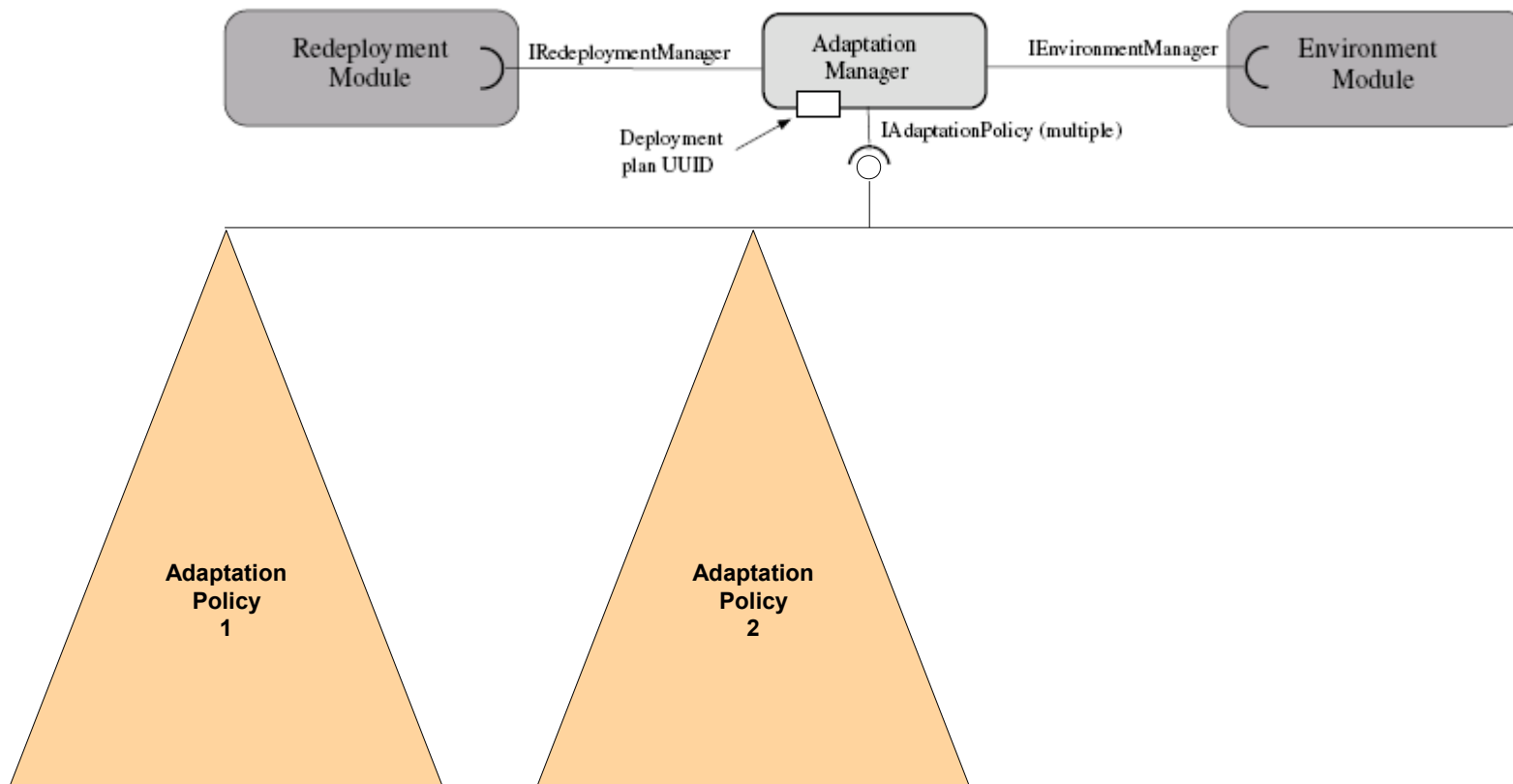
# The Proposed Framework

## 1 Adaptation module:



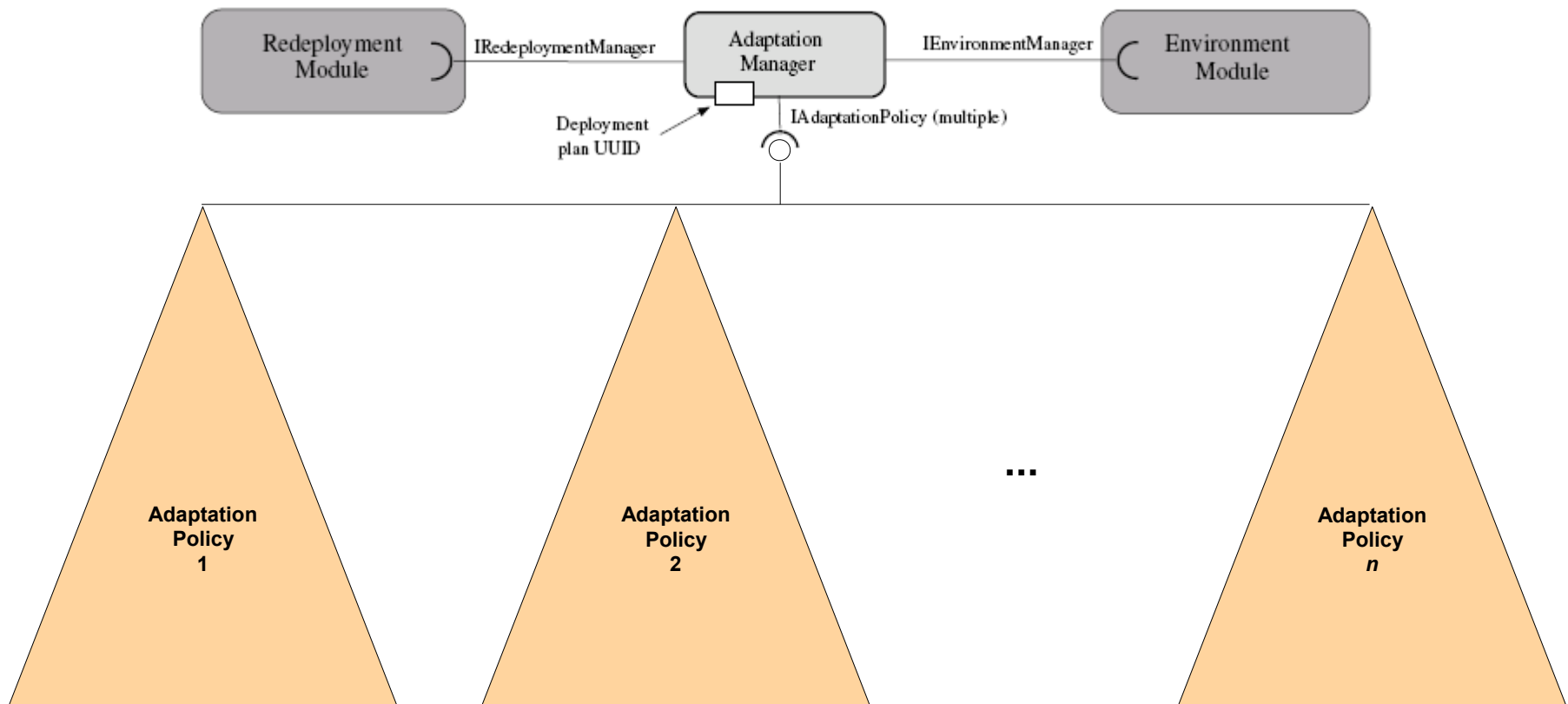
# The Proposed Framework

## 1 Adaptation module:



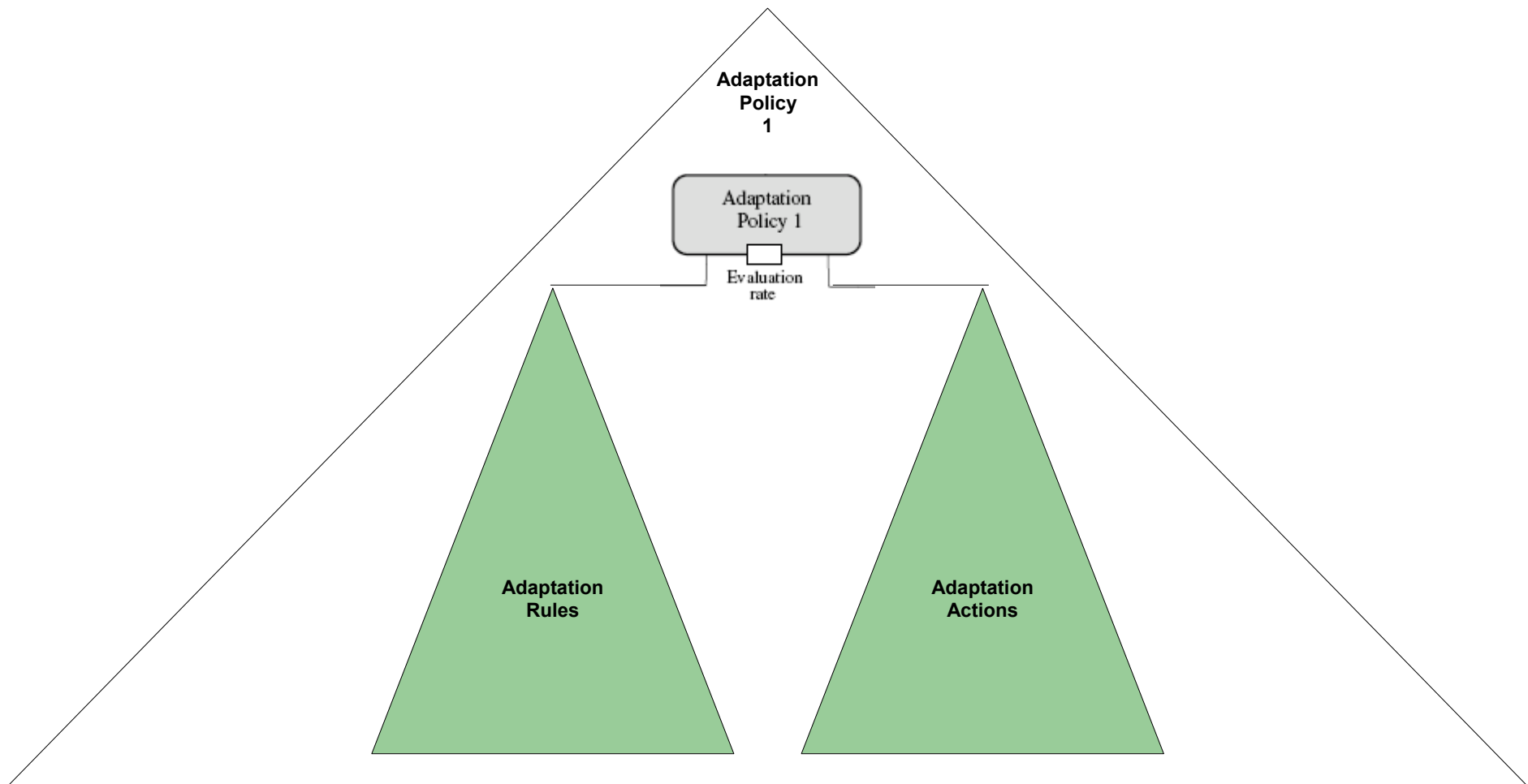
# The Proposed Framework

## 1 Adaptation module:



# The Proposed Framework

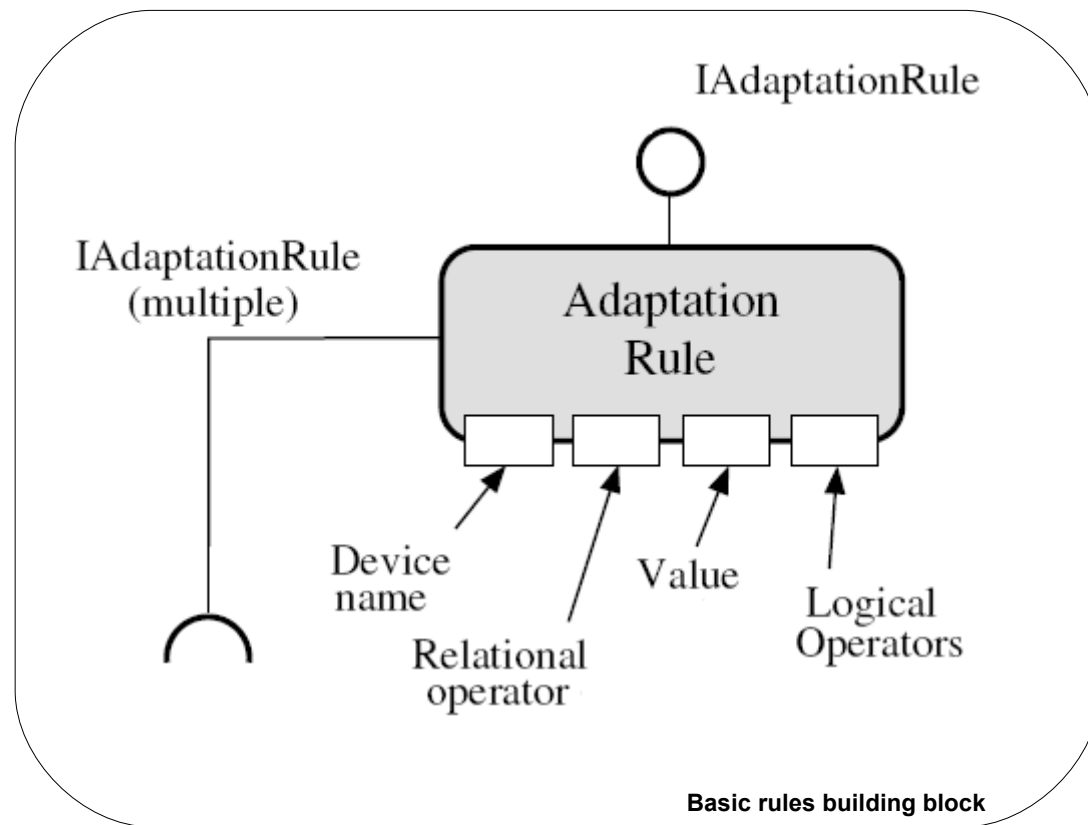
## 1 Adaptation module (*policies*):





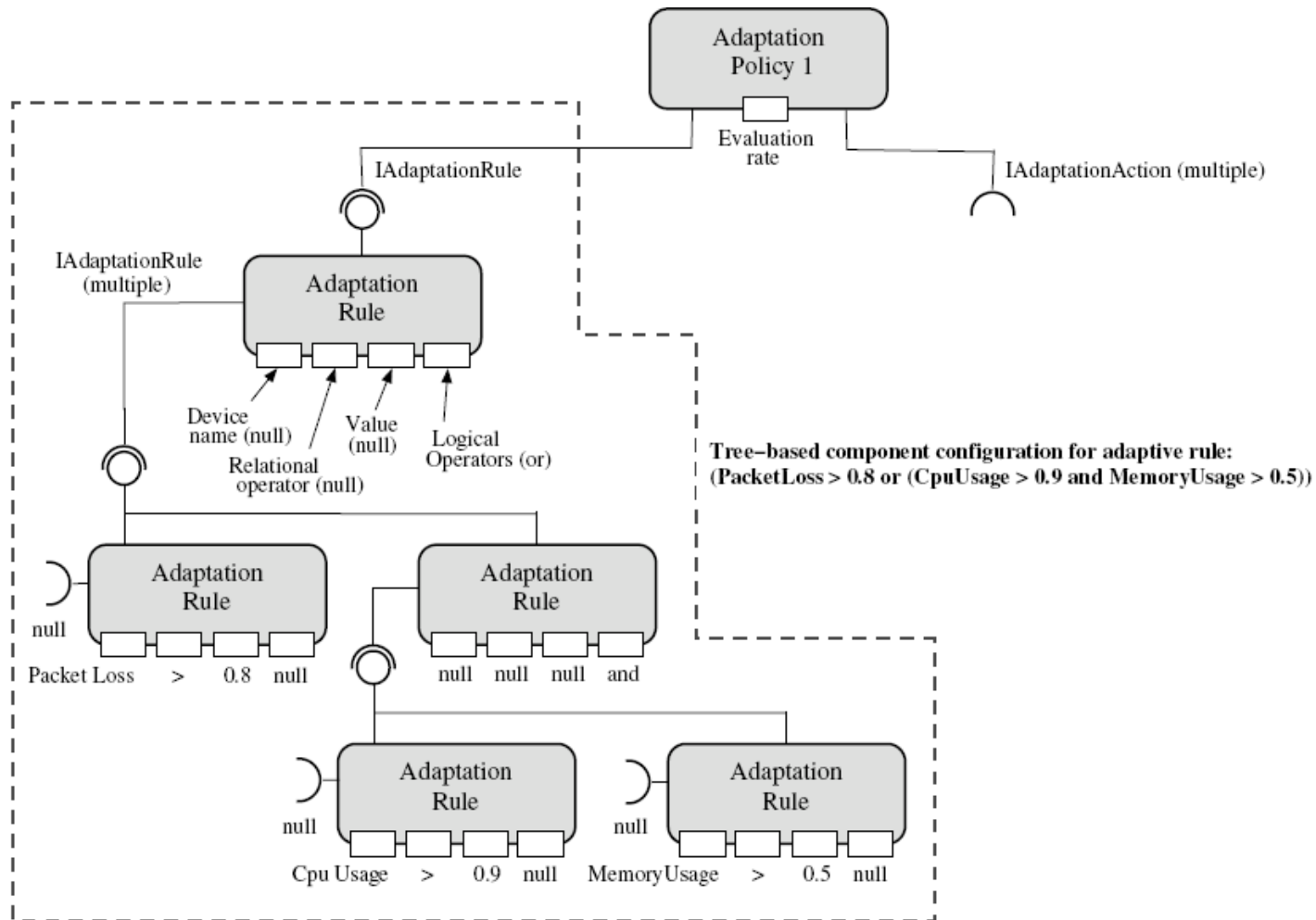
# The Proposed Framework

## 1 Adaptation module (*rules*):



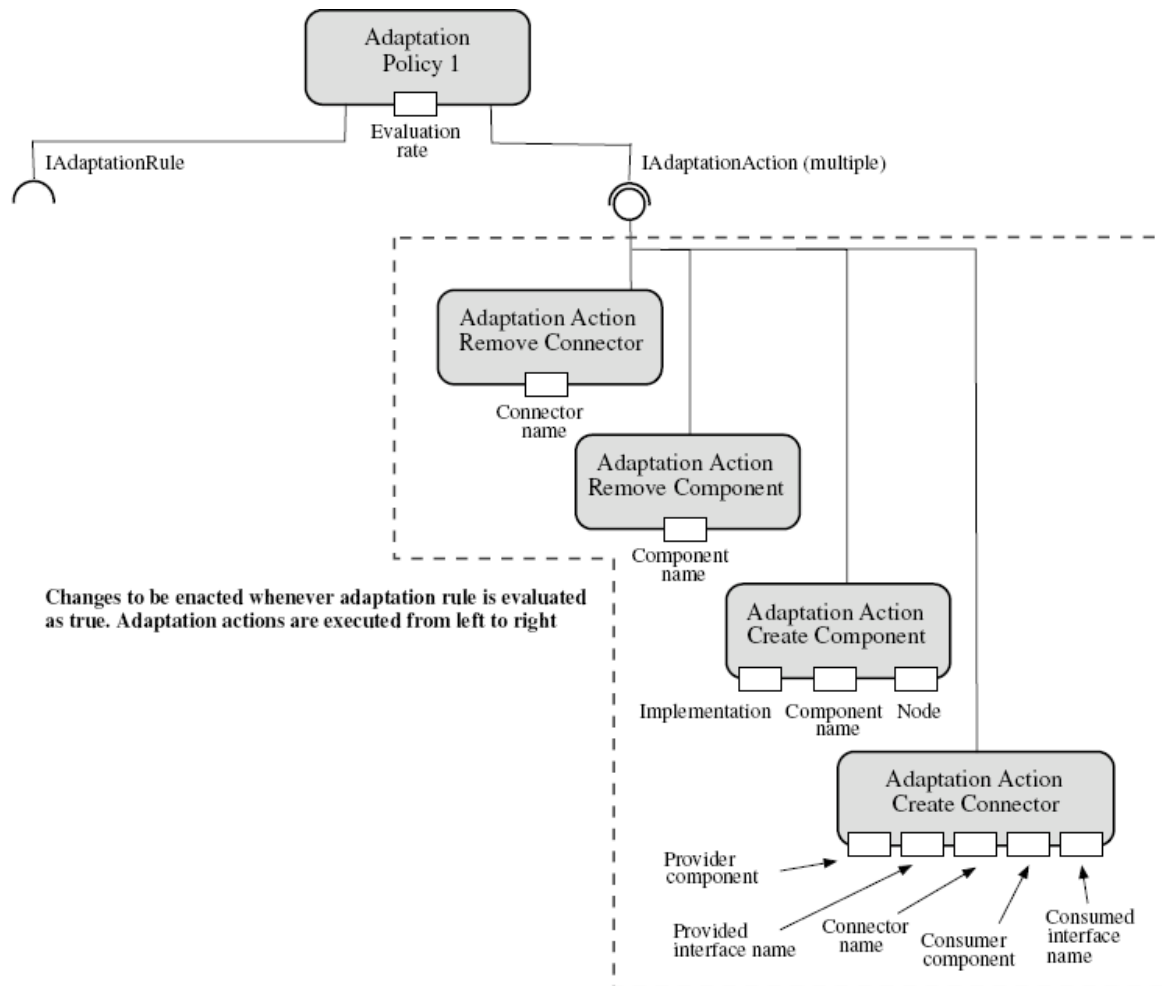
# The Proposed Framework

## 1 Adaptation module (*rules*):



# The Proposed Framework

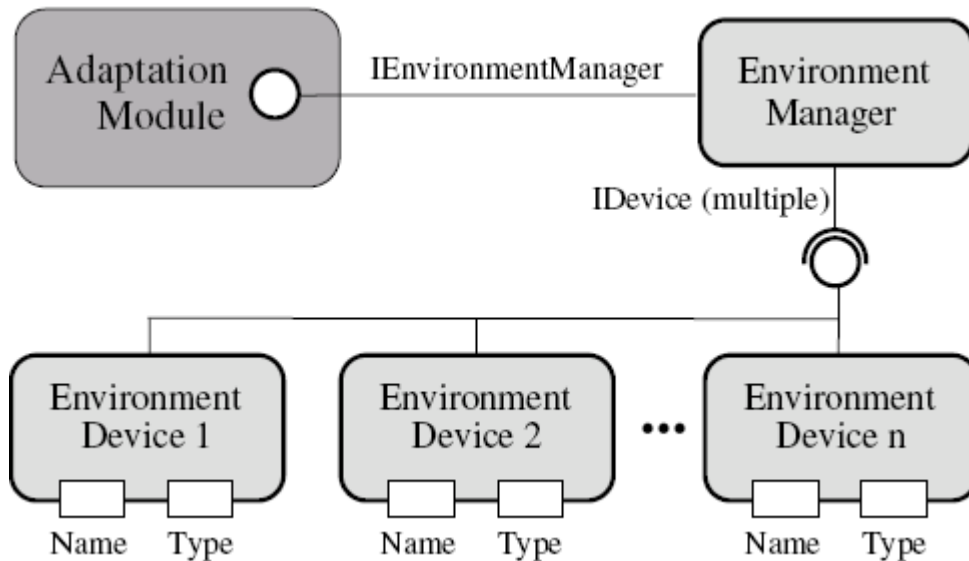
## 1 Adaptation module (*actions*):



# The Proposed Framework

## 2 Environment module:

- Environment monitoring components implement a specific interface (*Idevice*)
- Device name should match those used by adaptation policies

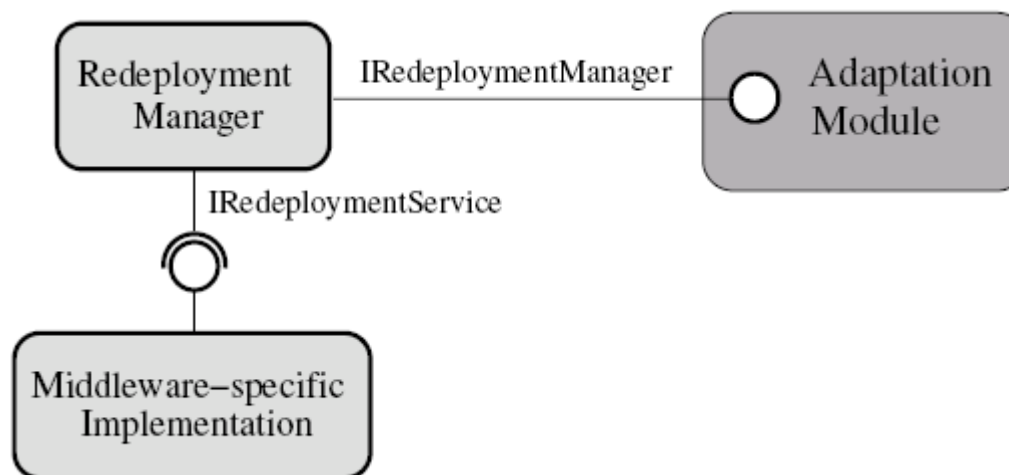


```
1 #pragma prefix "br.ufba"
2 module IRIDIUM {
3     module Environment {
4         interface IEnvironmentManager {
5             IDevice get_device_by_name(
6                 in string device_name);
7         };
8         interface IDevice {
9             string data();
10        };
11        component CpuUsage {
12            provides IDevice i_CpuUsage;
13            attribute string name;
14            attribute string type;
15        };
16    };
17 };
```

# The Proposed Framework

## 3 Redeployment module:

- Provides the glue code for a specific runtime support environment
- *IRedeploymentService* define methods for handling the adaptation actions



# Implementation Issues

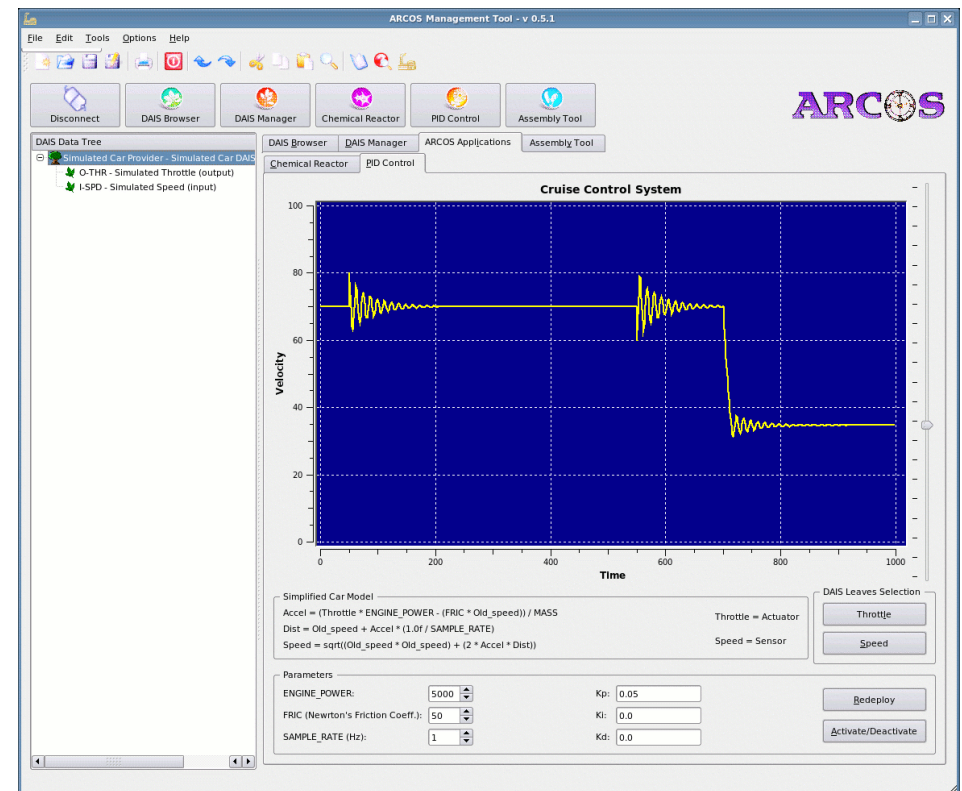
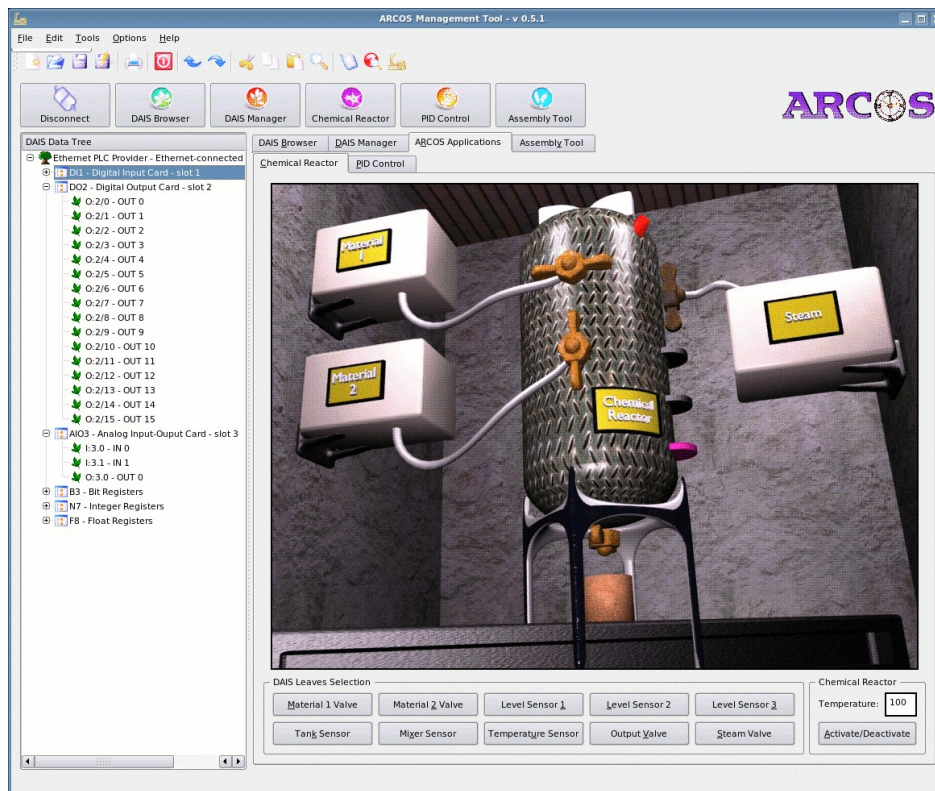
- Fully implemented on top of CIAO (*Component-Integrated ACE ORB*)
- DAnCE
  - XML deployment descriptors
  - Remote deployment, remote connections, component repository
- ReDaC
  - Dynamic redeployment of architectural changes
  - Basic services for consistency maintenance

# Deploying Self-Management

- Required steps:
  - Identify self-management requirements (adaptation rules and actions)
  - Identify required environment monitoring devices
  - Implement or reuse required environment monitoring devices
  - Create observer component configuration by composing the provided building blocks
  - Deploy observer system

# Evaluation Experiments

- ARCOS platform provides reusable services for interoperable industrial data acquisition, closed control loops, and supervisory activities





# Evaluation Experiments

- ARCOS platform provides reusable services for interoperable industrial data acquisition, closed control loops, and supervisory activities

ARCOS Management Tool - v 0.5.1

File Edit Tools Options Help

Disconnect DAIS Browser DAIS Manager Chemical Reactor PID Control Assembly Tool

ARCOS

DAIS Data Tree

- Ethernet PLC Provider - Ethernet-connected
  - D11 - Digital Input Card - slot 1
    - O.2/0 - OUT 0
    - O.2/1 - OUT 1
    - O.2/2 - OUT 2
    - O.2/3 - OUT 3
    - O.2/4 - OUT 4
    - O.2/5 - OUT 5
    - O.2/6 - OUT 6
    - O.2/7 - OUT 7
    - O.2/8 - OUT 8
    - O.2/9 - OUT 9
    - O.2/10 - OUT 10
    - O.2/11 - OUT 11
    - O.2/12 - OUT 12
    - O.2/13 - OUT 13
    - O.2/14 - OUT 14
    - O.2/15 - OUT 15
  - AI03 - Analog Input-Output Card - slot 3
    - I.3.0 - IN 0
    - I.3.1 - IN 1
    - O.3.0 - OUT 0
  - B3 - Bit Registers
  - N7 - Integer Registers
  - F8 - Float Registers

DAIS Leaves Selection

Material 1 Valve Material 2 Valve Level Sensor 1 Level Sensor 2 Level Sensor 3

Tank Sensor Mixer Sensor Temperature Sensor Output Valve Steam Valve

Chemical Reactor Temperature: 100

Activate/Deactivate

ARCOS Management Tool - v 0.5.1

File Edit Tools Options Help

Disconnect DAIS Browser DAIS Manager Chemical Reactor PID Control Assembly Tool

ARCOS

DAIS Data Tree

- Simulated Car Provider - Simulated Car DAIS
  - O-THR - Simulated Throttle (output)
  - I-SPD - Simulated Speed (input)

Chemical Reactor PID Control

Cruise Control System

Velocity

Time

Simplified Car Model

$$\text{Accel} = (\text{Throttle} * \text{ENGINE\_POWER} - (\text{FRIC} * \text{Old\_speed})) / \text{MASS}$$
$$\text{Dist} = \text{Old\_speed} + \text{Accel} * (1.0f / \text{SAMPLE\_RATE})$$
$$\text{Speed} = \text{sqrt}(\text{Old\_speed} * \text{Old\_speed} + (2 * \text{Accel} * \text{Dist}))$$

Throttle = Actuator

Speed = Sensor

Parameters

ENGINE\_POWER: 5000

FRIC (Newton's Friction Coeff.): 50

SAMPLE\_RATE (Hz): 1

Kp: 0.05

Ki: 0.0

Kd: 0.0

DAIS Leaves Selection

Throttle

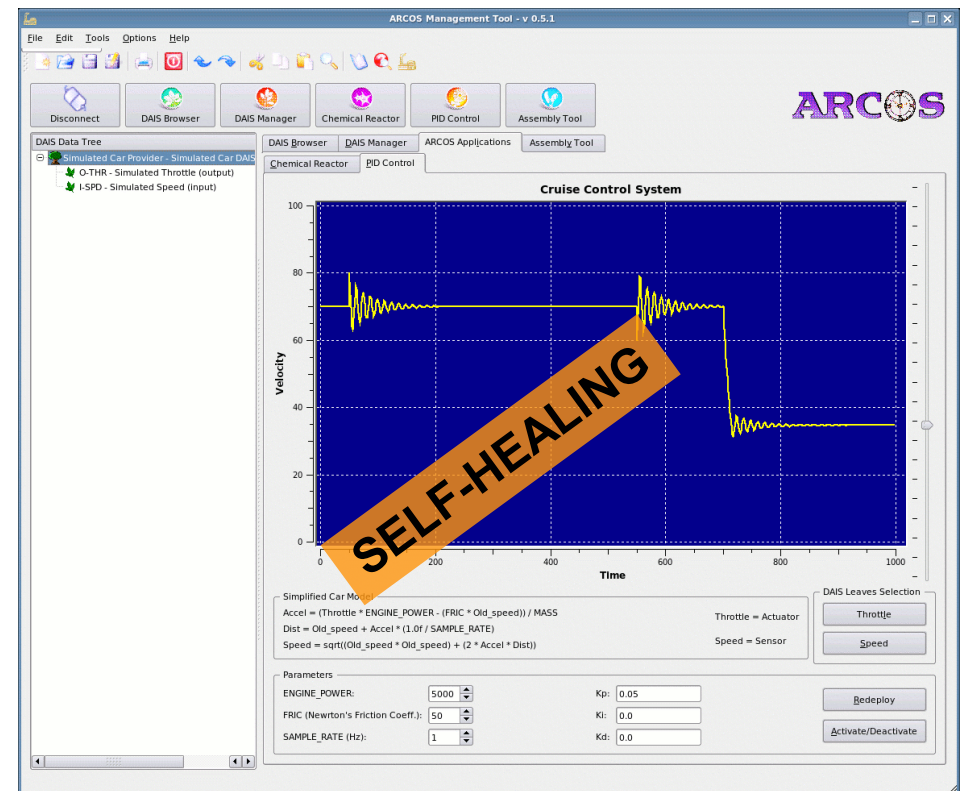
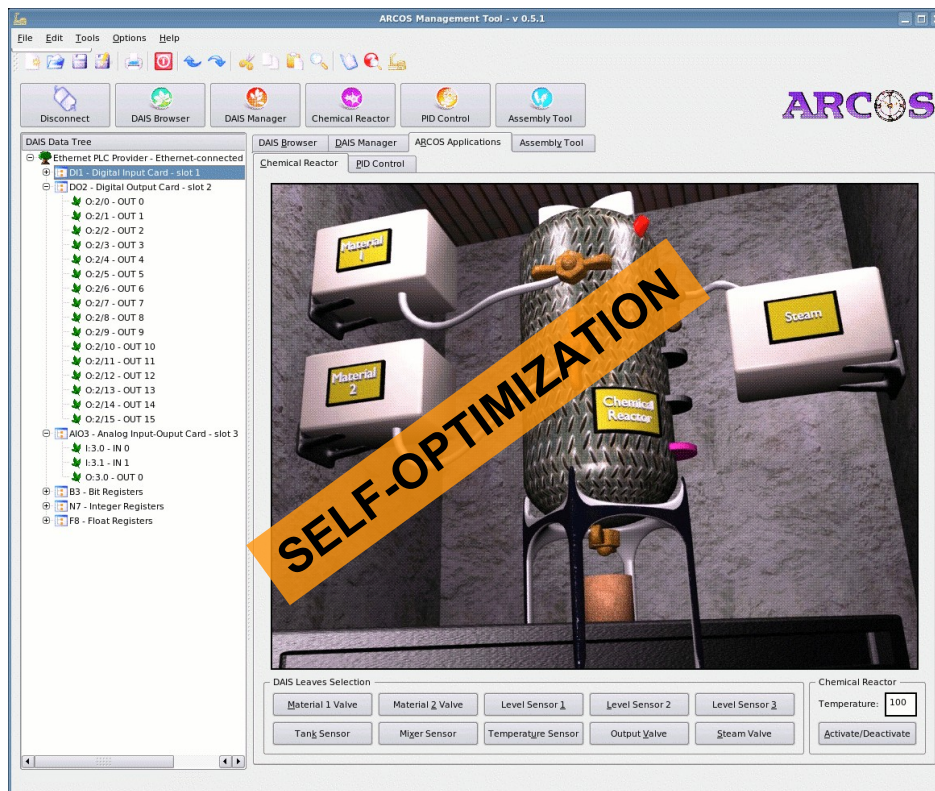
Speed

Bedeploy

Activate/Deactivate

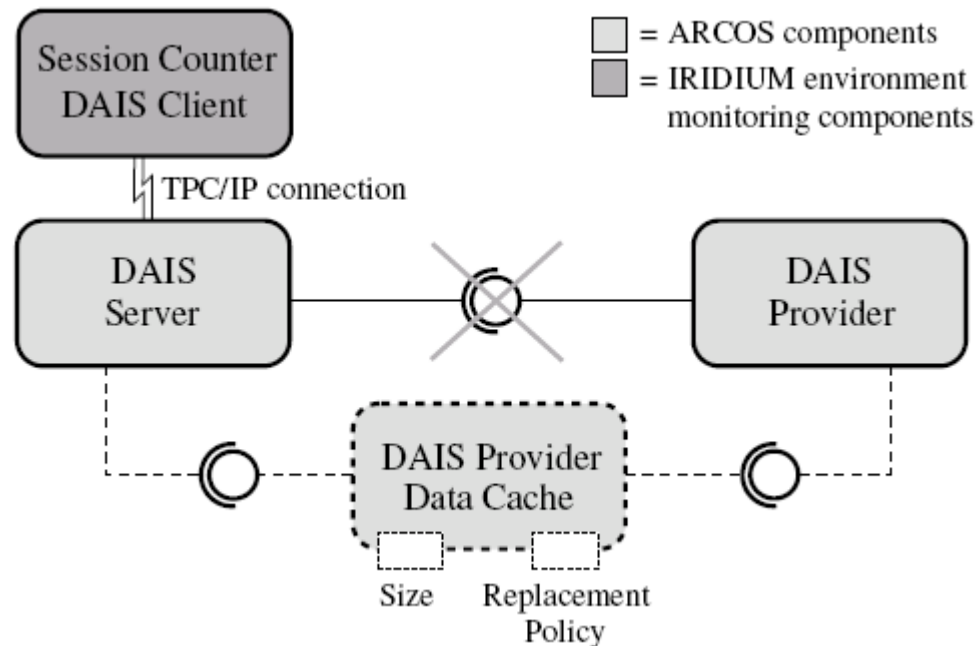
# Evaluation Experiments

- ARCOS platform provides reusable services for interoperable industrial data acquisition, closed control loops, and supervisory activities



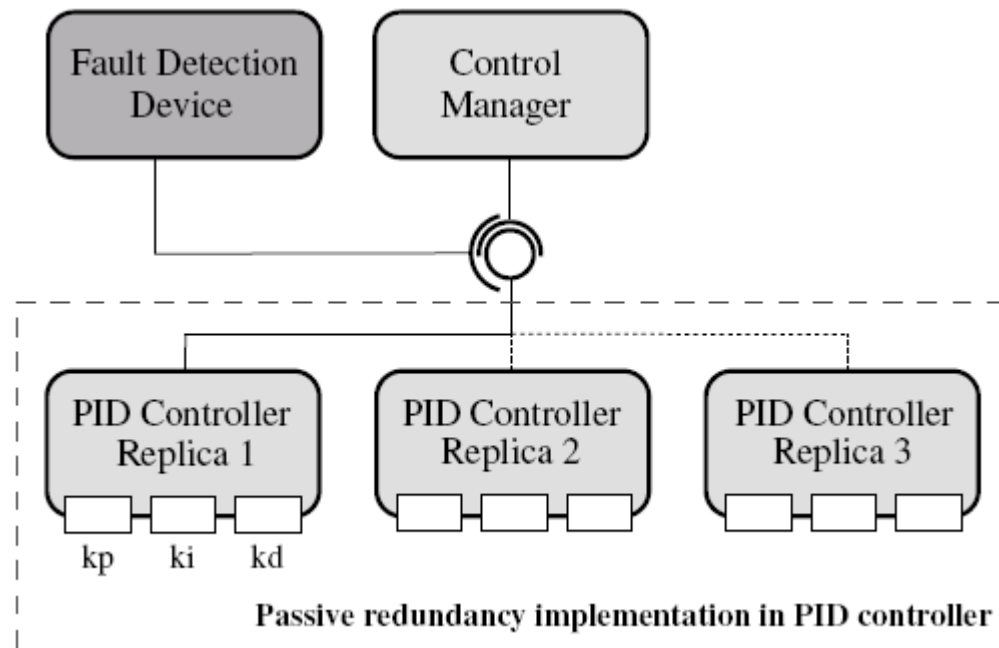
# Evaluation Experiments

## 1 Self-optimization



# Evaluation Experiments

## 2 Self-healing



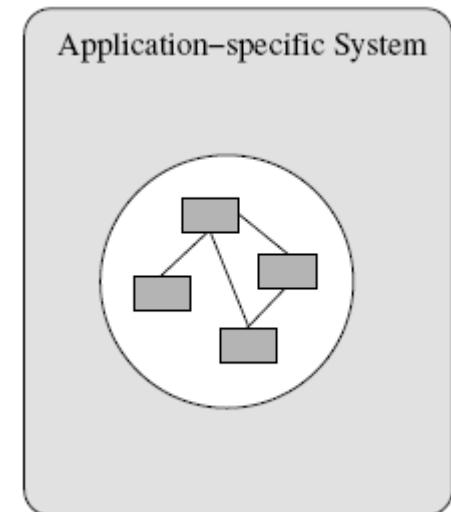


# Unanticipated Changes

- Unanticipated = unforeseen when creating the observer component configuration
- As the adaptation model itself is deployed as a component configuration it is able to undergo adaptation
- This can be done:
  - Manually: by requiring runtime changes in the observer component configuration
  - Automatically: by deploying another running instance of our framework

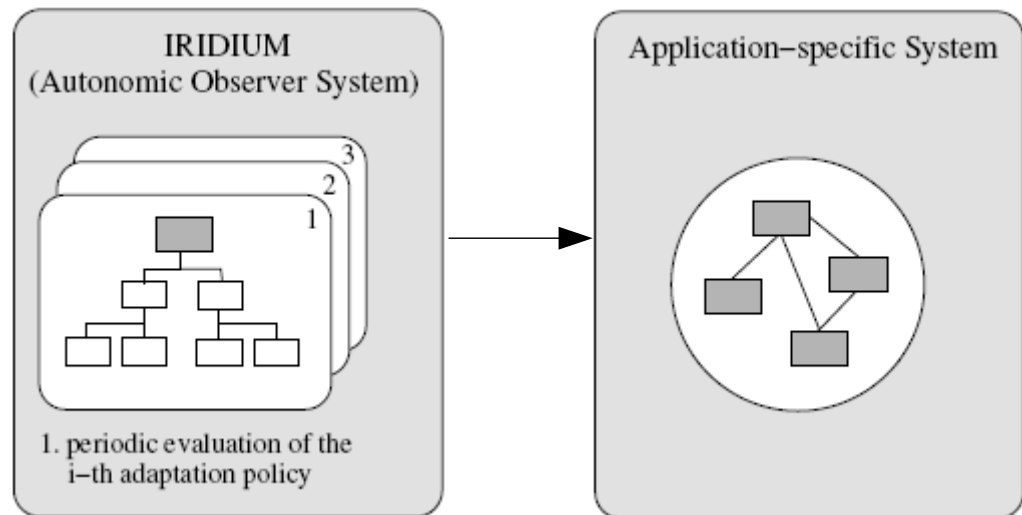
# Unanticipated Changes

- Automatically: by deploying another running instance of our framework



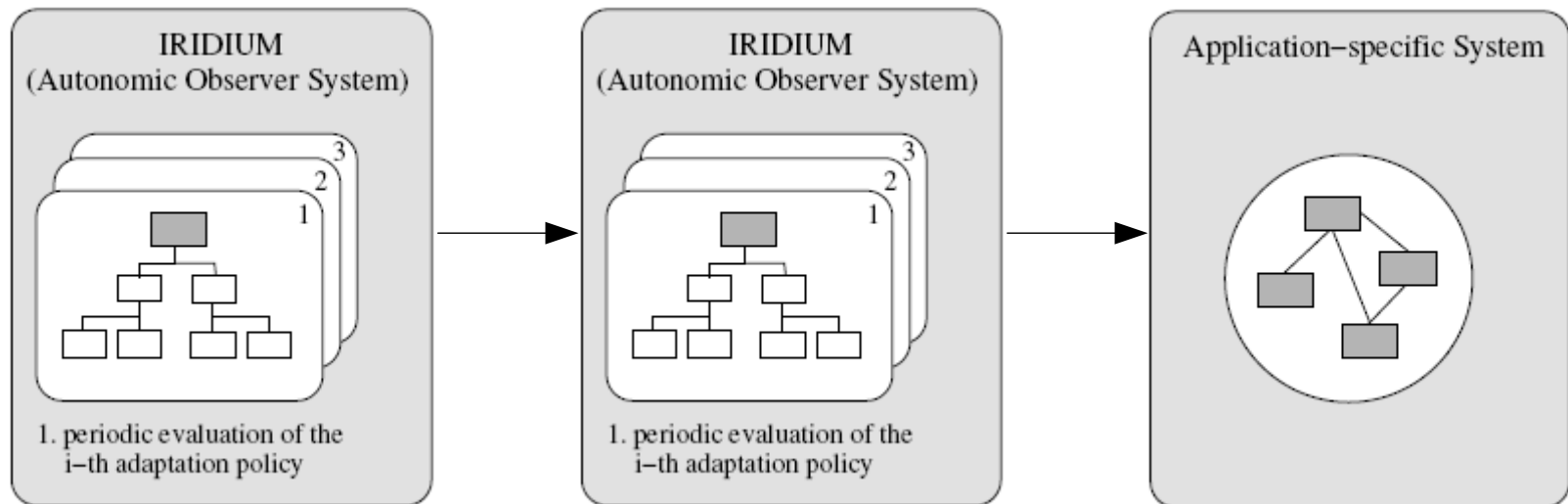
# Unanticipated Changes

- Automatically: by deploying another running instance of our framework



# Unanticipated Changes

- Automatically: by deploying another running instance of our framework





# Conclusion and Future Work

- Component-based representation of adaptation policies and adaptation changes
- Implemented framework for self-management of CCM-based applications
- Future work:
  - Experiments on unanticipated changes support
  - Mechanism for handling conflicting adaptation policies
  - Planning and high-level reasoning
  - Self-management with real-time constraints
  - Performance evaluation experiments

Questions ?