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)

31st International Conference on Software Engineering - Workshop on Software Engineering for Adaptive and Self-Managing Systems

Motivation

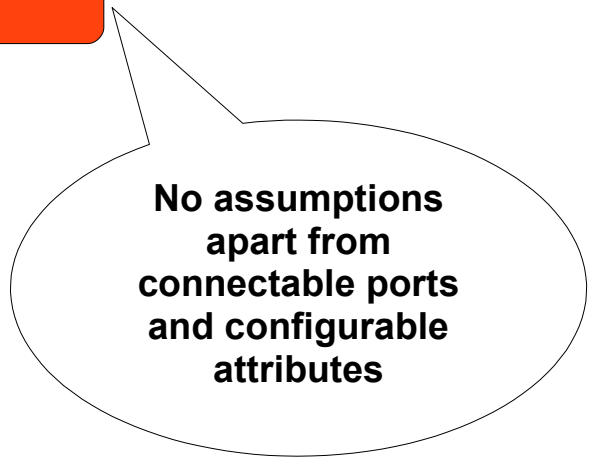
- Architecture-based approaches have proven to be useful for early rationale of non-functional requirements
- Increasing complexity demands the use of fully automatic self-managed software platforms
- Underlying runtime support environment can provide self-management behaviour to already deployed systems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

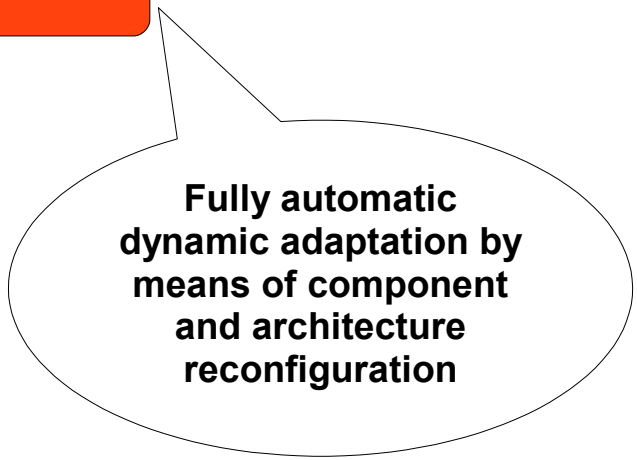
Motivation

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



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

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

Motivation

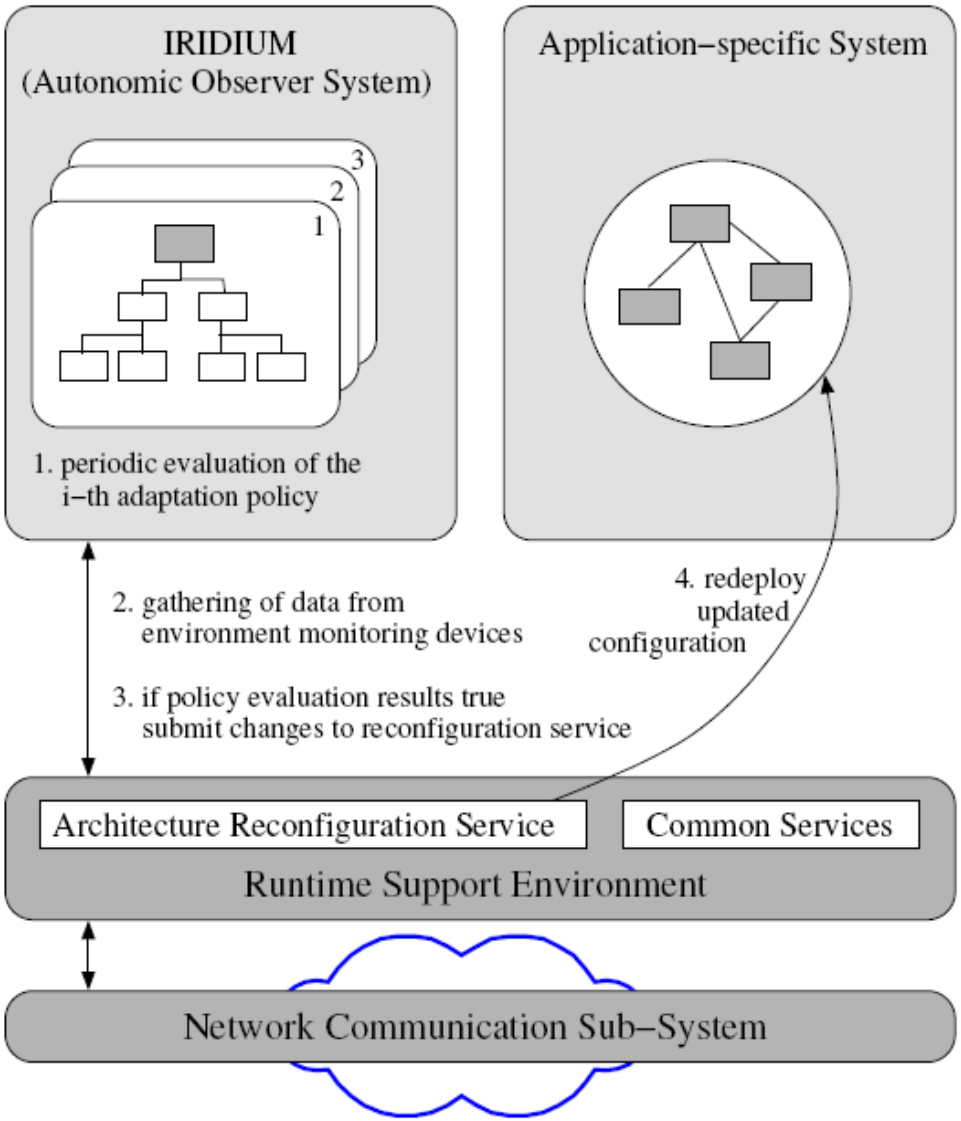
- Aspects of dynamic software architectures:
  - Underlying component model
  - Degree of autonomicity
  - Planning and high-level reasoning
  - Runtime support environment
  - Consistency and system integrity

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

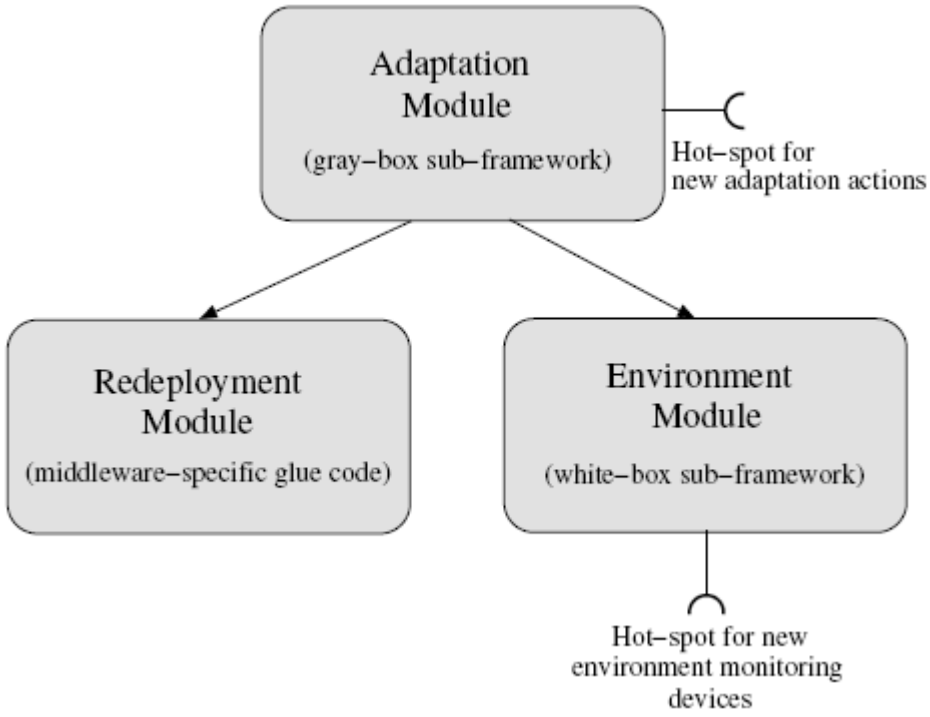
The Proposed Framework

- Our approach:



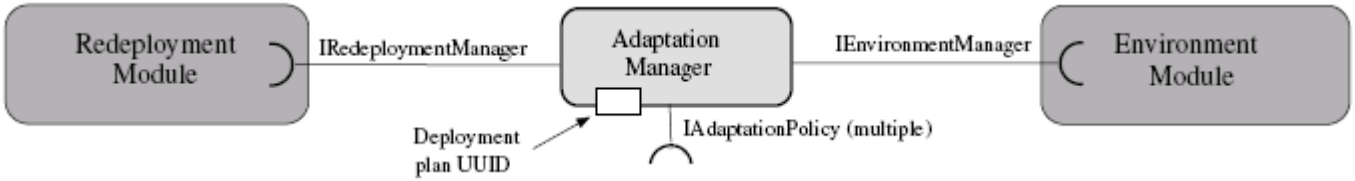
The Proposed Framework

- Framework architecture:



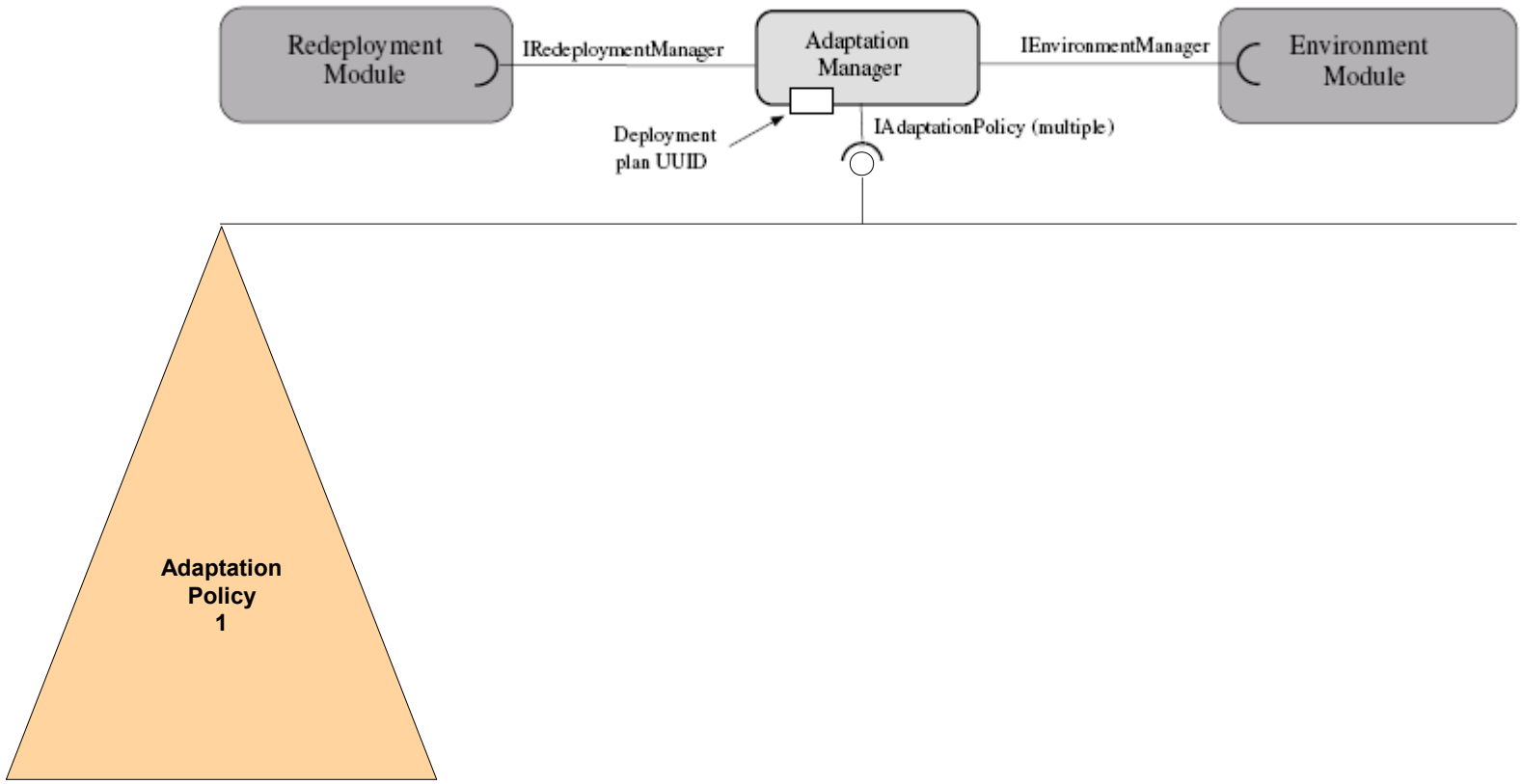
The Proposed Framework

1. Adaptation module:



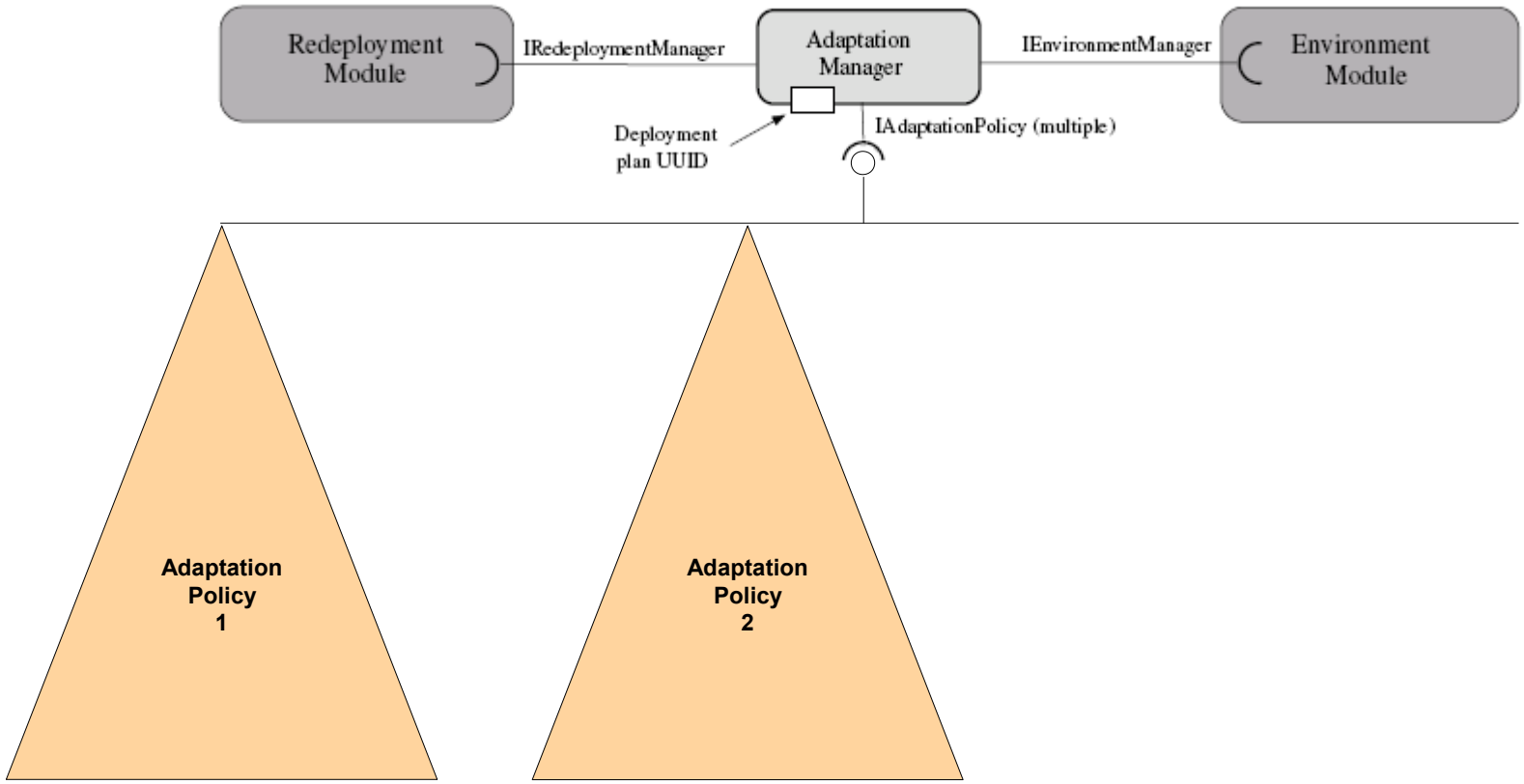
The Proposed Framework

1. Adaptation module:



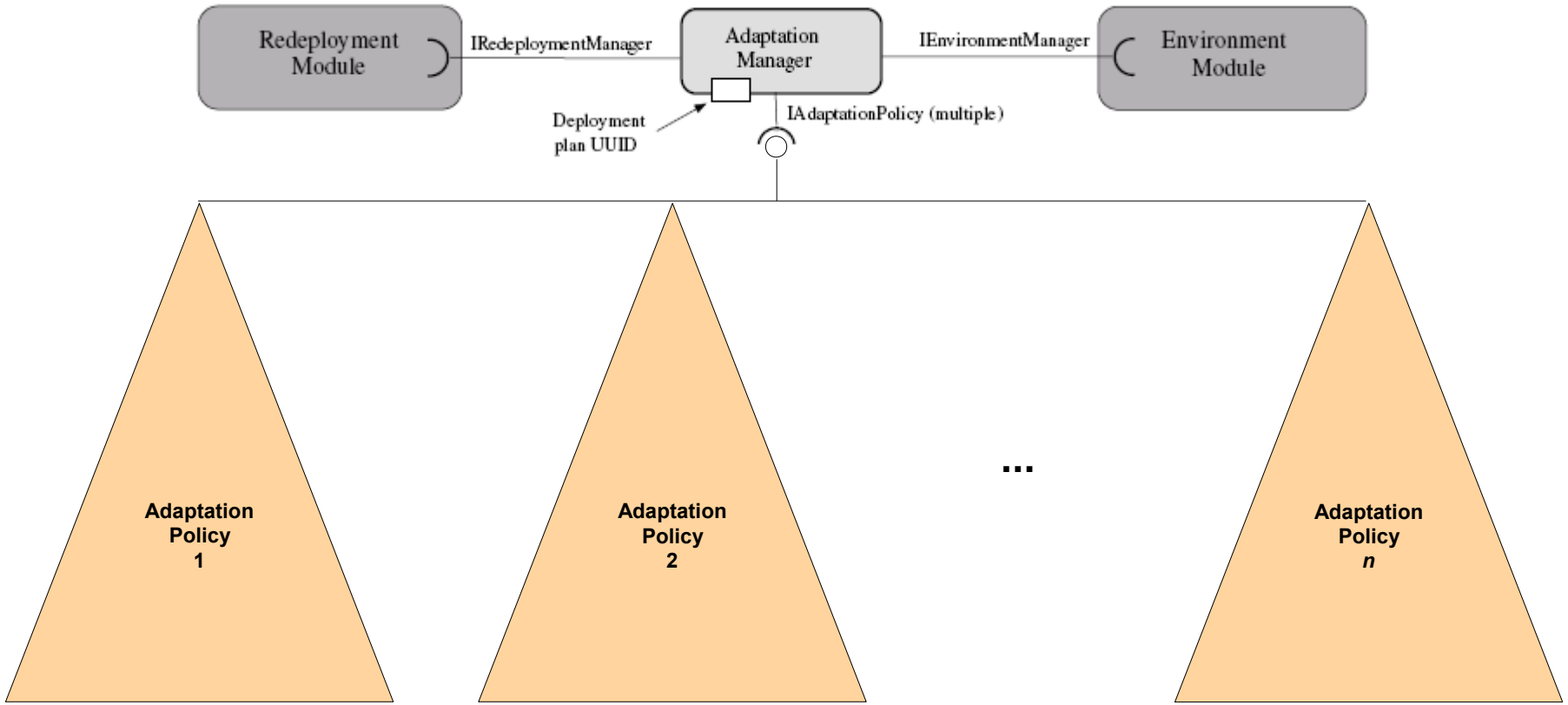
The Proposed Framework

1. Adaptation module:



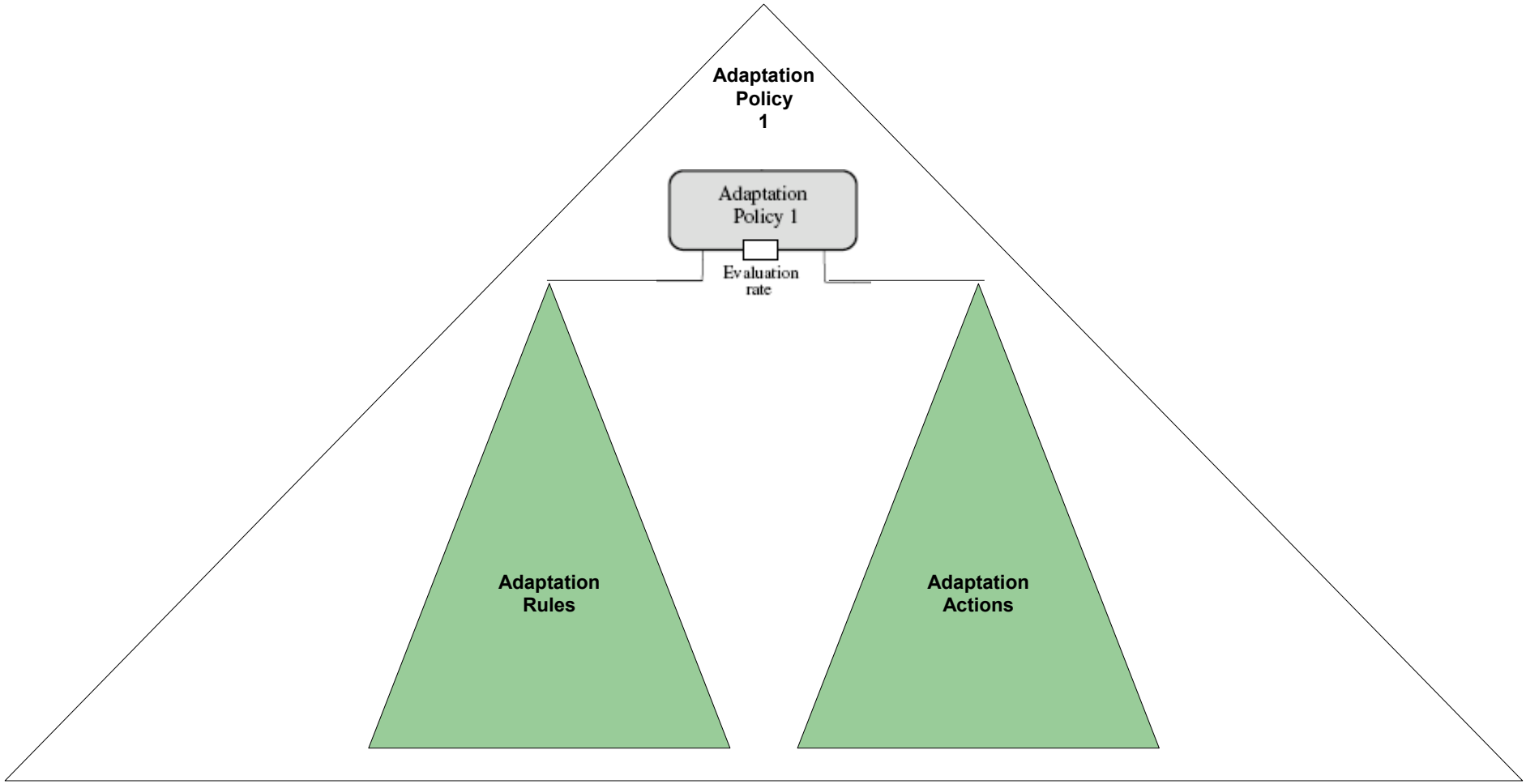
The Proposed Framework

1. Adaptation module:



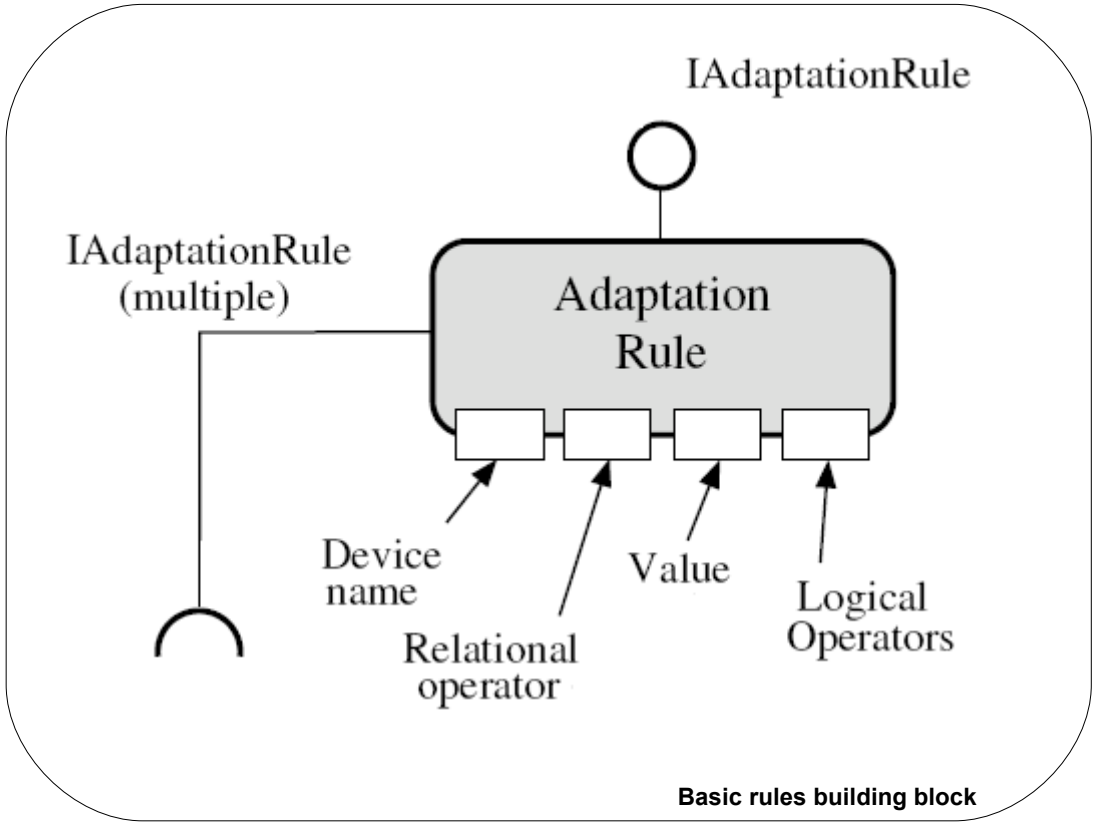
The Proposed Framework

1. Adaptation module (*policies*):



The Proposed Framework

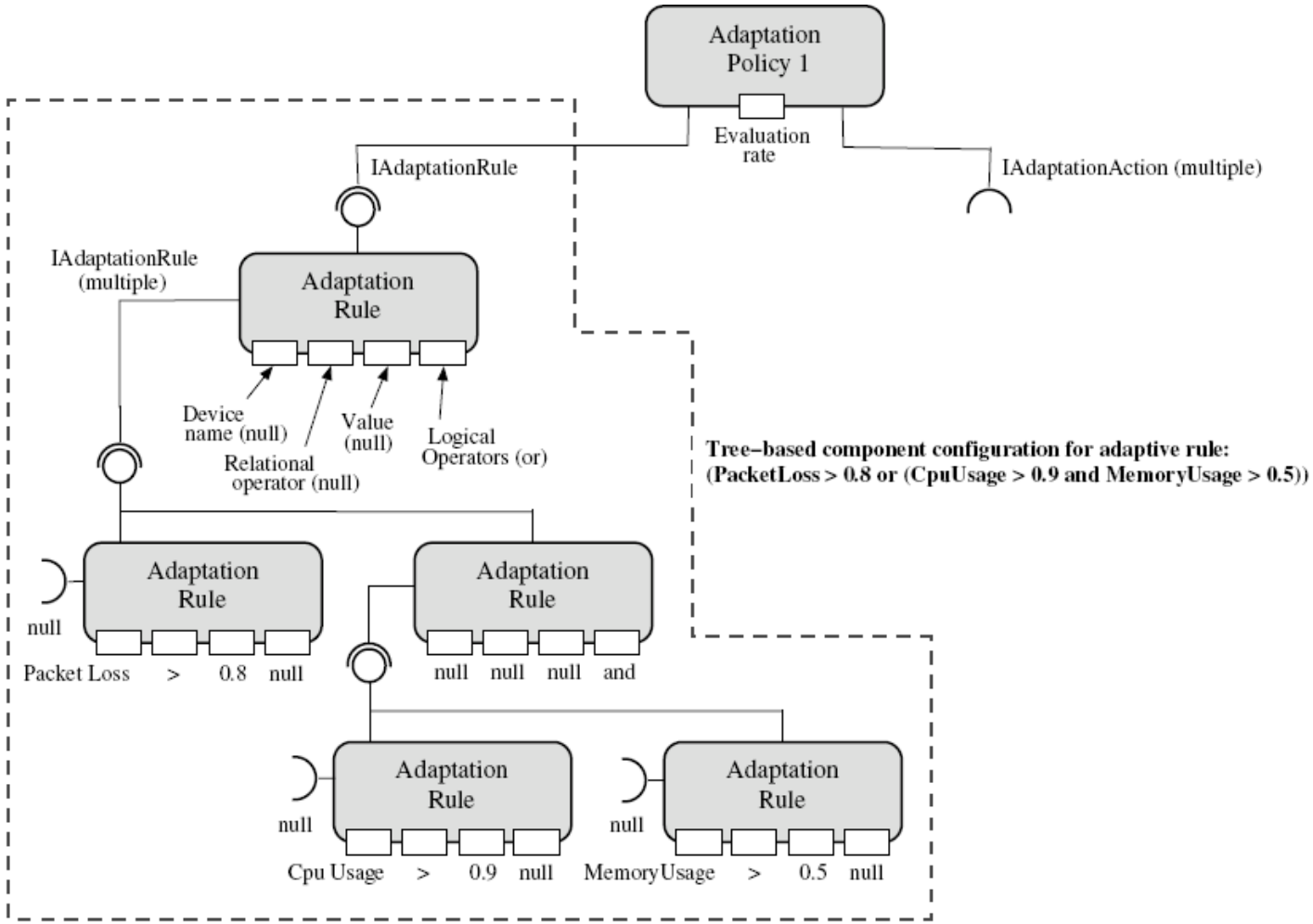
1. Adaptation module (*rules*):



Basic rules building block

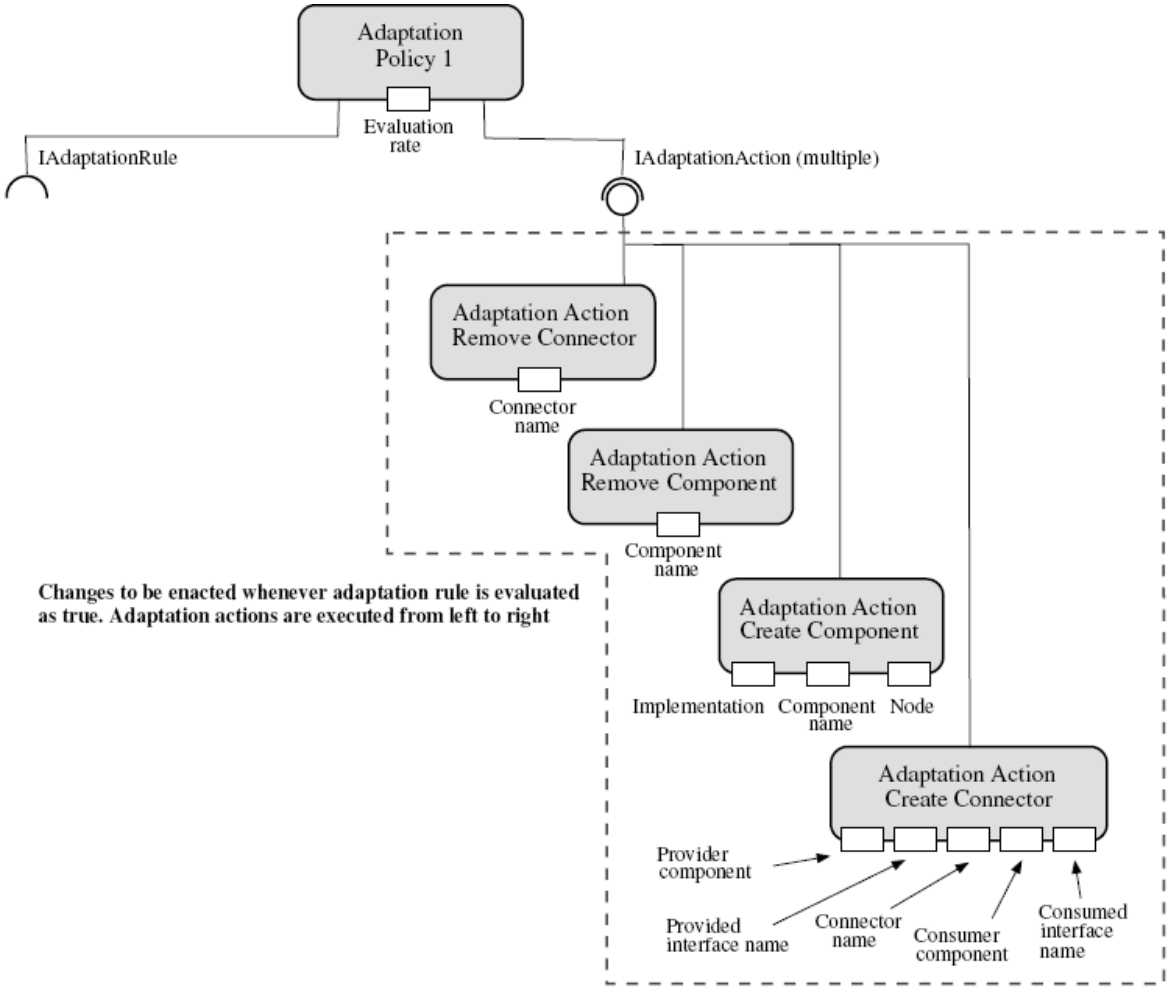
The Proposed Framework

1. Adaptation module (*rules*):



The Proposed Framework

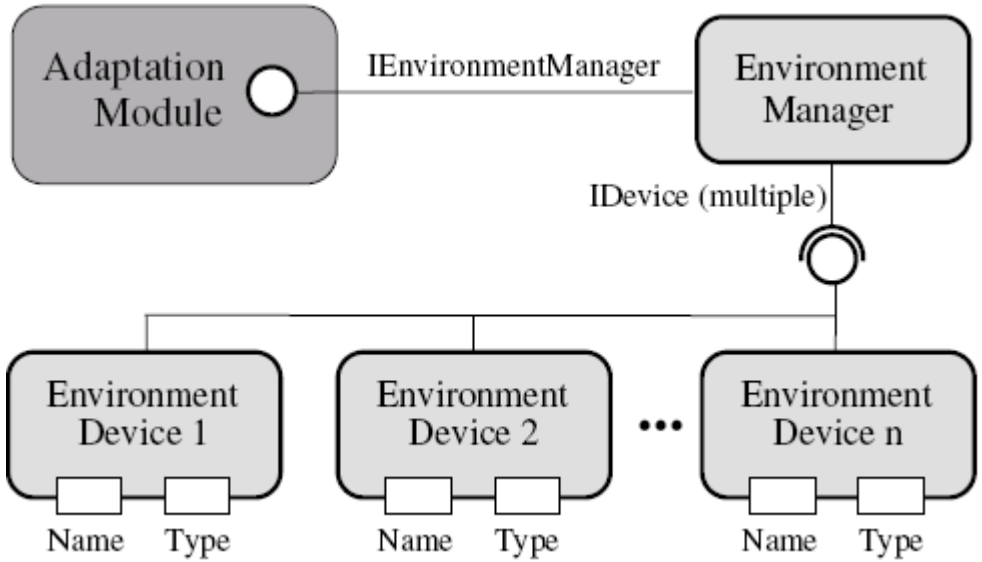
1. Adaptation module (*actions*):



The Proposed Framework

2 Environment module:

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

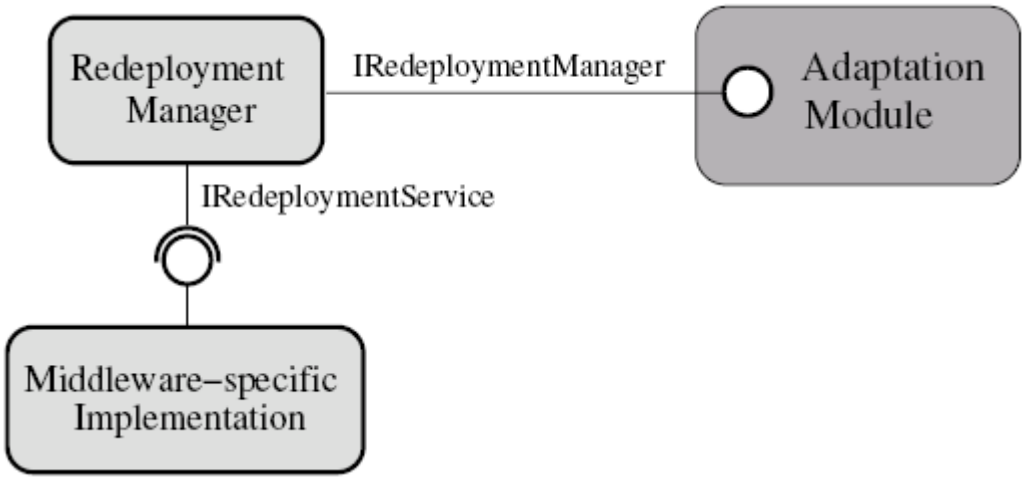



```
1  #pragma prefix "br.ufba"
2  module IRIDIUM {
3    module Environment {
4      interface IEnvironmentManager {
5        IDevice get_device_by_name(
6                              in string device_name);
7      };
8      interface IDevice {
9        string data();
10     };
11     component CpuUsage {
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

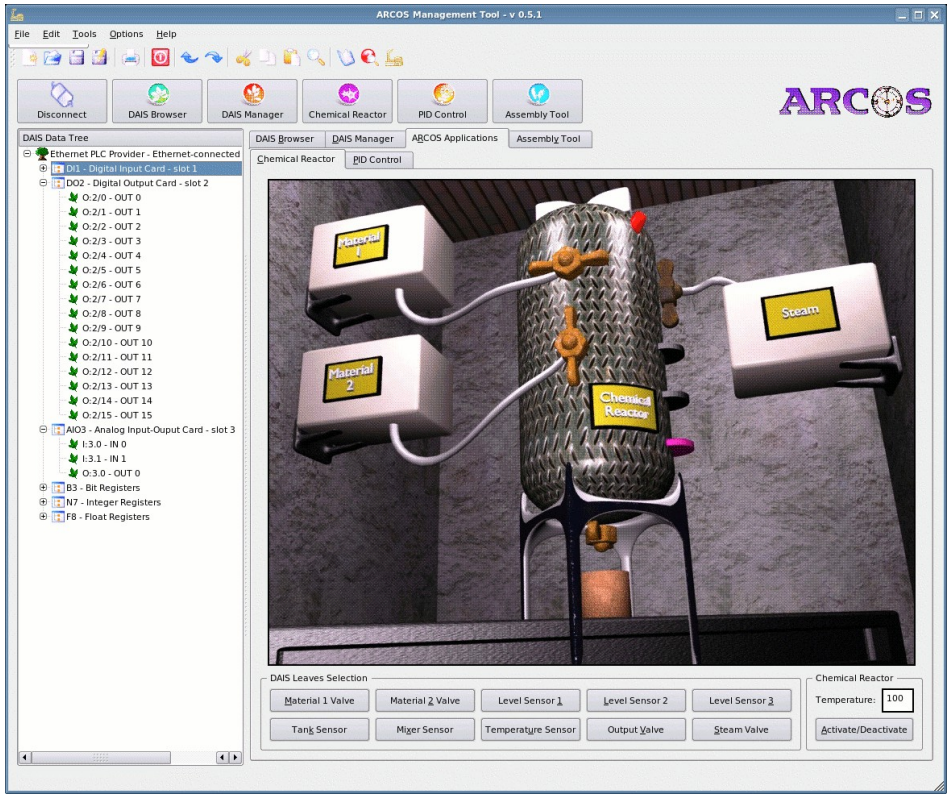
- Fully implemented on top of CIAO (*Component-Integrated ACE ORB*)
- DAnCE
  - XML deployment descriptors
  - Remote deployment, remote connections, component repository
- ReDaC
  - Dynamic redeployment of architectural changes
  - Basic services for consistency maintenance

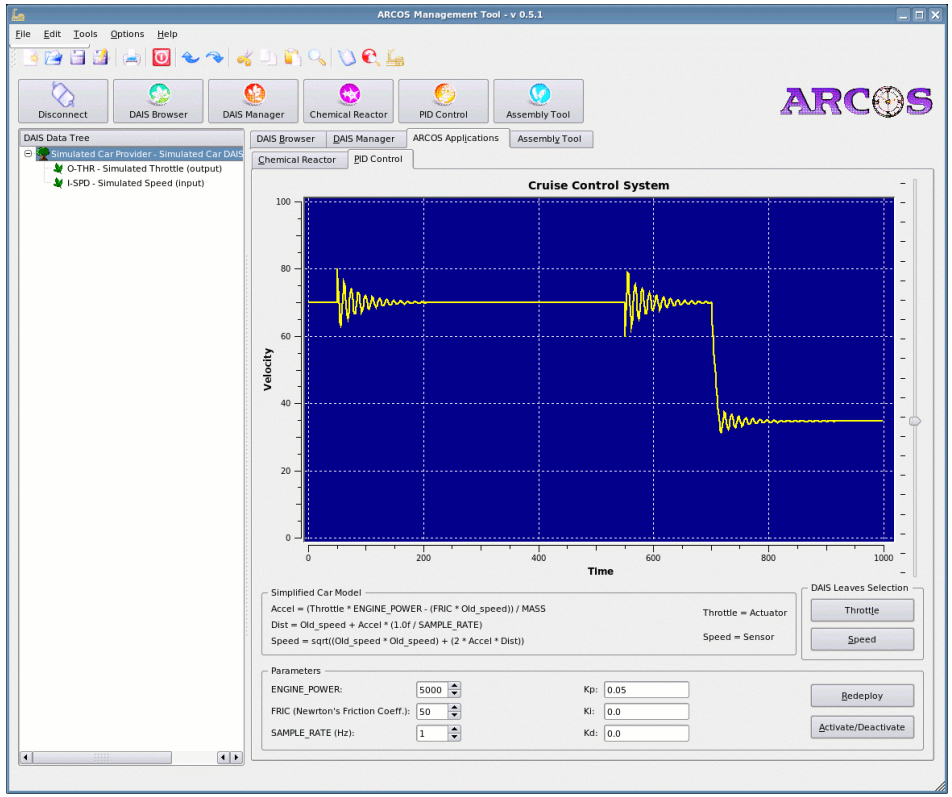
Deploying Self-Management

- 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

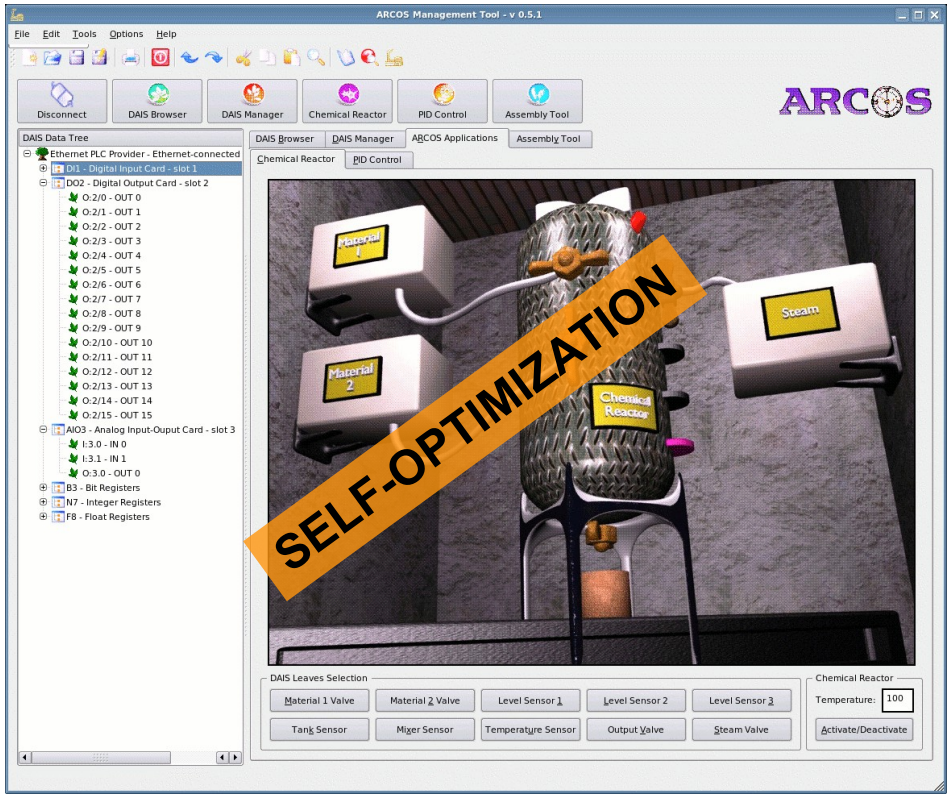
- ARCOS platform provides reusable services for interoperable industrial data acquisition, closed control loops, and supervisory activities

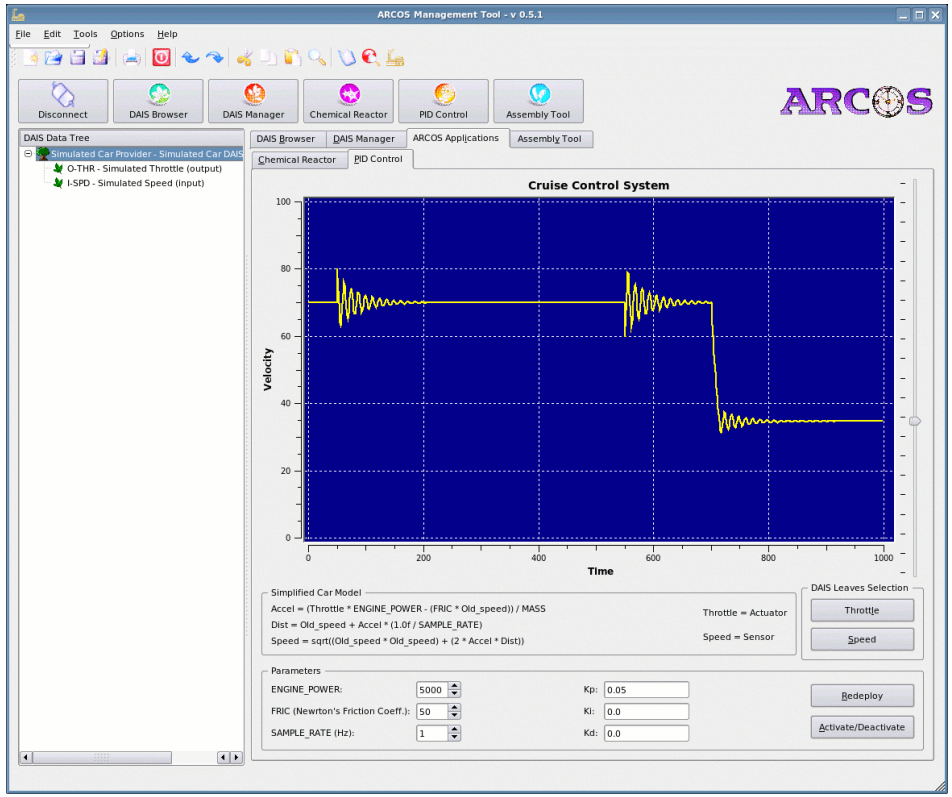




Evaluation Experiments

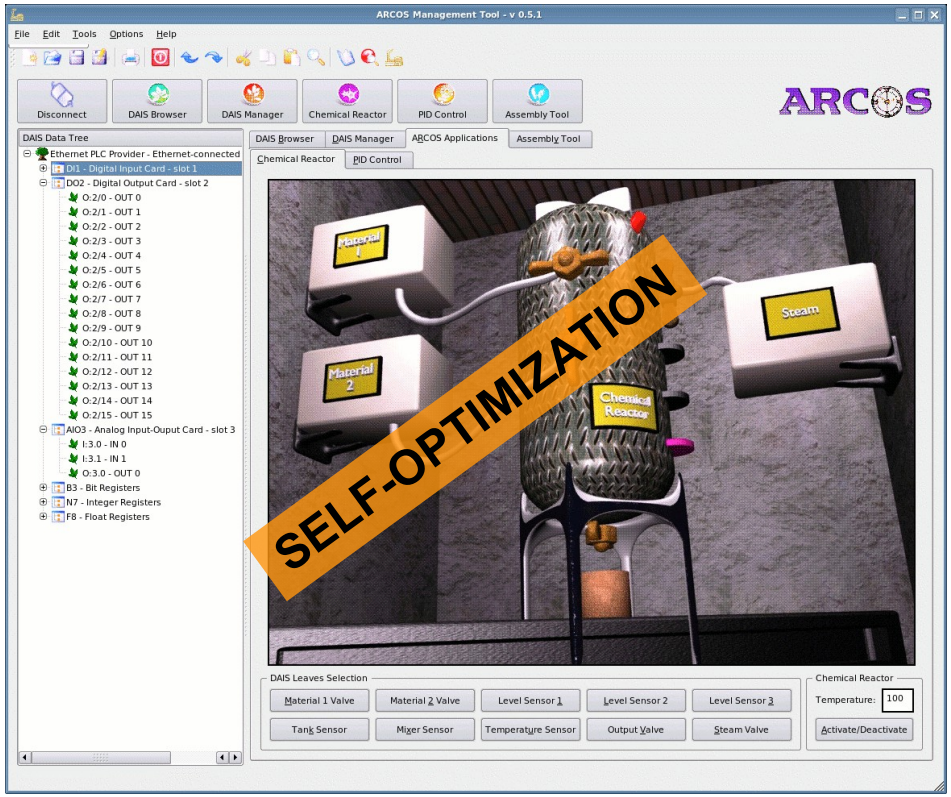
- ARCOS platform provides reusable services for interoperable industrial data acquisition, closed control loops, and supervisory activities

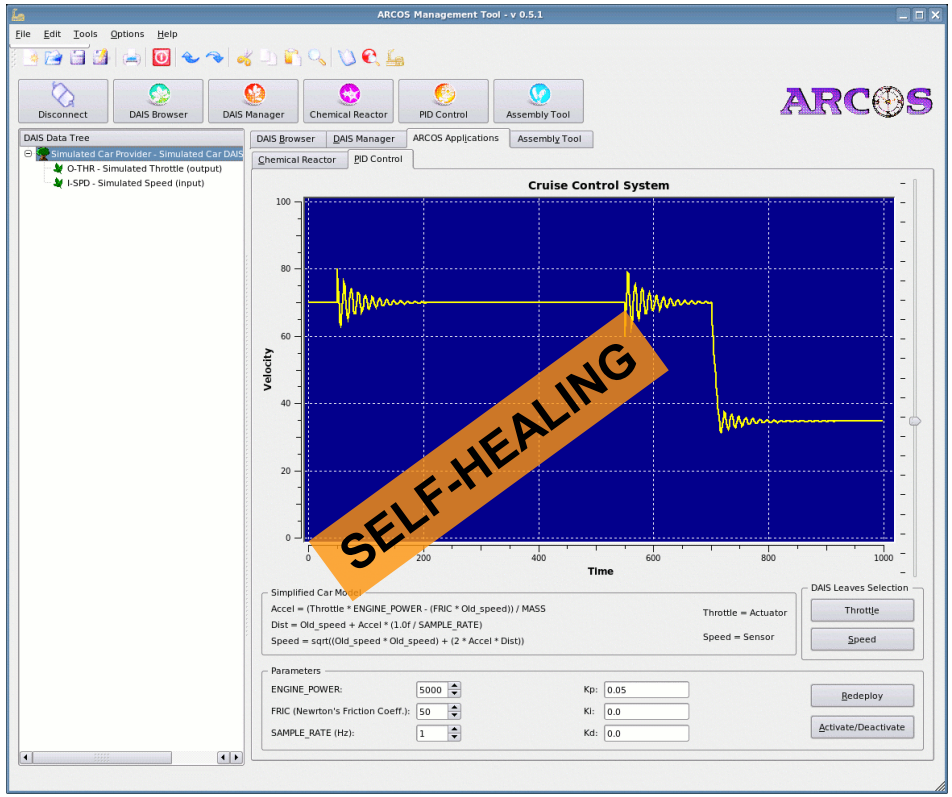




Evaluation Experiments

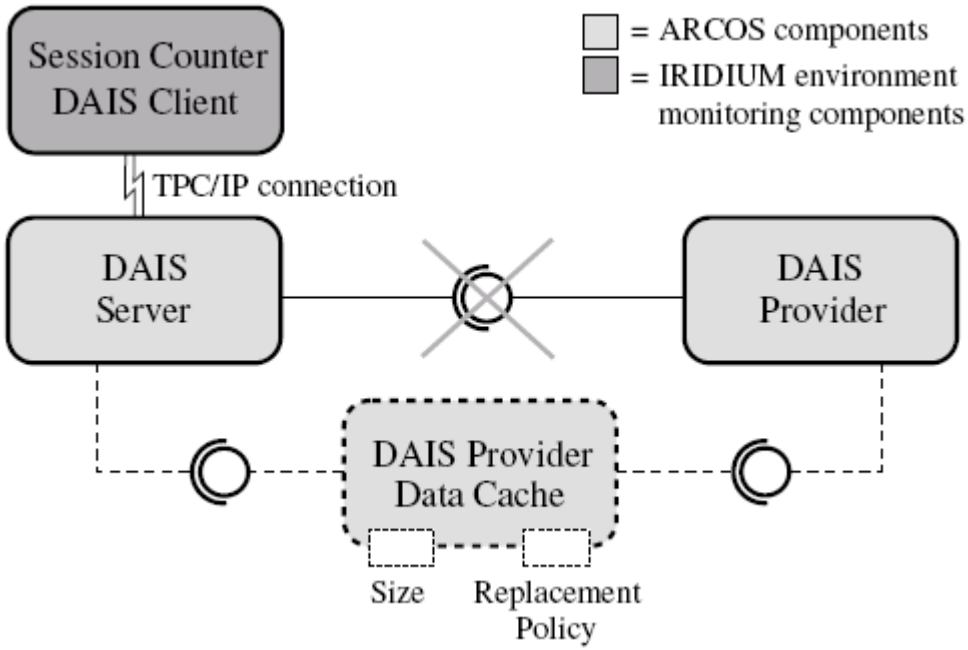
- ARCOS platform provides reusable services for interoperable industrial data acquisition, closed control loops, and supervisory activities





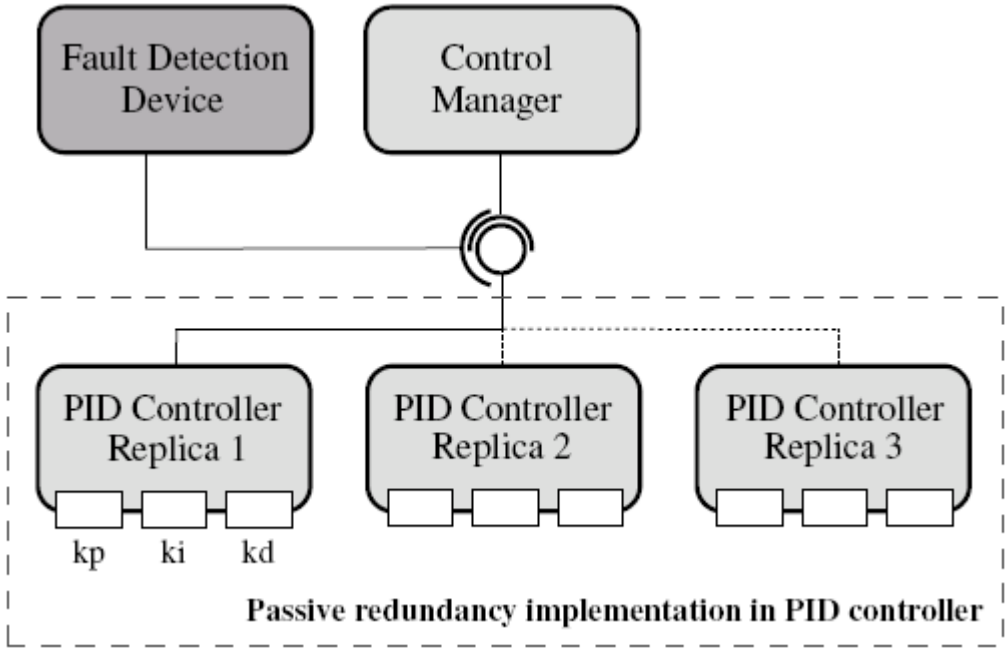
Evaluation Experiments

1 Self-optimization



Evaluation Experiments

2 Self-healing

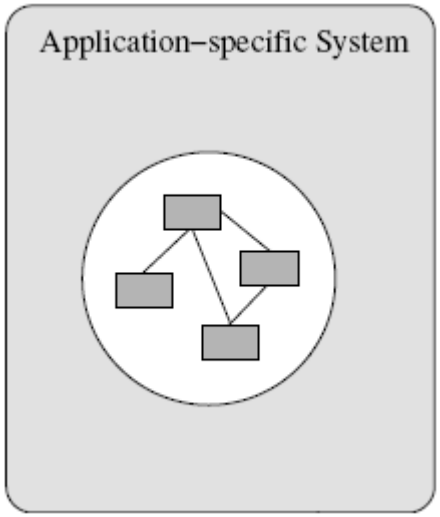


Unanticipated Changes

- 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

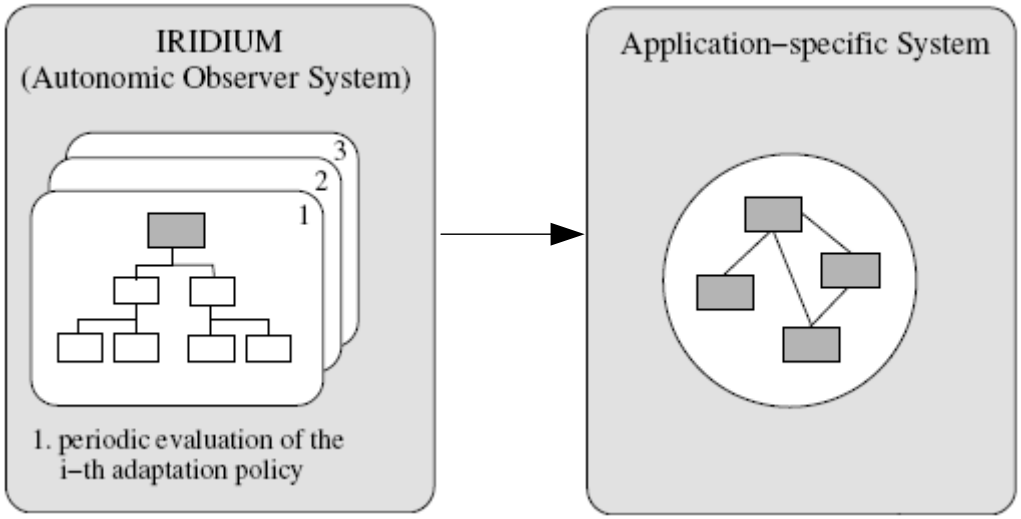
Unanticipated Changes

- Automatically: by deploying another running instance of our framework



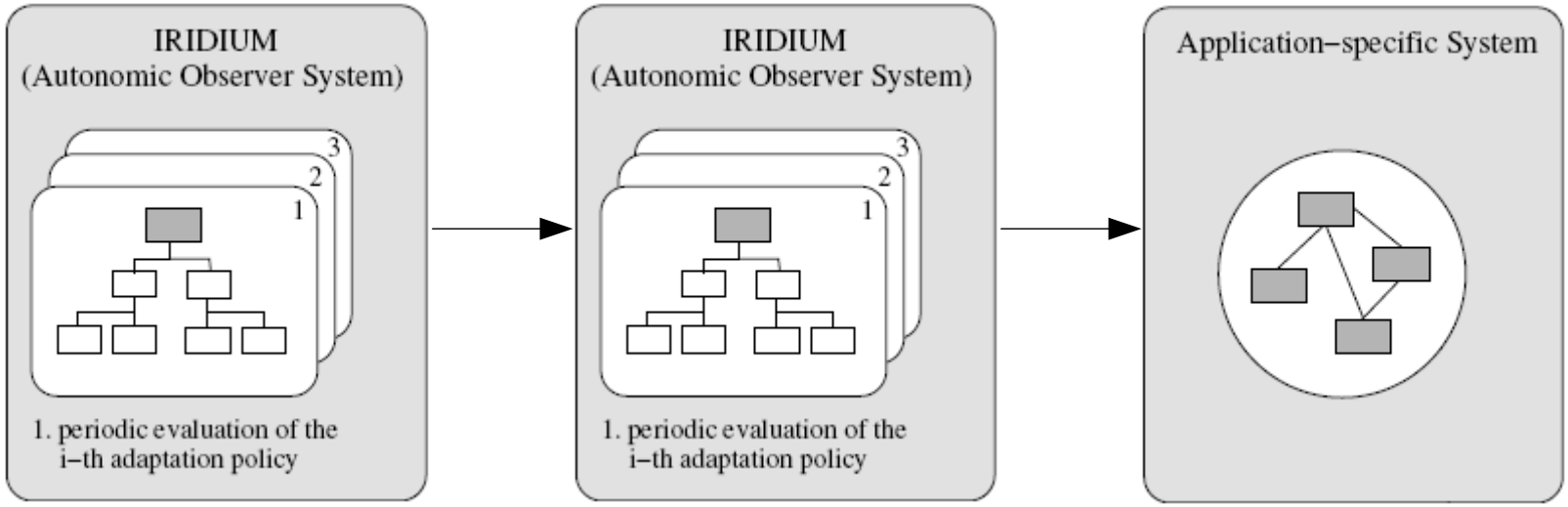
Unanticipated Changes

- 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 ?