Approaches for Adaptive and Dependable Distributed Systems

Raimundo Macêdo,
Distributed System Lab
(LaSiD)/DCC/UFBA Brazil

IFIP Workshop on Dependability of Large-Scale and Dynamic Systems
February 21-25, 2008
Natal, Rio Grande do Norte, Brazil

Aim of the talk

To Present a brief overview on our current work on adaptive and dependable distributed systems

Outline

1. Introduction
2. The HD (hybrid and dynamic) model
3. A reusable framework for control and supervision applications (ARCOS)
4. Dynamic reconfiguration in ARCOS
5. Underlying mechanisms

A primary motivation

To build an experimental platform for distributed industrial supervision and control systems

….using hardware and software COTS

For Mechatronics students ….

Motivation

Requirements for Modern Real-Time Distributed Supervision and Control Systems:

- Distribution
- Flexibility, Reusability, and Extensibility
- Self-Adaptation and Intelligent Algorithms
- Interoperability
- Temporal Predictability and Dependability

To address these issues we need:

- Software-intensive methodologies and solutions to handle the increasing complexity

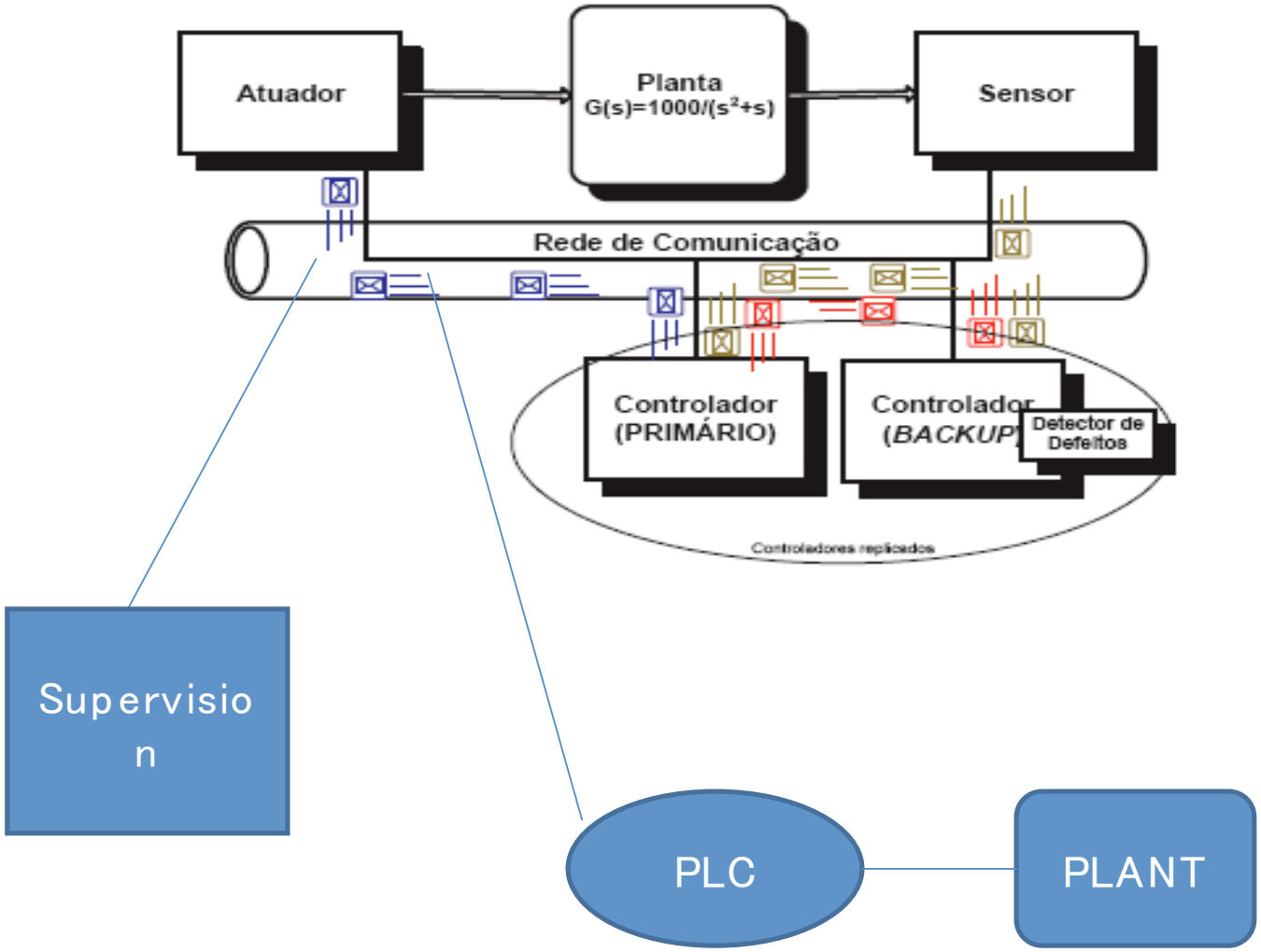
Motivation

Requirements for Modern Real-Time Distributed Supervision and Control Systems:

- Distribution
- Flexibility, Reusability, and Extensibility
- Self-Adaptation and Intelligent Algorithms
- Interoperability
- Temporal Predictability and Dependability

To address these issues we need:

- Software-intensive methodologies and solutions to handle the increasing complexity

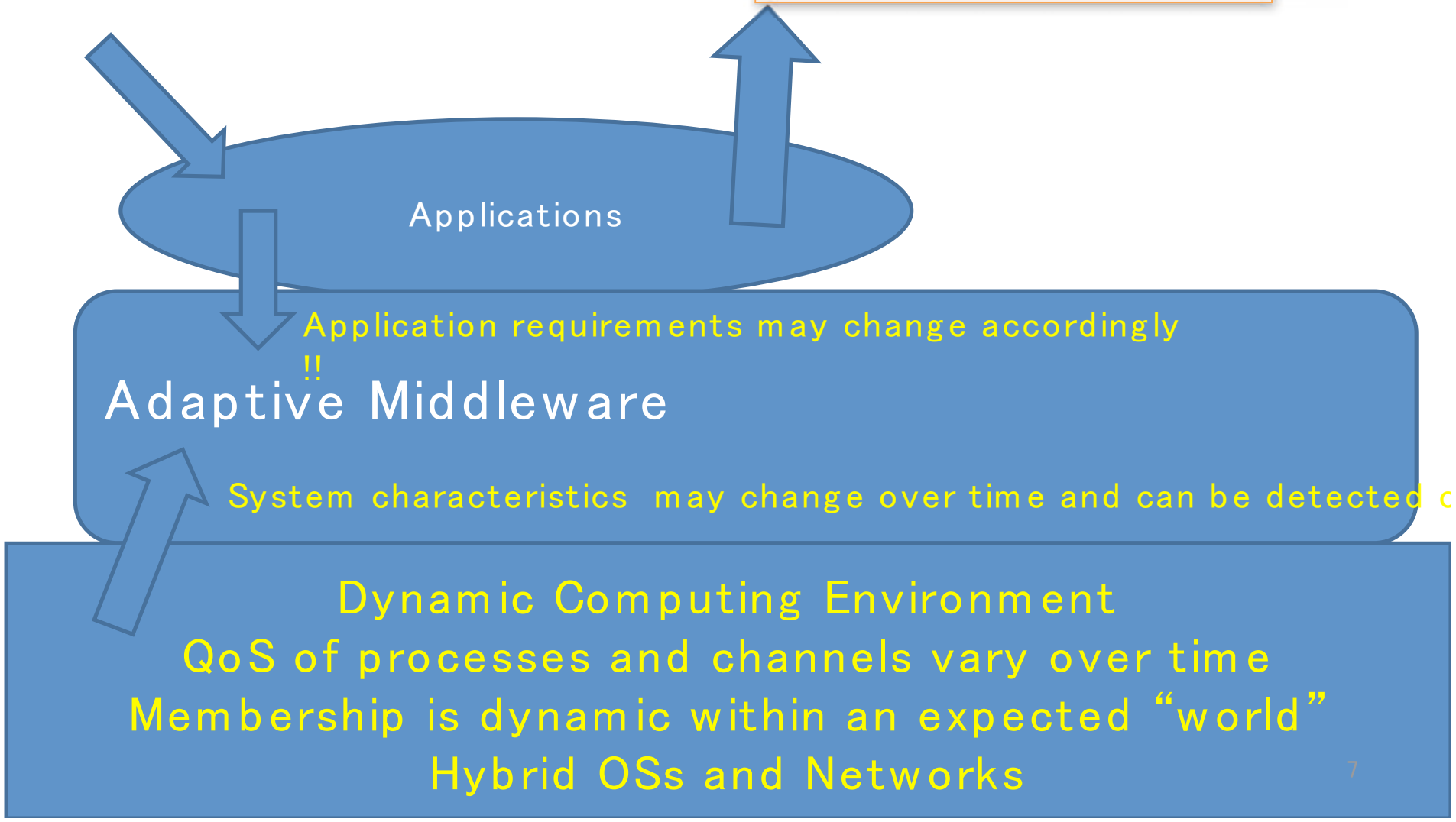


A typical Distributed Real-Time S & C System

What do we mean by a Adaptive and Dependable Distributed

Correct service despite some
- system component failures
- environment changes
- user requirement changes

User requirements may change !!!



How are we approaching this problem?
(talk`s outline)

In the system model level

We assume a hybrid and dynamic model – HD

→ Adaptive algorithms for uniform consensus, GMS, replication …

In the application/middleware level

→A component-based adaptive framework for Real-Time Control and Supervision Applications (ARCOS)

→Dynamic application reconfiguration

→Sensing/monitoring mechanisms for

How are we approaching this problem?

In the system model level

We assume a hybrid and dynamic model – HD

→ Adaptive algorithms for uniform consensus, GMS, replication …

In the application/middleware level

→A component-based adaptive framework for Real-Time Control and Supervision Applications (ARCOS)

→Sensing/monitoring mechanisms for

- adaptive failure detection
- system configuration detection (QoS-awareness of channels and processes)

What we mean by an hybrid and dynamic (HD) model?

DS = (Π, C) – Π is a set of processes is a set C of channels.

Each element of Π (or C) is either synchronous (S) or asynchronous (A)
and its state (i.e., QoS) may change over time ....

So, DS itself may be S, A, or made up of any combination of sub-graphs that can be S, or A …. and the configuration of DS may change over time.

Advantages

- ✓ More General Algorithms

If P is proved correct in HD ➔ P is correct in A, S, or any combination of A and S inside the system

- ✓ Temporal degradation/adaptation is taken into account in the design phase of the algorithms – degraded and fail-safe behavior !!

Some developments

Uniform Consensus on the HD model
(Gorender-Macêdo-Raynal DSN05 and TDSC2007)

Basic Motivation: provide fault tolerance for systems that

- are hybrid with synchronous/asynchronous features
- and may loose QoS (due to failures) or it may be re-negotiated

Uniform Consensus is provided !!

but we still need ◇S or equivalent mechanism as parts of the system (or even the entire system) may be (or become) asynchronous

➔ *live,down,uncertain* + R0…R6 + ◇S

◊S:

Strong completeness: Eventually, every process that crashes is permanently suspected by every correct process

Eventual Weak Accuracy: There is a time after which some correct process is not suspected by correct processes

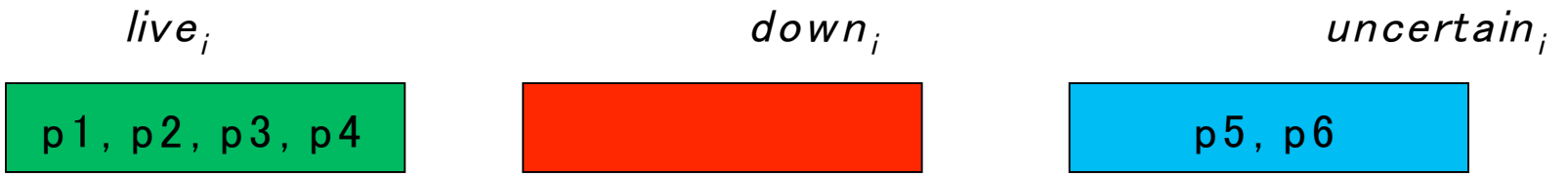
Applying HD to Consensus
The (abstract) hybrid and adaptive programming m

Consider $\Pi = (p_1, \cdots, p_n)$
Each p_i maintains the local sets: $live_i$, $uncertain_i$, and $down_i$ (composed from Π)

p_i in live: it is a hint that p_i is functioning (pi belongs to a **S** sub-graph)

p_i in down: p_i is crashed

p_i in uncertain: no idea about the state of p_i (pi belongs to a **A** sub-graph)



(application) processes can only read the sets, and may have distinct views of them at a given time !

Modifications on the sets are regulated by the rules R0-R6

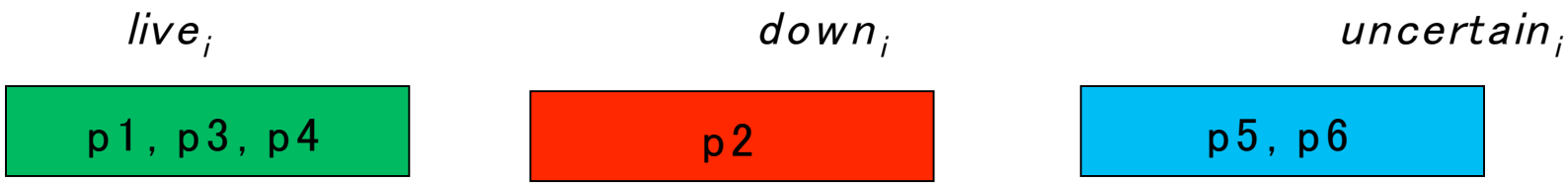
The (abstract) hybrid and adaptive programming m

Consider $\Pi = (p_1, \cdots, p_n)$
Each p_i maintains the local sets: $live_i$, $uncertain_i$, and $down_i$ (composed from Π)

p_i in live: it is a hint that p_i is functioning (pi belongs to a S sub-graph)

p_i in down: p_i is crashed

p_i in uncertain: no idea about the state of p_i (pi belongs to a A sub-graph)



(application) processes have can only read the sets, and may have distinct vi
of them at a given time instant!

Modifications on the sets are regulated by the rules R0-R6

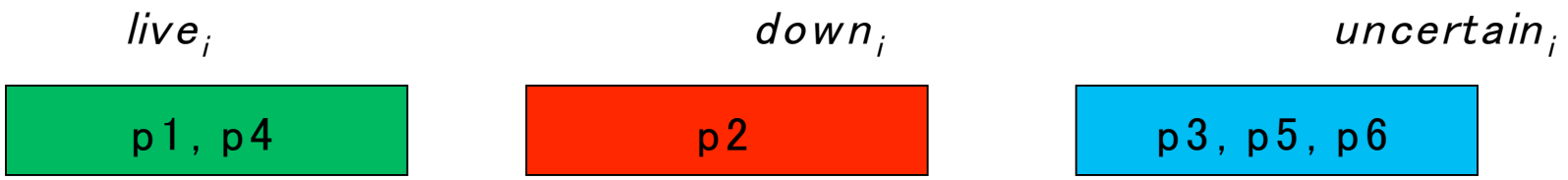
The (abstract) hybrid and adaptive programming m

Consider $\Pi = (p_1, \cdots, p_n)$
Each p_i maintains the local sets: $live_i$, $uncertain_i$, and $down_i$ (composed from Π)

p_i in live: it is a hint that p_i is functioning (pi belongs to a S sub-graph)

p_i in down: p_i is crashed

p_i in uncertain: no idea about the state of p_i (pi belongs to a A sub-graph)



(application) processes have can only read the sets, and may have distinct vi
of them at a given time instant!

Modifications on the sets are regulated by the rules R0-R6, which must be res
Implementation of the model

For details about the rules and proofs, see the papers (DSN05, TDSC 2007)

The consensus algorithm

An adaptation of the ◊S circulating coordinator round-based algorithm (decentralized communication version)

1st phase

From coordinator Pc to all, send phase1(estc)

members wait until either received phase1(estc) or pc ∈ (downi ∪ suspecti)

live = Π(synchronous)

uncertain = Π(asynchronous)

2nd phase

Send to all phase2(union of received phase1 (estc or null))

wait until received phase2

(1) from all in livei, and

(2) majority in uncertaini

Pay-offs

✓Resilience of the system depends on the actual QoS provided

(as it is the case for any dynamic dist system)

$f \leq |live| + ((|uncertain| - 1) / 2)$

✓We can explore the model in order to improve fault-tolerant services.

For instance, by choosing only processes in *life* to be the coordinator of a round we avoid false suspicions that could delay consensus termination

✓If QoS cannot be guaranteed for some reason (e.g., failure of components), the system becomes totally asynchronous (uncertain = Π) and safety is sustained (indulgent) - no need to change the consensus algorithm

How are we approaching this problem?

In the system model level

We assume a hybrid and dynamic model – HD

→ Adaptive algorithms for uniform consensus, GMS, replication …

In the application/middleware level

→A component-based adaptive framework for Real-Time Control and Supervision Applications (ARCOS)

→Dynamic application reconfiguration

→Sensing/monitoring mechanisms for

- adaptive failure detection

Engineering Components for Flexible and Interoperable Real-Time Distributed Supervision and Control Systems

Sandro S. Andrade and Raimundo J. de A. Macêdo

Distributed Systems Laboratory (LaSiD)
Department of Computer Science (DCC)
Post-Graduation Program on Mechatronics
Federal University of Bahia - Brazil

September, 2007
12th IEEE International Conference on Emerging Technologies and Factory Automation
Patras - Greece

Motivation

Requirements for Modern Real-Time Distributed Supervision and Control Systems:

- Distribution
- Flexibility, Reusability, and Extensibility
- Self-Adaptation and Intelligent Algorithms
- Interoperability
- Temporal Predictability and Dependability

To address these issues we need:

- Software-intensive methodologies and solutions to handle the increasing complexity

Motivation

Requirements for Modern Real-Time Distributed Supervision and Control Systems:

- Distribution
- Flexibility, Reusability, and Extensibility
- Self-Adaptation and Intelligent Algorithms
- Interoperability
- Temporal Predictability and Dependability

To address these issues we need:

- Software-intensive methodologies and solutions to handle the increasing complexity

Software Engineering and Mechatronics

Current trends include the use of:

- Hardware and software COTS (Commercial Off-The-Shelf) components
- Object- and/or aspect-oriented programming paradigms
- Component-oriented middleware solutions for distribution, reusability, and flexibility
- Architectural models, application frameworks, design patterns, and software product lines

In particular:

- Distributed components is being used in real-time systems development
  - TAO, CIAO, Cadena, RCCF, ACCORD
- Application frameworks $\longrightarrow$ reusability, productivity and quality

Software Engineering and Mechatronics

Current trends include the use of:

- Hardware and software COTS (Commercial Off-The-Shelf) components
- Object- and/or aspect-oriented programming paradigms
- Component-oriented middleware solutions for distribution, reusability, and flexibility
- Architectural models, application frameworks, design patterns, and software product lines

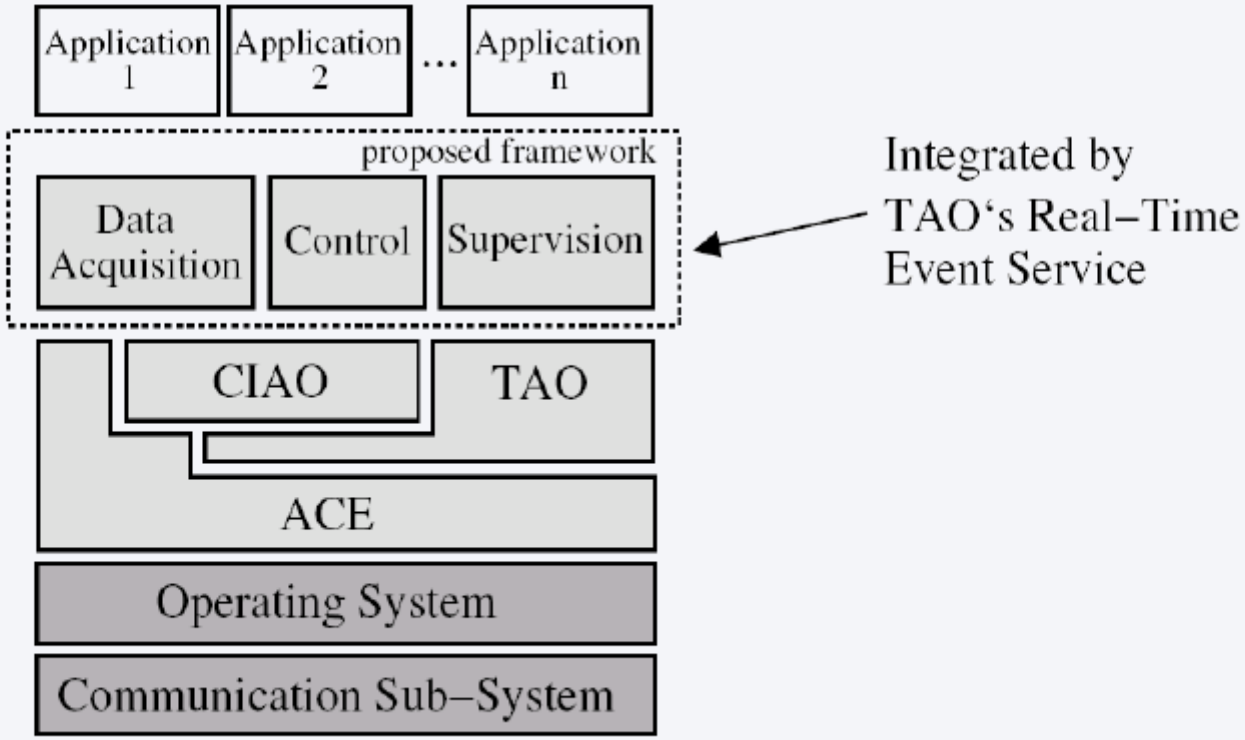
In particular:

- Distributed components is being used in real-time systems development
  - TAO, CIAO, Cadena, RCCF, ACCORD
- Application frameworks $\rightarrow$ reusability, productivity and quality

ARCOS: Architecture for Control and Supervision

ARCOS

Component-based application framework for the construction of real-time supervision and control systems



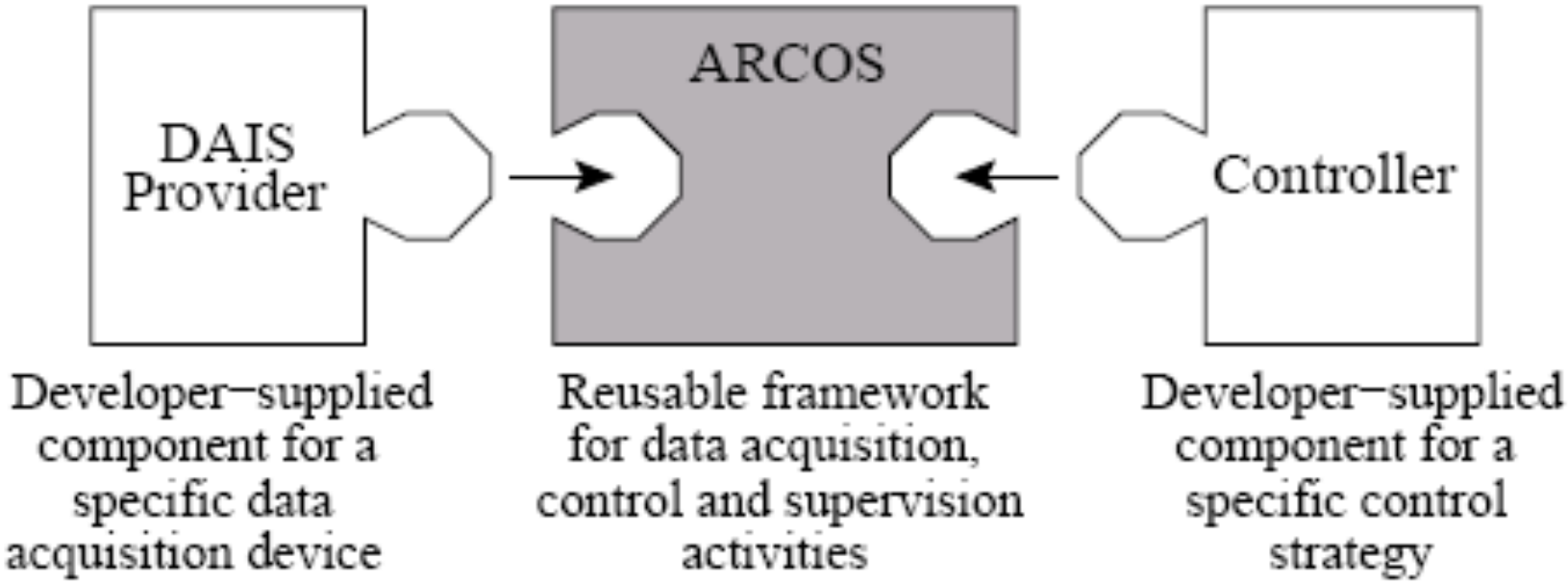


Figure 1. The reusable framework provided by ARCOS and the required components for data acquisition and control.

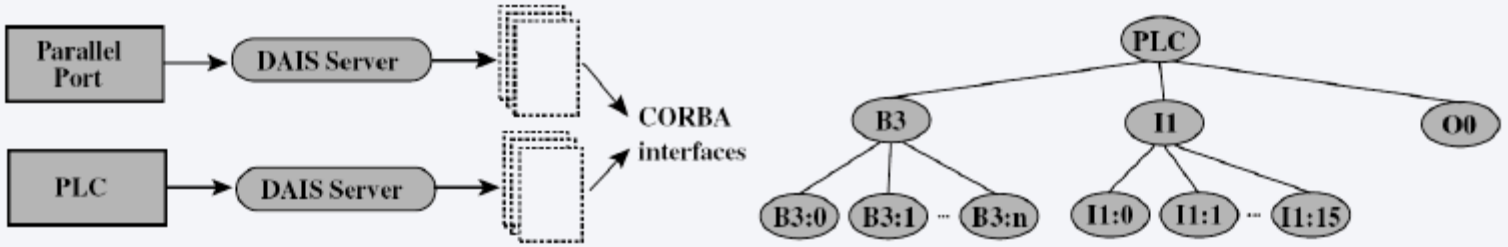
Data Acquisition

- Implements the DAIS (Data Acquisition from Industrial Systems) OMG specification

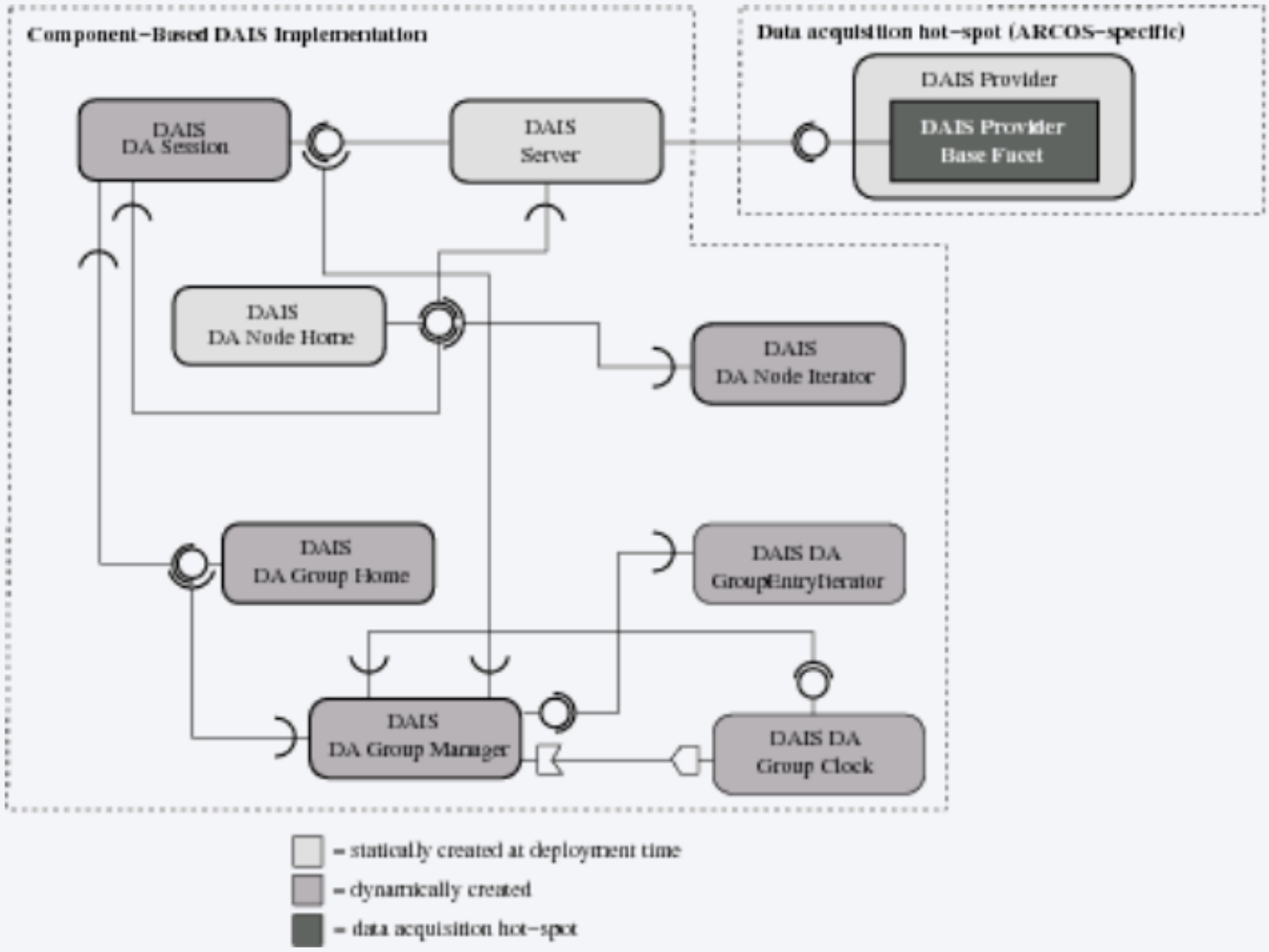
For browsing, collecting, and updating data from distinct devices

DAIS

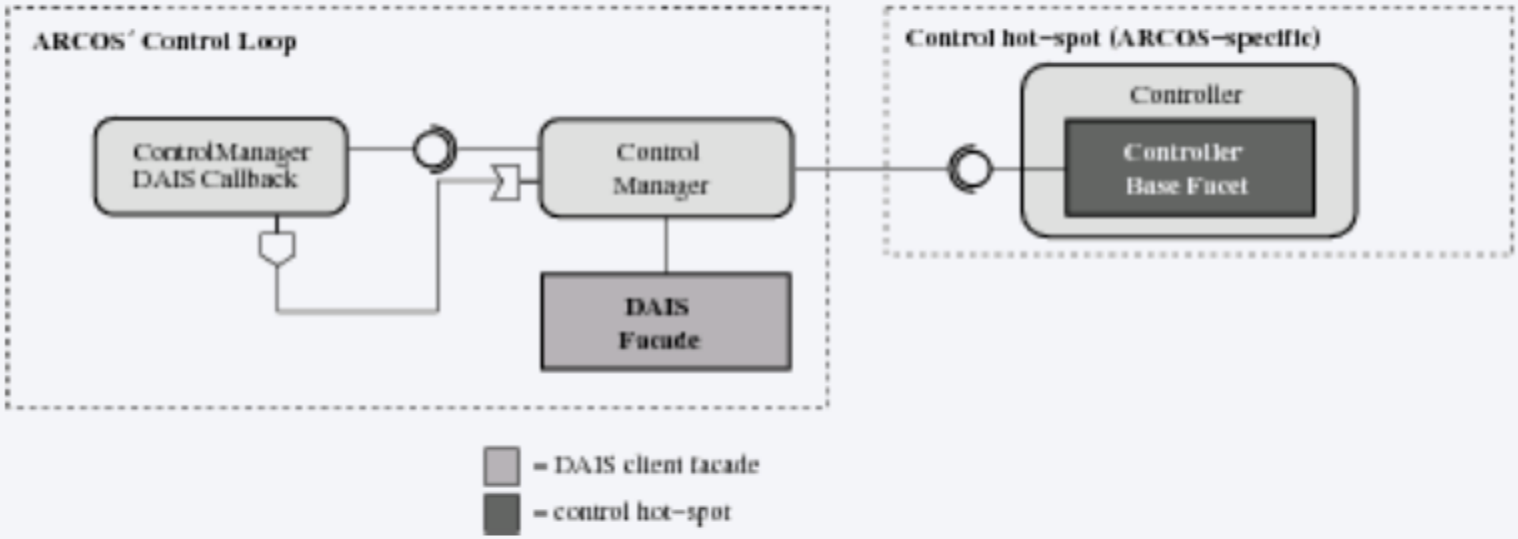
The DAIS specification maps data acquired from industrial devices in standardized CORBA 2.x interfaces



Data Acquisition Components

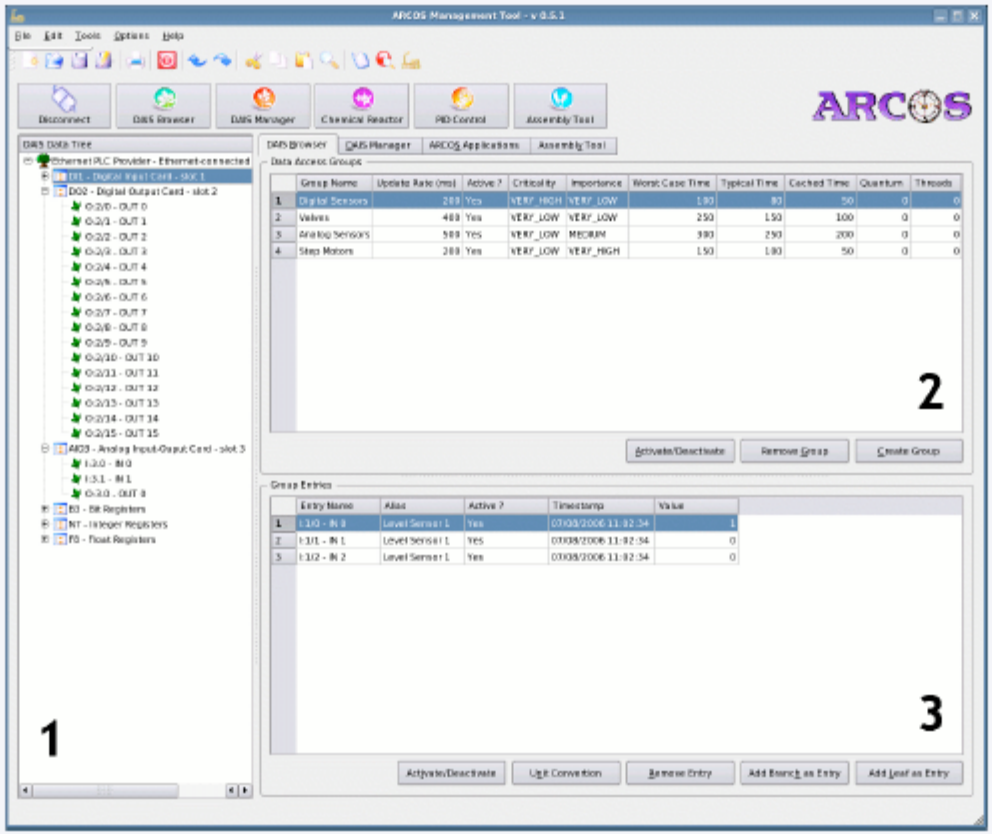


Control Components



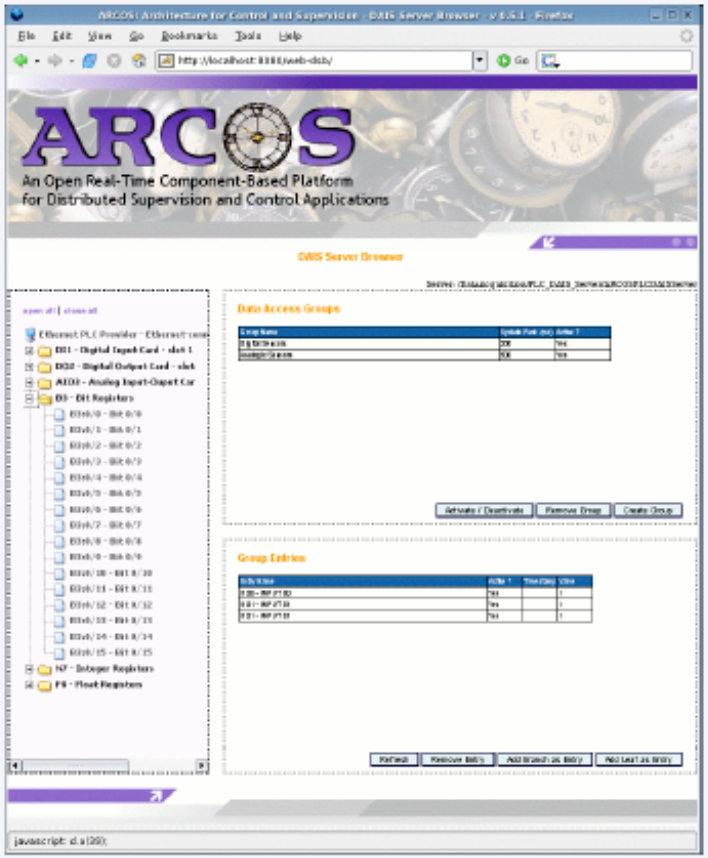
Supervision Applications

- DAIS Server Browser: desktop version



Supervision Applications

- DAIS Server Browser: web version

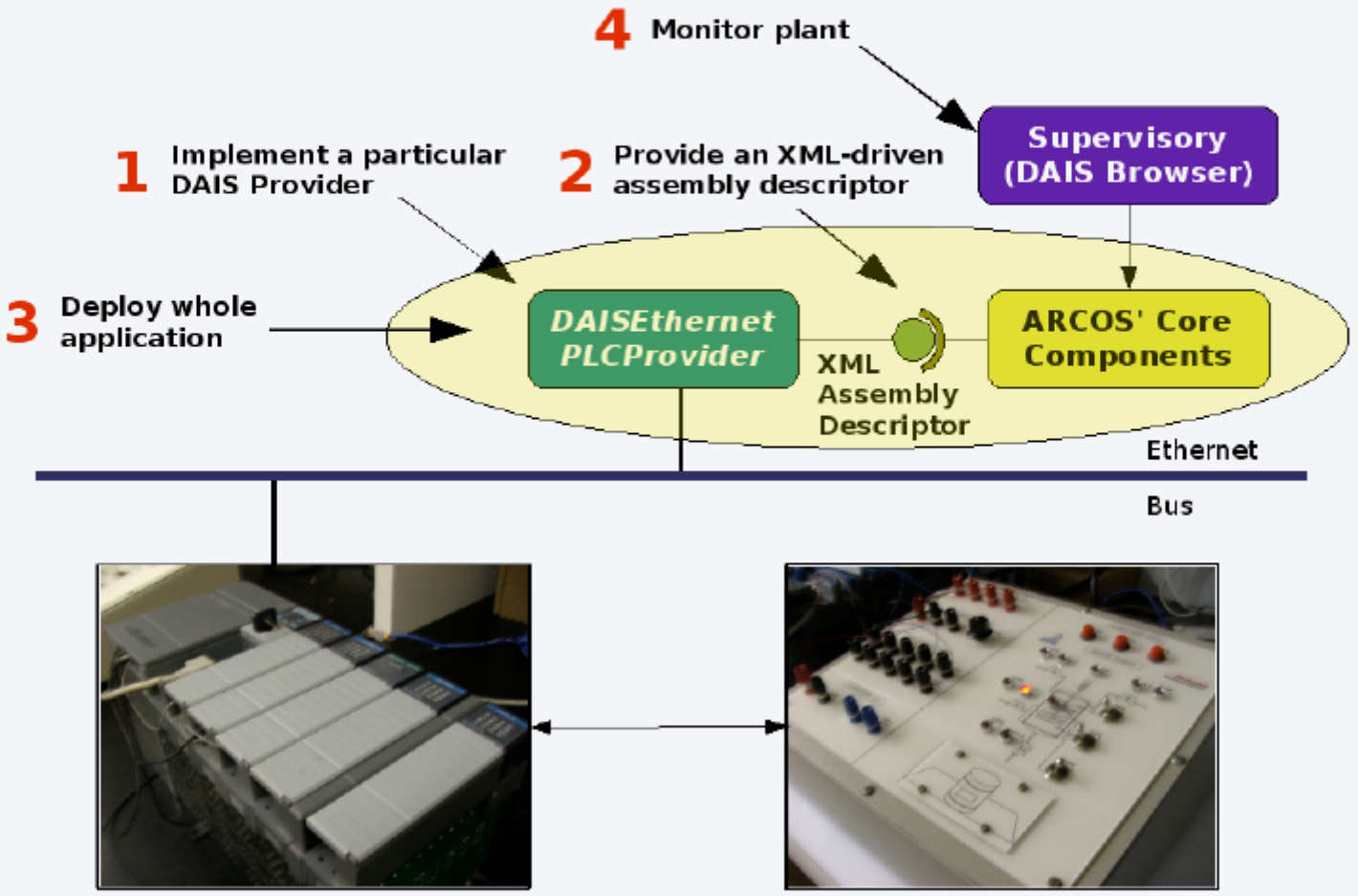


Developing ARCOS-Based Industrial Applications

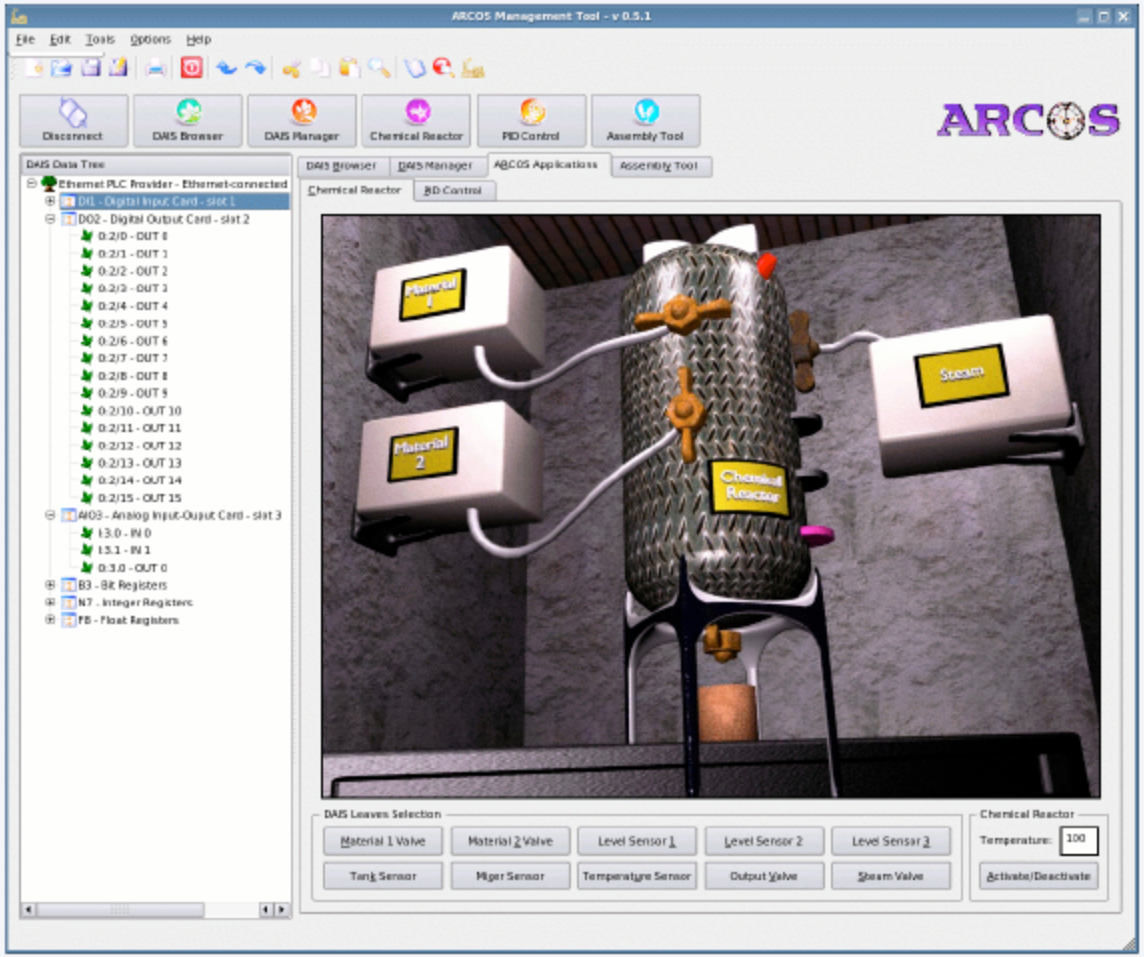
Required Steps:

- Implement or reuse particular DAIS Providers for data acquisition devices
- Implement or reuse particular control components
- Connect these components into the ARCOS' core architecture using a XML descriptor file
- Deploy whole application
- Monitor plant using the DAIS Server Browser or a specific supervisory

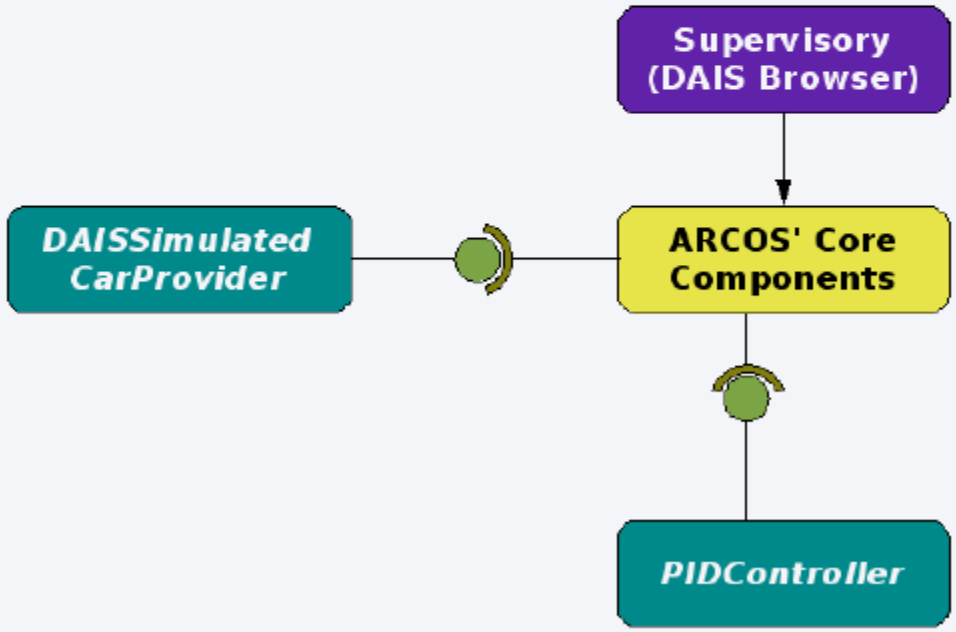
Experiment 1: Chemical Reactor Supervision



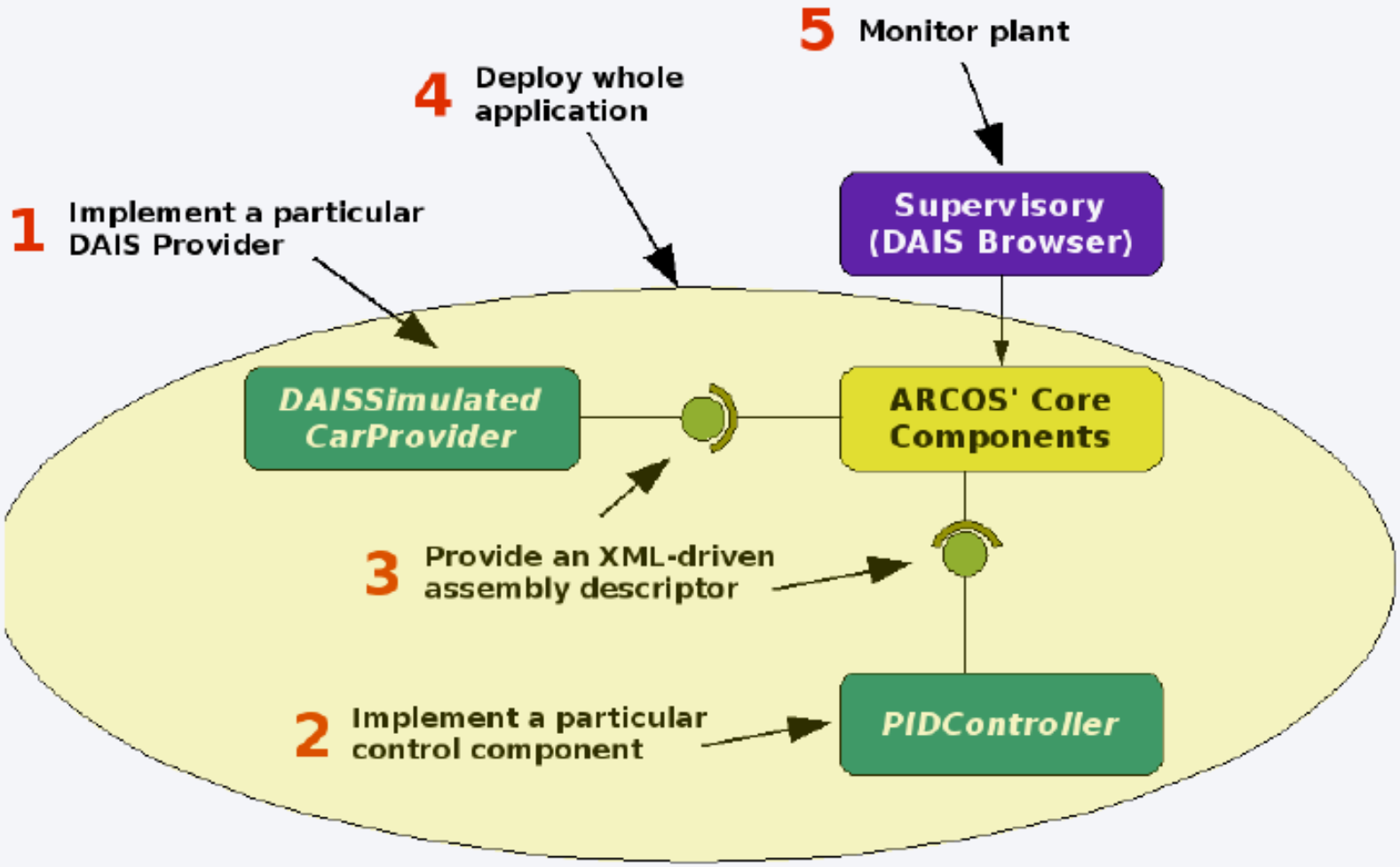
Experiment 1: Chemical Reactor Supervision



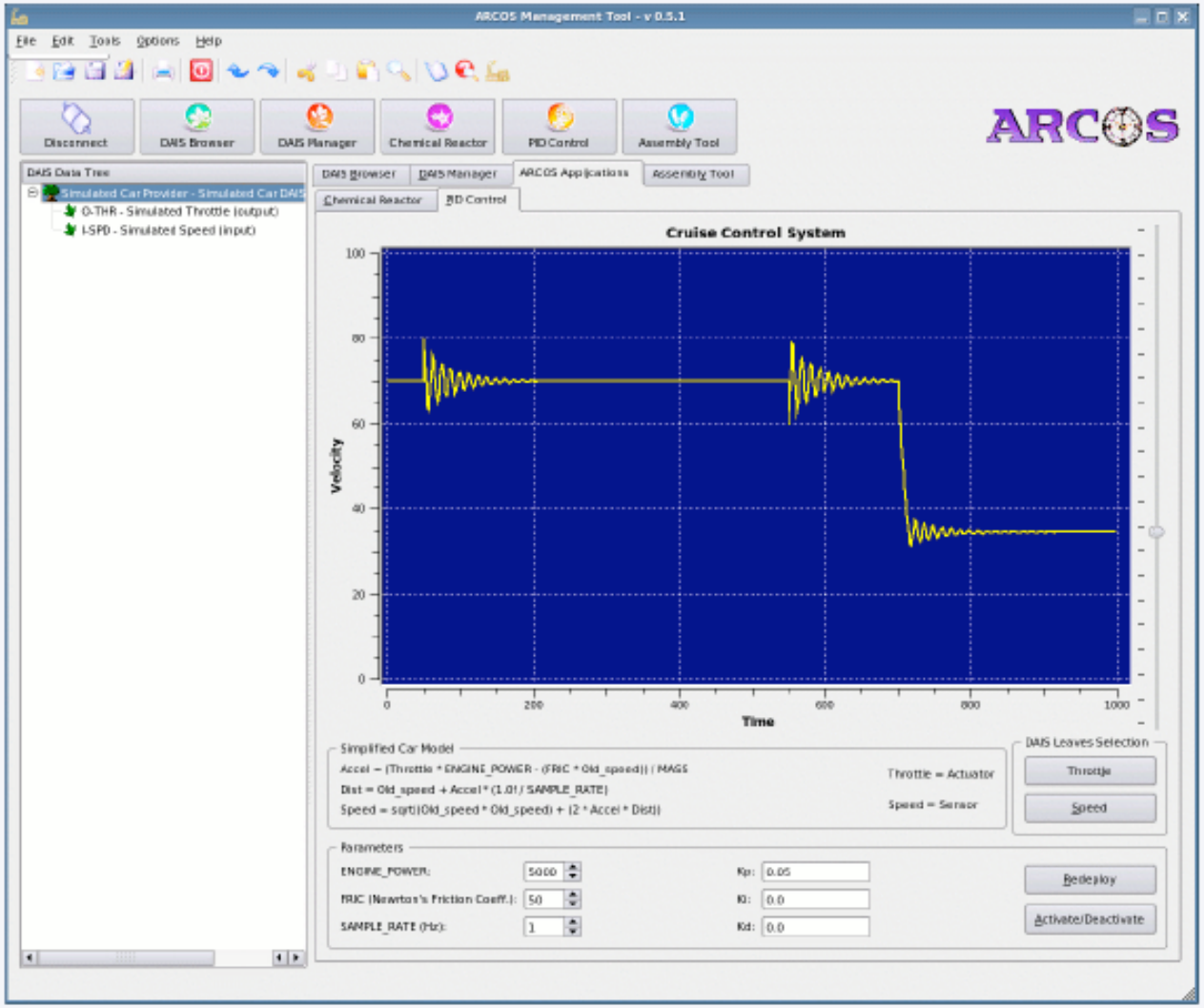
Experiment 2: Cruise Control System



Experiment 2: Cruise Control System



Experiment 2: Cruise Control System



Implementation Issues

- Underlying technologies: ACE (ADAPTIVE Communication Environment), TAO (The ACE ORB), and CIAO (Component-Integrated ACE ORB)
- Implemented on the GNU/Linux platform, using the Standard C++ programming language
- 15,000 lines of source-code implemented (data acquisition - 8,000, control - 2,000, and supervision - 5,000)
- Footprint: 6Mb (client) and 10Mb (server)

How are we approaching this problem?

In the system model level

We assume a hybrid and dynamic model – HD

→ Adaptive algorithms for uniform consensus, GMS, replication …

In the application/middleware level

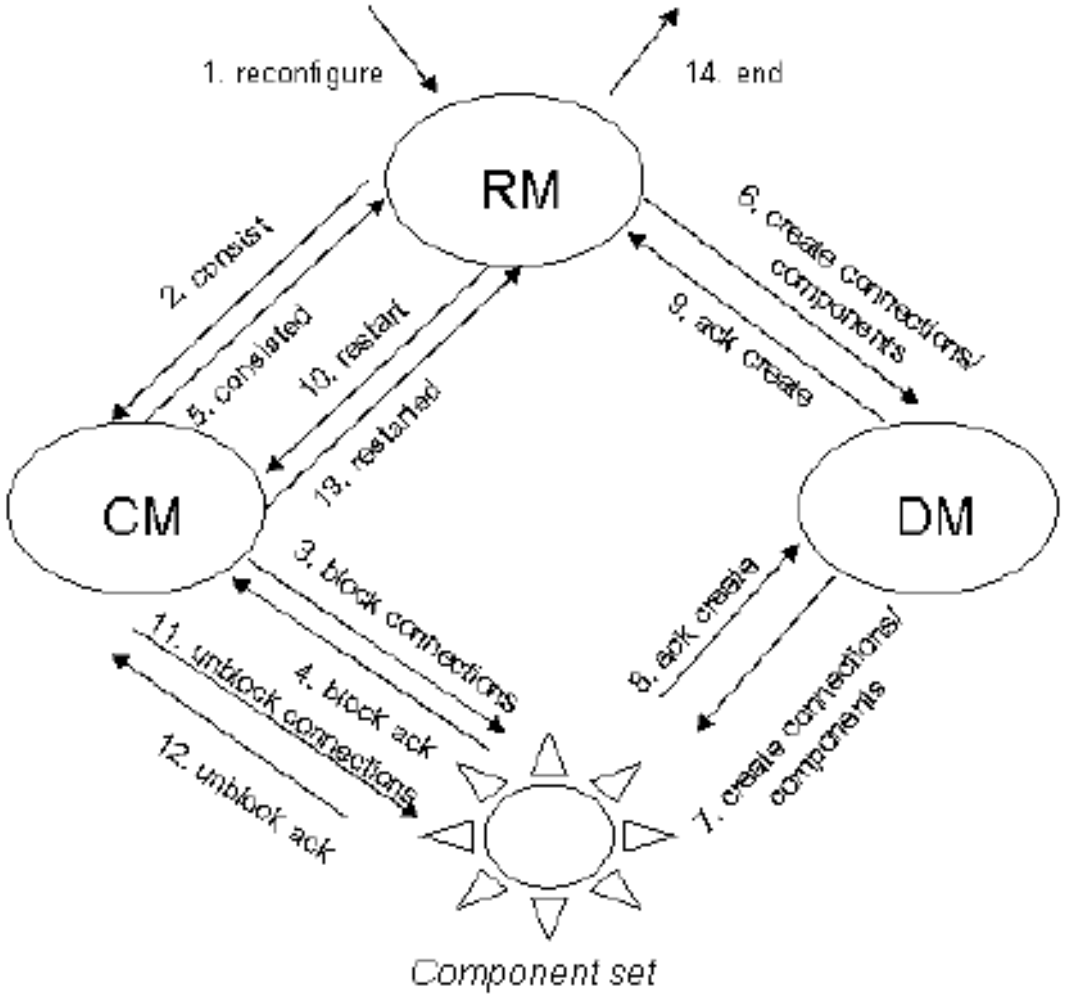
→A component-based adaptive framework for Real-Time Control and Supervision Applications (ARCOS)

→Dynamic application reconfiguration

→Sensing/monitoring mechanisms for

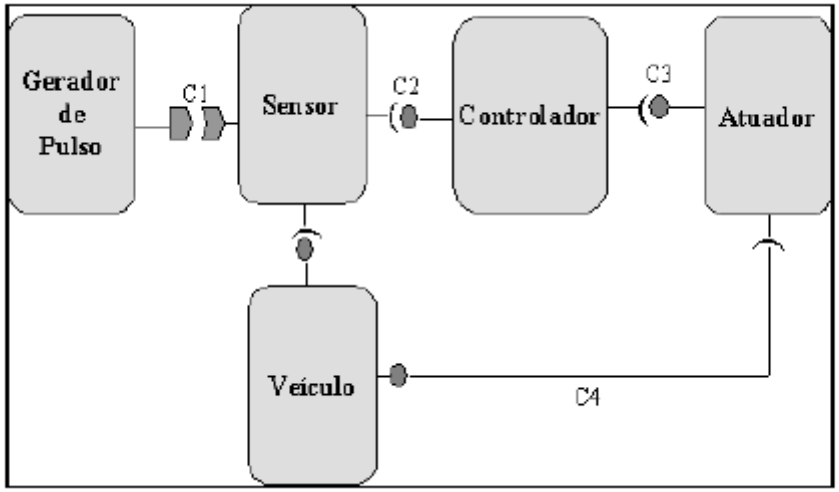
- adaptive failure detection

Reconfiguration Manager (RM) - *coordinates the reconfiguration process*
Consistency Manager (CM) - *Implements policies for system consistency*
Deployment Manager (DM) - Creates and destroy components and connections

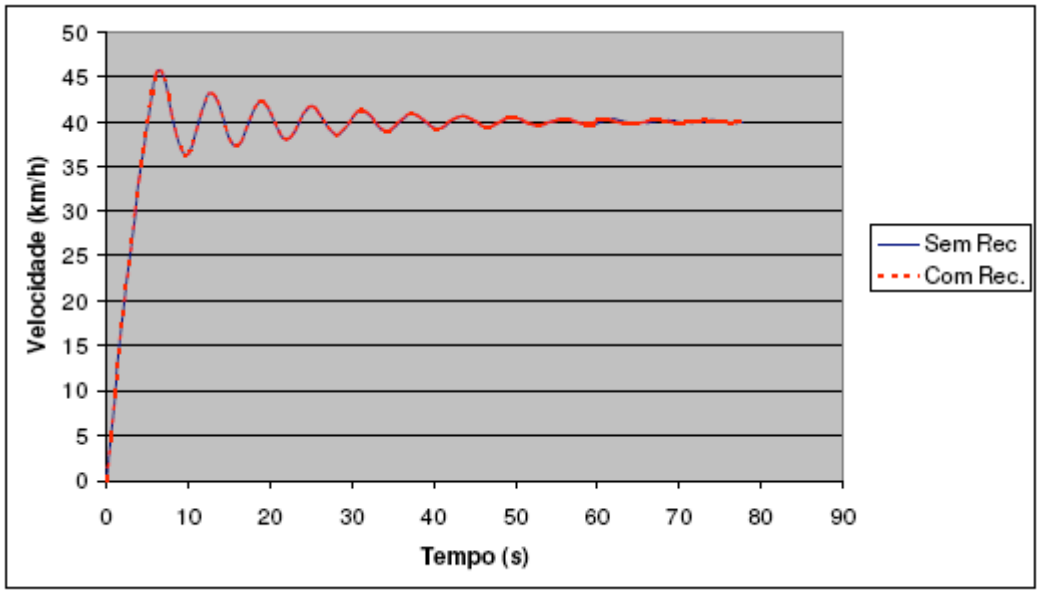


A General View of the Reconfiguration Process

Example: replacement of a faulty controller.




```
01 (Controlador2.failed) => {
02    destroy connection C1, C2
03    create new Controlador3 from Controlador2 on NodeA
04    create connection C3 : Controlador3.f1 -> Sensor.r1
05    create connection C4 : Controlador3.f1 ->  Atuador.r1  }
```



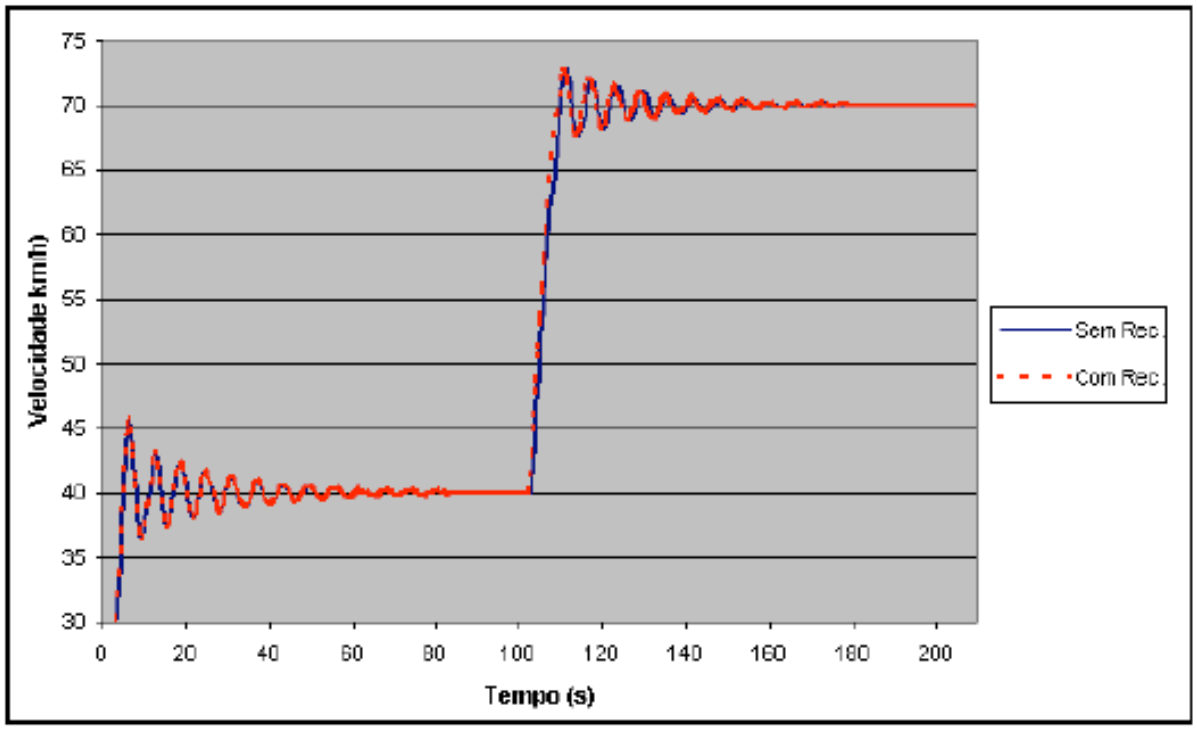


Figura 7. Evolução do Controlador PID com alteração da velocidade de 40km/h para 70km/h aos 100 segundos

How are we approaching this problem?

In the system model level

We assume a hybrid and dynamic model – HD

→ Adaptive algorithms for uniform consensus, GMS, replication …

In the application/middleware level

→A component-based adaptive framework for Real-Time Control and Supervision Applications (ARCOS)
→Dynamic application reconfiguration

→Sensing/monitoring mechanisms for
- adaptive failure detection
- system configuration detection (QoS-awareness of

What is required to implement the model and how to implem

What are the underlying mechanisms?

1) end-to-end QoS management (resource reservation, QoS monitoring,

An interface with the underlying QoS system has been defined and implemented

The QoS Provider → CreateChannel(), DefineQoS(), Delay(), and QoS()

2) Additional mechanisms that provide the sets with appropriate semantics (R0-R6) have been developed and implemented

→Failure detector
→State Detector

3) Adaptive failure detectors have been developed and implemented

4) Integration with a hybrid networks and op. systems in underway

Final words

The approach has been validated for a few simulated applications

But !!

It remains to be tested in real applications

And it still not completed integrated with a hybrid environment (SO and Network)

Future work include autonomic behavior (self*) and integration with open and wireless environment

References – see www.lasid.ufba.br

1. **An Adaptive Programming Model for Fault-Tolerant Distributed Computing**
Sérgio Gorender, Raimundo José de Araújo Macêdo, Michel Raynal
In IEEE Transactions on Dependable and Secure Computing Vol. 4, no. 1, pp. 18-31, January

2..**An Integrated Group Communication Infrastructure for Hybrid Real-Time Distributed Syste**
Raimundo José de Araújo Macêdo
In 9th Workshop on Real-Time Systems (WTR 2007), p.81-88
Belém, Brazil, May 2007.

3. **Um Framework para Prototipagem e Simulação de Detectores de Defeitos na Construção Sistemas de Tempo Real**
Alírio Sá, Raimundo José de Araújo Macêdo
In Proceedings of VIII Workshop de Teste e Tolerância a Falhas (WTF 2007), p.43-56
Belém, Brazil, May 2007.

4.**Engineering Components for Flexible and Interoperable Real-Time Distributed Supervision and Control Systems**
Sandro Andrade, Raimundo Macêdo
In 12th IEEE Conference on Emerging Technologies and Factory Automation
Greece , September 2007 .

5. **Avaliando o Impacto de Detectores de Defeitos na Estabilidade de Sistemas de Controle c Real sobre Redes Convencionais.**
Alírio Santos de Sá, Raimundo José de Araújo Macêdo
In WTF 2006 - VII Workshop de Testes e Tolerância à Falhas. pp. 55-66
Curitiba/Brasil, May 2006.

6, A Management Tool for Component-Based Real-Time Supervision and Control Systems.
Sandro Santos Andrade, Raimundo José de Araújo Macêdo.
In Simpósio Brasileiro de Engenharia de Software (XIII Sessão de Ferramentas)v. 1. p. 139-144
Florianópolis, October 2006.

7. **Real-Time Component Software For Flexible And Interoperable Automation Systems.**
Sandro Santos Andrade, Raimundo José de Araújo Macêdo
In Anais do XII Congresso Brasileiro de Automática (XII CBA) v. 1. p. 3014-3019
Salvador, 2006.

8. **Detectores de Defeitos Adaptáveis para Sistemas de Controle Distribuídos de Tempo-Real sobre Redes Ethernet**
Alírio Santos de Sá e Raimundo José de Araújo Macêdo
In VII Brazilian Workshop of Real-Time Systems,
Fortaleza, Brazil, May 2005.

9. A Hybrid and Adaptive Model for Fault-Tolerant Distributed Computing
Sergio Gorender, Raimundo J. Araújo Macêdo, Michel Raynal
In Proceedings of IEEE/IFIP Int. Conference on Computer Systemas and Networks (DNS05).p. 412-42.
Yokohama, Japan, June 2005.

10. **The Implementation of a Distributed System Model for Fault Tolerance with QoS.**
Raimundo José de Araújo Macêndo, Sérgio Gorender and Paulo Cunha
In Anais do Simposio Brasileiro de Redes de Computadores (SBRC 2005). p. 827-840
Fortaleza, May 2005.

11. **An Adaptive Failure Detection Approuch For Real-time Distributed Control Systems Over Shared Ethernet**
Alírio Sá, Raimundo José de Araújo Macêdo
In Proceedings of the 18th International Congress on Mechanical Engeneering - Mechatronics Symposium,
Ouro Preto - Brasil, 2005.

12, **A Component-Based Real-Time Architecture for Distributed Supervision and Control Applications**
Sandro Andrade and Raimundo José de Araújo Macêdo
In Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA2005) p. 15-22.
Catania - New York, 2005.

13. **Adapting Failure Detectors to Communication Network Load Fluctuations Using SNMP and Artificial Neural Nets**
Fábio Ramon de Lima e Lima, Raimundo José de Araújo Macêdo
In Second Latin-American Symposium on Dependable Computing (LADC2005), Lecture Notes in Computer Science, Volume 3747, Pages 191 - 205.
Salvador, Bahia, Brasil, October 2005.

Thanks !!!

Obrigado !!!