

Using UML-based Framework to Integrate Real-Time Object-Oriented Programming Models

Carlos Eduardo Pereira e Leandro Buss Becker

cpereira@delet.ufrgs.br

lbecker@inf.ufrgs.br

Electrical Engineering Department

Computer Science Institute

Universidade Federal do Rio Grande do Sul - UFRGS

Brazil



Agenda

- ▶ Historical Overview
 - ▶ OO Requirements Specification and Validation
 - ▶ Methodology, AO/C++
 - ▶ SIMOO, SIMOO-RT
- ▶ Current Work
 - ▶ SIMOO/RT-Deployment Diagram
 - ▶ Integrated Scheduling, QoS
 - ▶ HW-Coprocessors
- ▶ Conclusions



Historical Perspective

- Research work has started on 1990: PhD thesys
 - Context: few OOA methods (Coad/Yourdon, Shlaer/Mellor, OMT, ...), no UMLS
- Main goal:
 - how to apply OO to the development of distributed real-time systems (DRTS) with special emphasys for industrial automation systems



Historical Perspective

- Results of a state-of-the-art analysis in 1991:
 - OO is very suitable for modeling DRTS
 - Semantic mapping of automation equipments
 - Reuse/Inheritance/Extensions
 - Major Drawbacks:
 - Timing requirements specification
 - Requirements validation and verification
 - CASE tool support
 - Runtime support for distributed objects



Historical Perspective

⇒ Proposal made in 1992 (presented at OOPSLA -
Workshop on RT Analysis and Design Methods):

Notation for describing timing requirements

Based on RTL

Tradeoff: understandable vs. verifiable

Periodical activities, synchronization, end-
to-end requirements (between events)



Historical Perspective

- Research work during 92-94:
 - Requirements verification (use of Constraint Propagation algorithms - CobaltBlue)
 - Requirements validation through simulation
 - RT-extension to C++ (AO-C++)



Historical Perspective

AO/C++

Main idea: mapping of OO "logical concurrency" to "physical concurrency" in RT-UNIX (QNX)

Use of **keywords** (keep it as simpler as possible)



```

active class Sensor {
private:
    Conveior REF theConveyor; //references to other objects
    Arm REF theArm;
public:
    void DetectWorkPiece(cycle_t){ //Time-triggered Method
        // initialization code
        begin_cycle
            // cyclic operation
        end_cycle
    }
    void StoreWorkPiece(dead_1){ //Method with deadline
        // exception code
        begin_exception
            // exception code
        end_exception
    } };

```



AO/C++ code example

Historical Perspective

AO/C++

Parser translates .ph .pC files into:

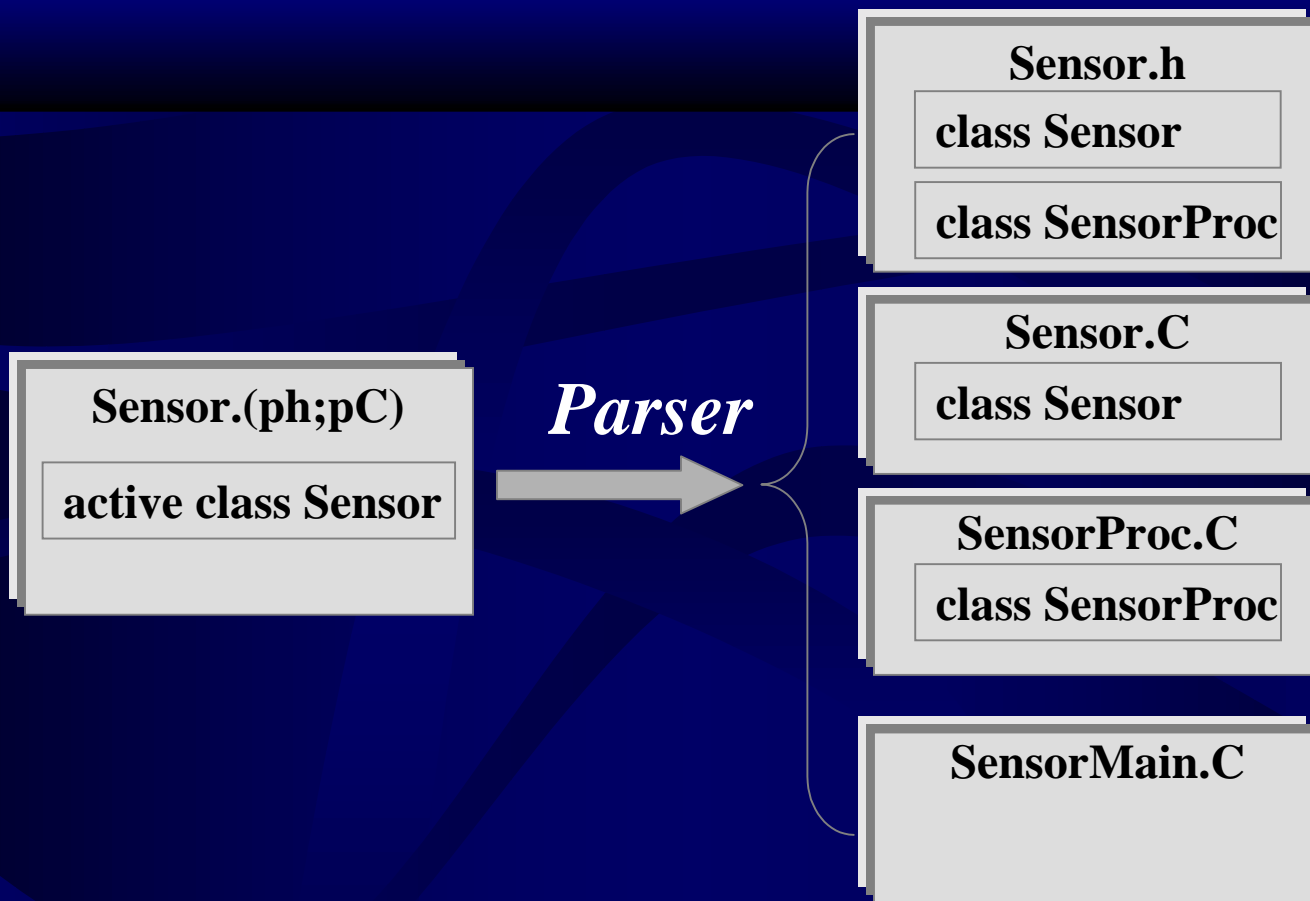
Skeleton, stubs, ...

=> similarities with CORBA IDL <=

Papers: QNX93 and ACM SIGPLAN Workshop on Algorithms and Workshop on Languages, Compiler, and Tool Support for Real-Time Systems.



Pre-processing transformations:



Historical Perspective

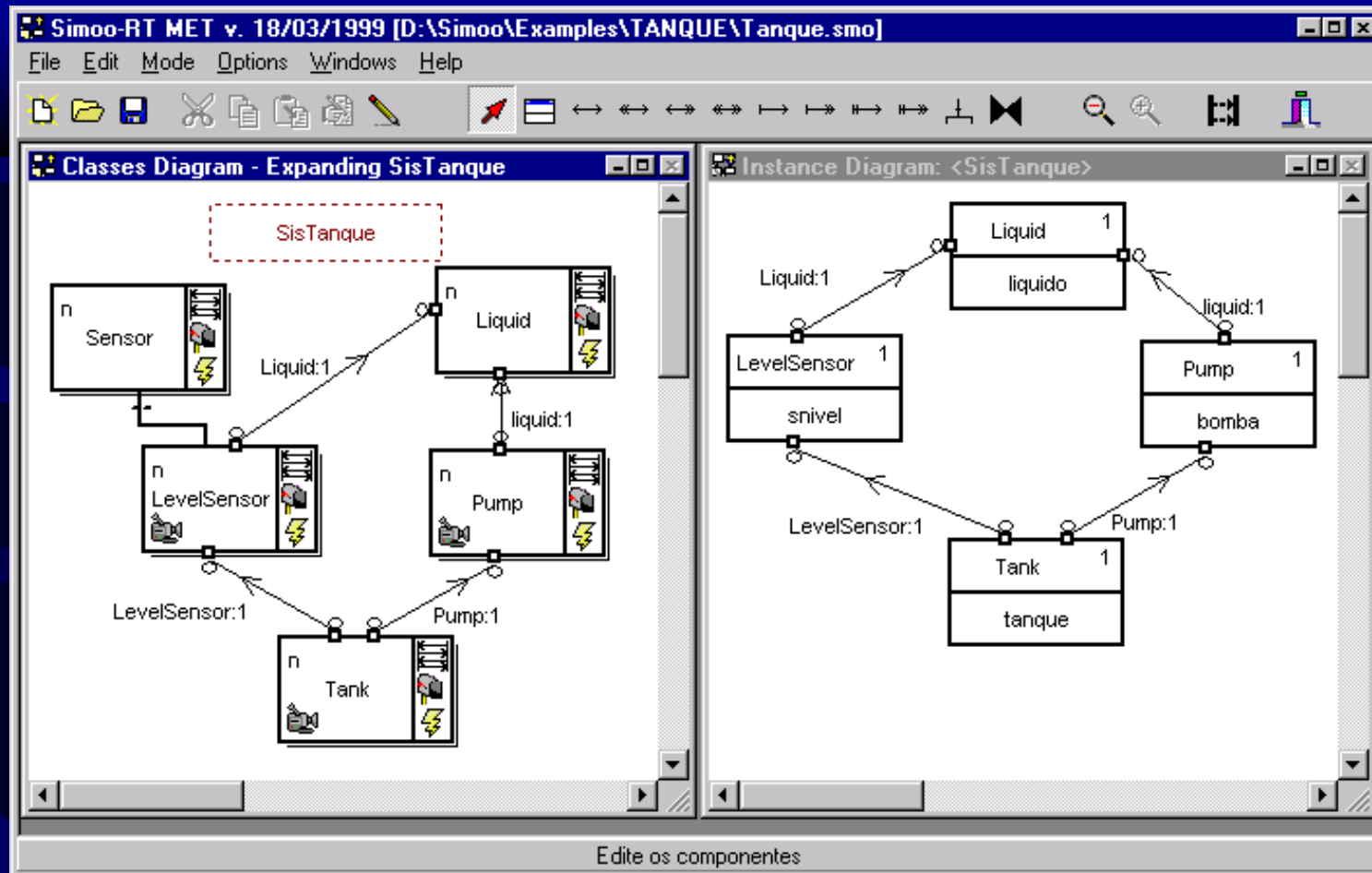
1996-1997: SIMOO Environment

Object-Oriented modeling tool

Simulation tool



Structure modeling: Classes and Instances diagram



SIMOO Model Editing Tool (MET)



Simulation/animation output:

SIMOO

File Simulation View Insert Remove Logs Help

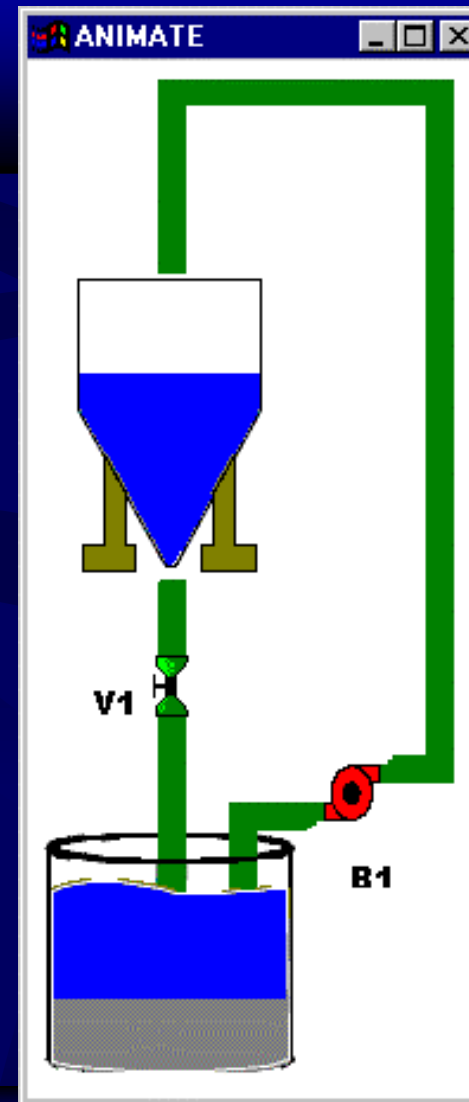
Sensor: Li ValorBruto
PULSO
EstadoBomba: 1
EstadoValvula: 0
Incrementou
Nivel TANQUE: 17
Estado TANQUE: 4
Tanque mudou de nivel
Sensor: Li ValorBruto



ANIMATE

Ligada

Desligada



Historical Perspective

1998-1999: SIMOO-RT

- Timing requirements specification
- Behavior definition with state-machines
- Functional specification with use-cases and MSD
- Consistence checking based in Meta-model
- Automatic Code Generation in AO/C++



Timing specification: special attributes

Method Name: FindMaxLevel

Activating Message: MAX_LEVEL

Message Parameters:

CString
int
unsigned int
double
float
char

char

Type of Method:

Normal Method
 Activated by Events

Method can interrupt
 Cyclic Method
 Timed Method
Deadline: 100

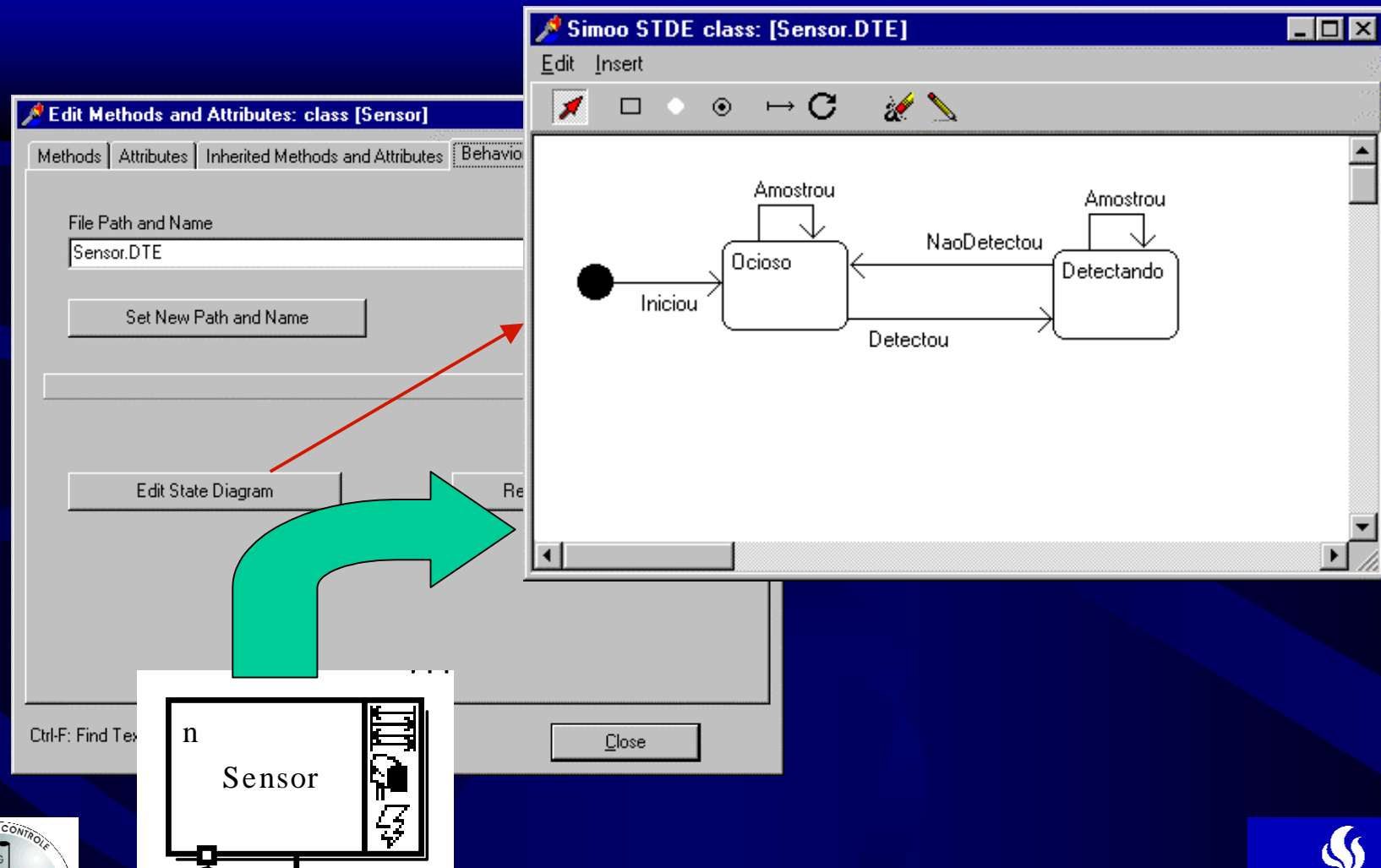
Virtual
 Instrumented

OK Cancel Exception Code: <<

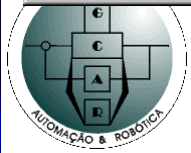
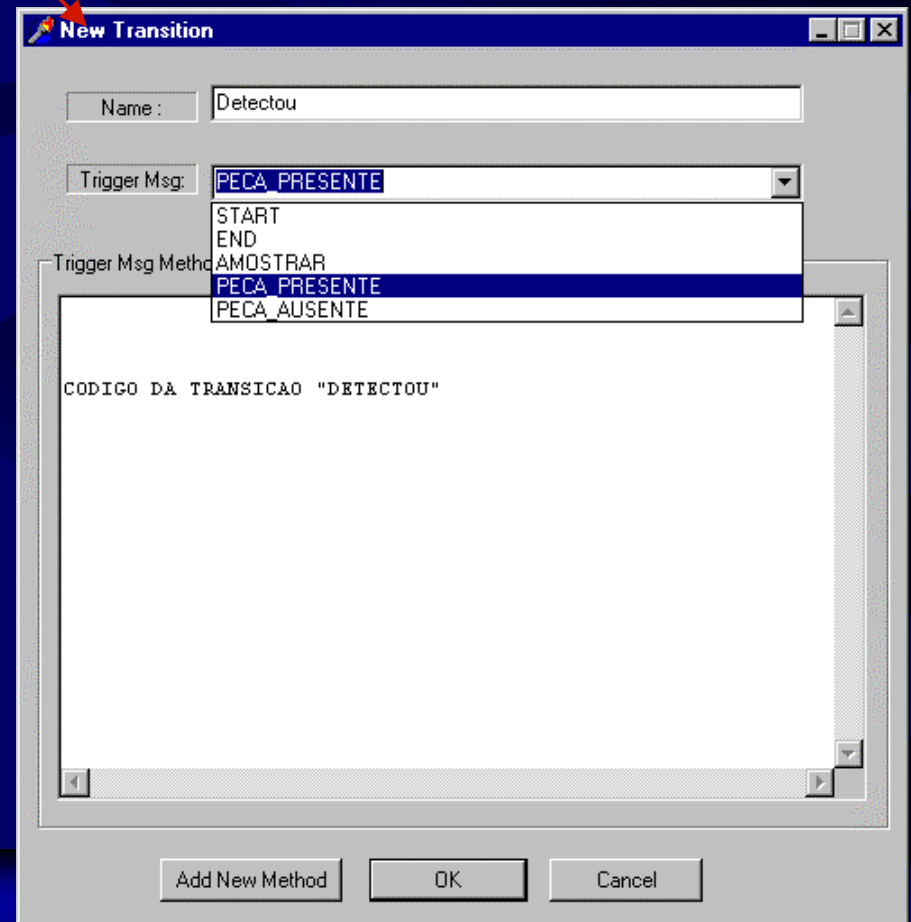
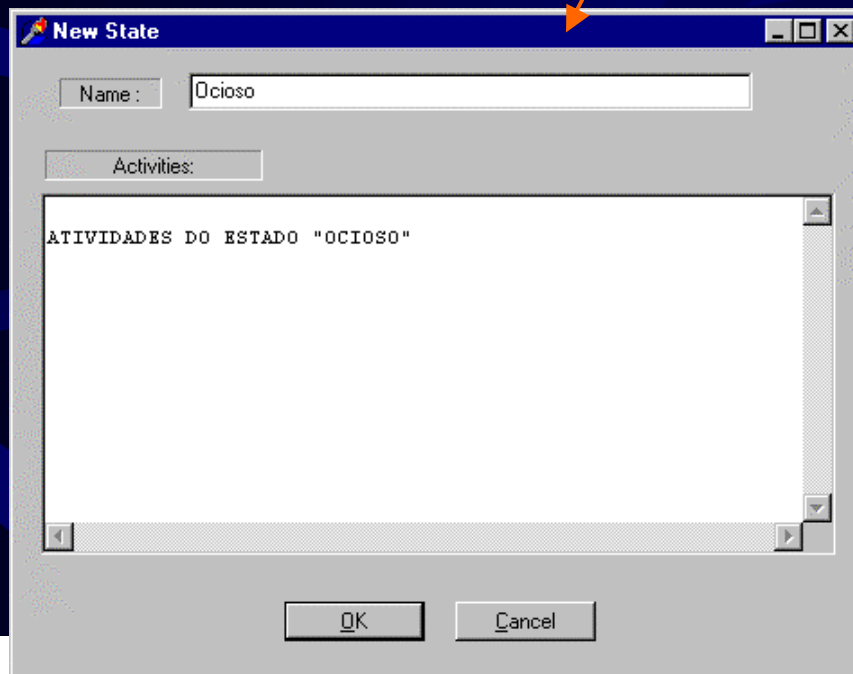
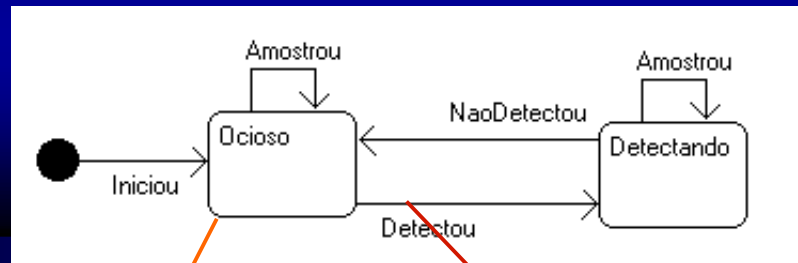
Alarm->TurnOn();

- Cyclic operations
- Timed operations (deadline)
- Default value for the class
- Exception handling code

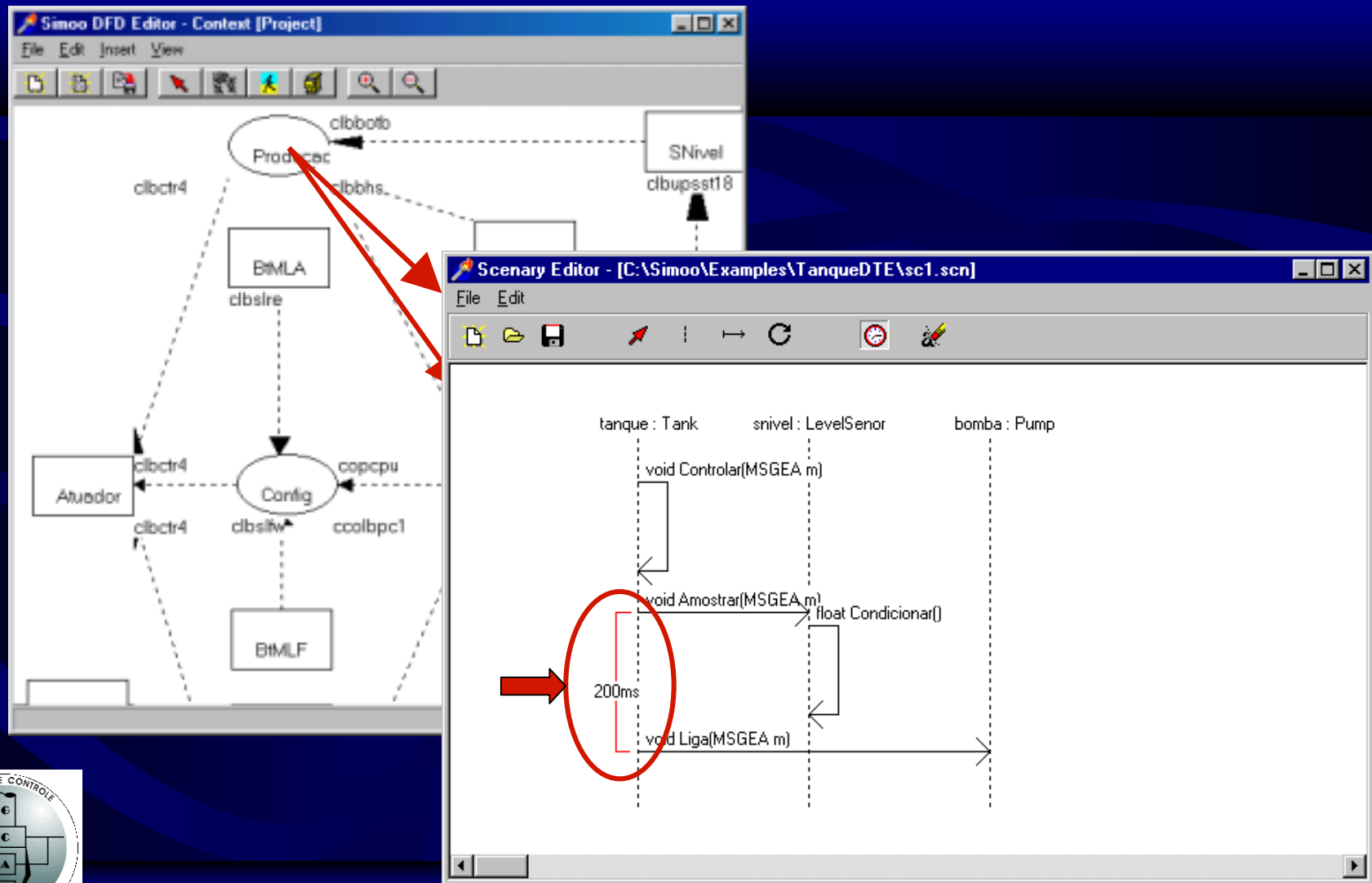
Behavior Specification: State-Transition Diagrams



Behavior Specification: C++ Templates

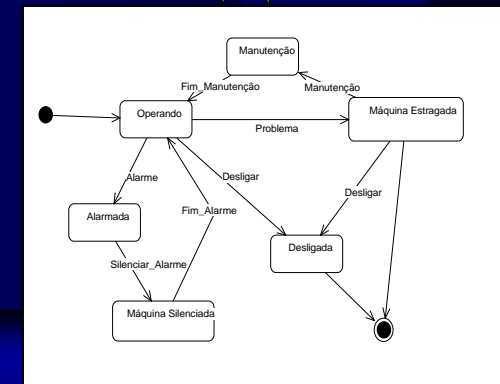
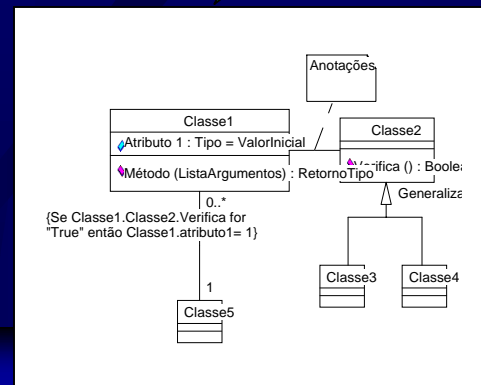
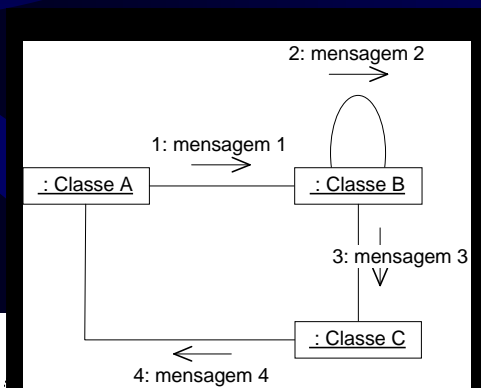
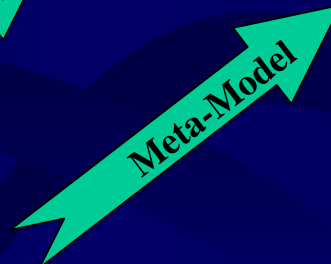
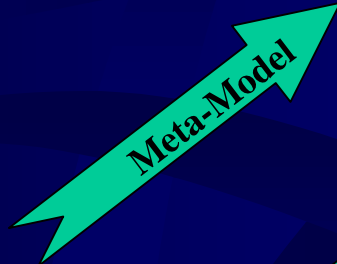
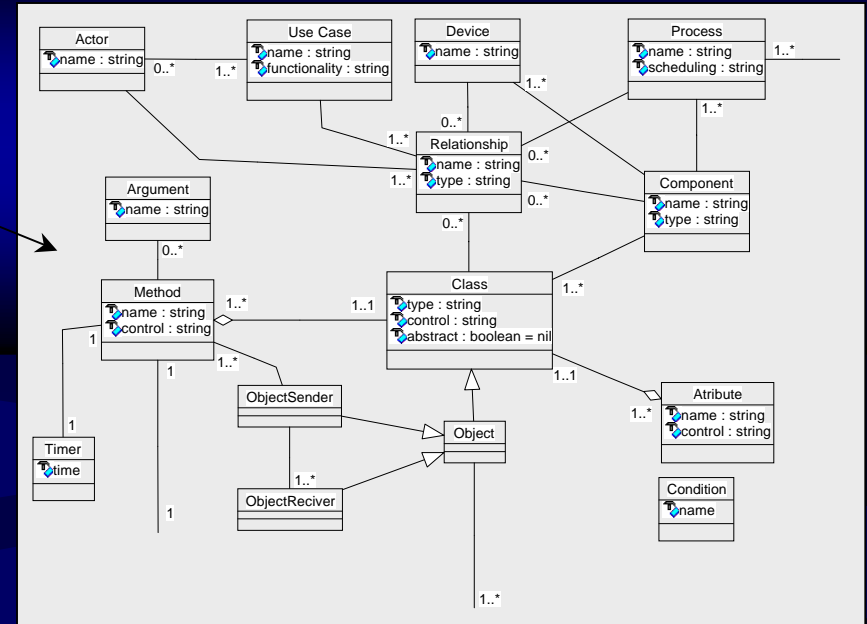
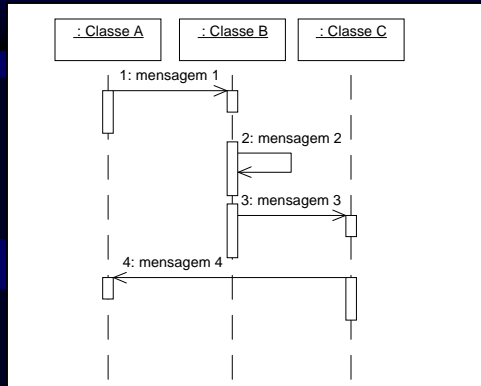


Functional specification: use-cases editor

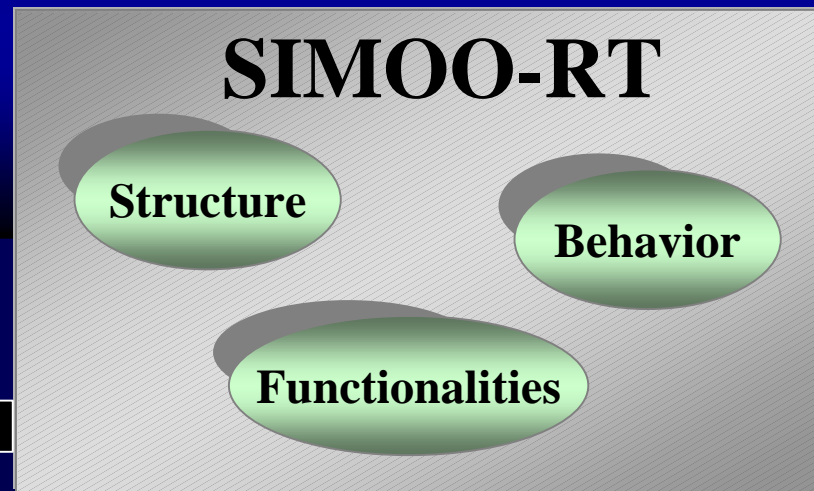


Meta-Model

Consistency Checking



RT-code Generation: AO/C++ program



AO/C++ Classes
and
Makefile

Pre-processor

Instrumented
C++ Classes

C++ Compiler

RT Application



Current work

- Further extensions to SIMOO-RT:
 - Instrumentation: Gergeleit (ISORC 99 + special issue)



Instrumentation: Validation of the temporal requirements

The image displays a software development tool interface with three main windows:

- Edit Class:** Shows a class named "Sensor". It includes fields for "Class Name", "Connection Points List", "Messages List", and "Initialization Parameter". The "Cardinality" is set to "0". The "Instrumented" checkbox is checked.
- Add Attribute:** A small dialog box showing the addition of a new attribute: "float LogicValue;". The "Instrumented" checkbox is checked.
- Insert Method:** A dialog box for configuring a new method named "Sample". It is activated by the message "SAMPLE". The message parameters are "CString", "int", "unsigned int", "double", "float", and "char". The "Type of Method" is set to "Activated by Events". The "Instrumented" checkbox is checked. The "Cyclic Method" checkbox is checked, with a "Period" of "500".

An arrow points from the "Edit Class" window to a green oval containing the text "Distributed C++ Program".

Current work

- Further extensions to SIMOO-RT
 - Instrumentation: Gergeleit (ISORC 99 + special issue)
 - Use cases + EDFDs: Automation Object Identification (ISORC 00 + AARTC 00 best paper)

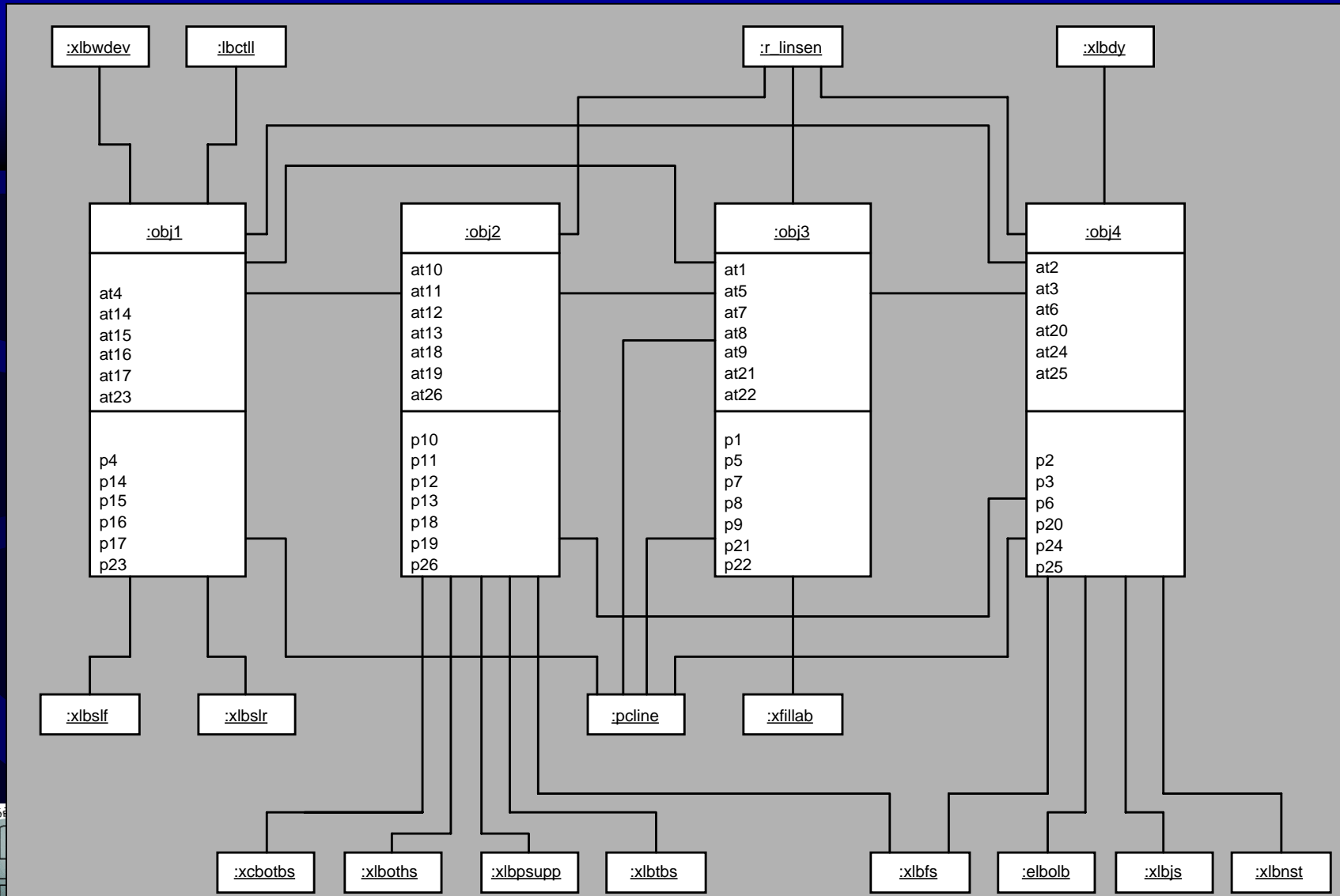


MOSYS Design Methodology

- Functional model is mapped into a graph description
- A graph partition is carried out according to different criteria, leading to different possible task clustering
- Clustering criteria:
 - min-cut algorithm (enhance object autonomy)
 - number of objects (suggested by the designer)
 - object weight balance (in the algorithm)



Example of automatic architecture

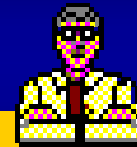
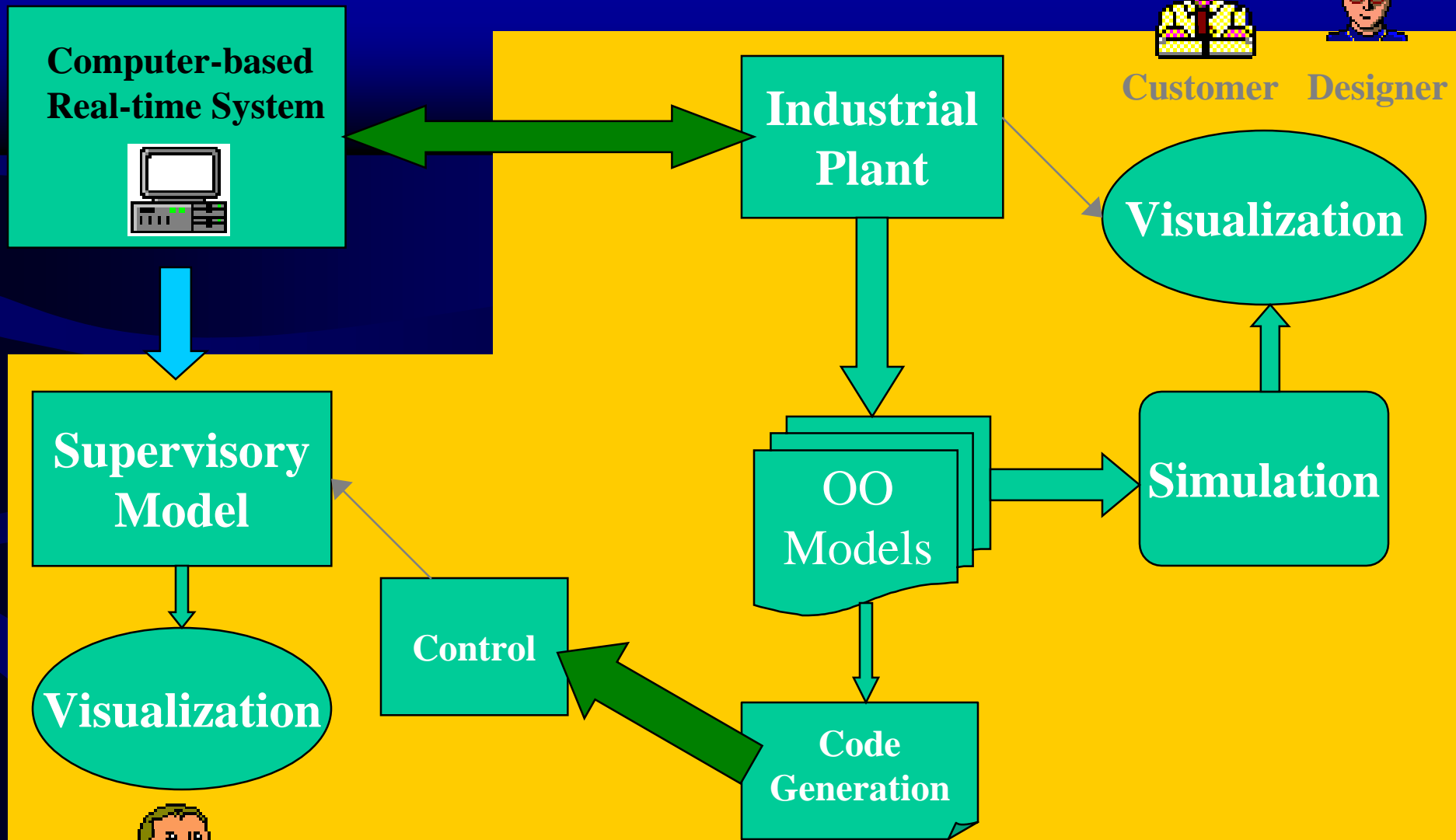


Current work

- Further extensions to SIMOO-RT
 - Instrumentation: Gergeleit (ISORC 99 + special issue)
 - Use cases + EDFDs: Automation Object Identification (ISORC 00 + AARTC 00 best paper)
 - Supervisory systems (WORDS 99)



Proposed Environment:



Customer Designer



Multi/Mono Processor

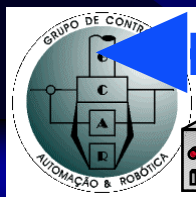
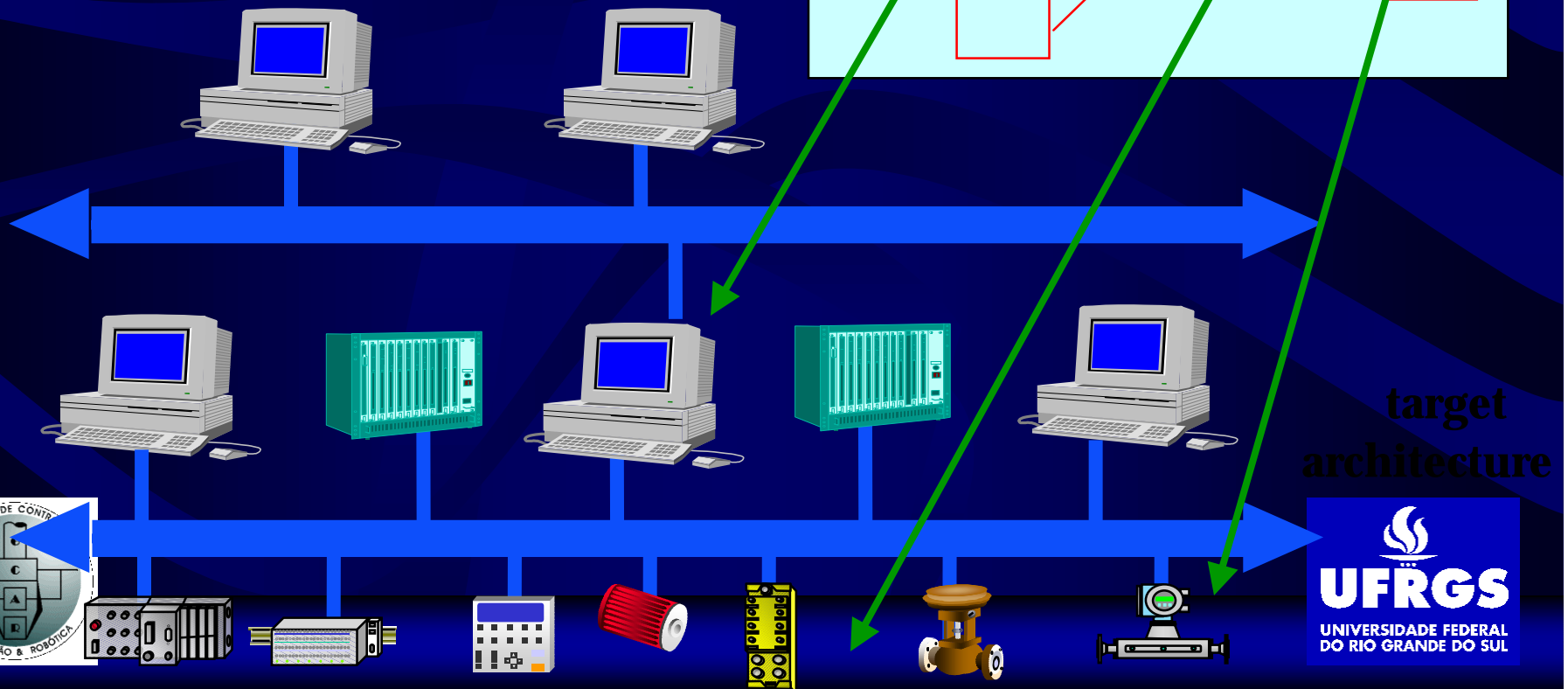
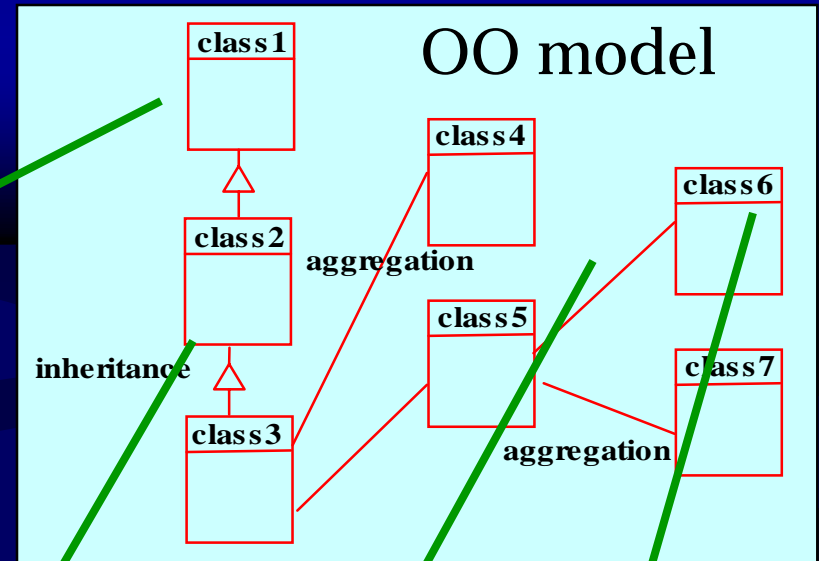
UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Current work

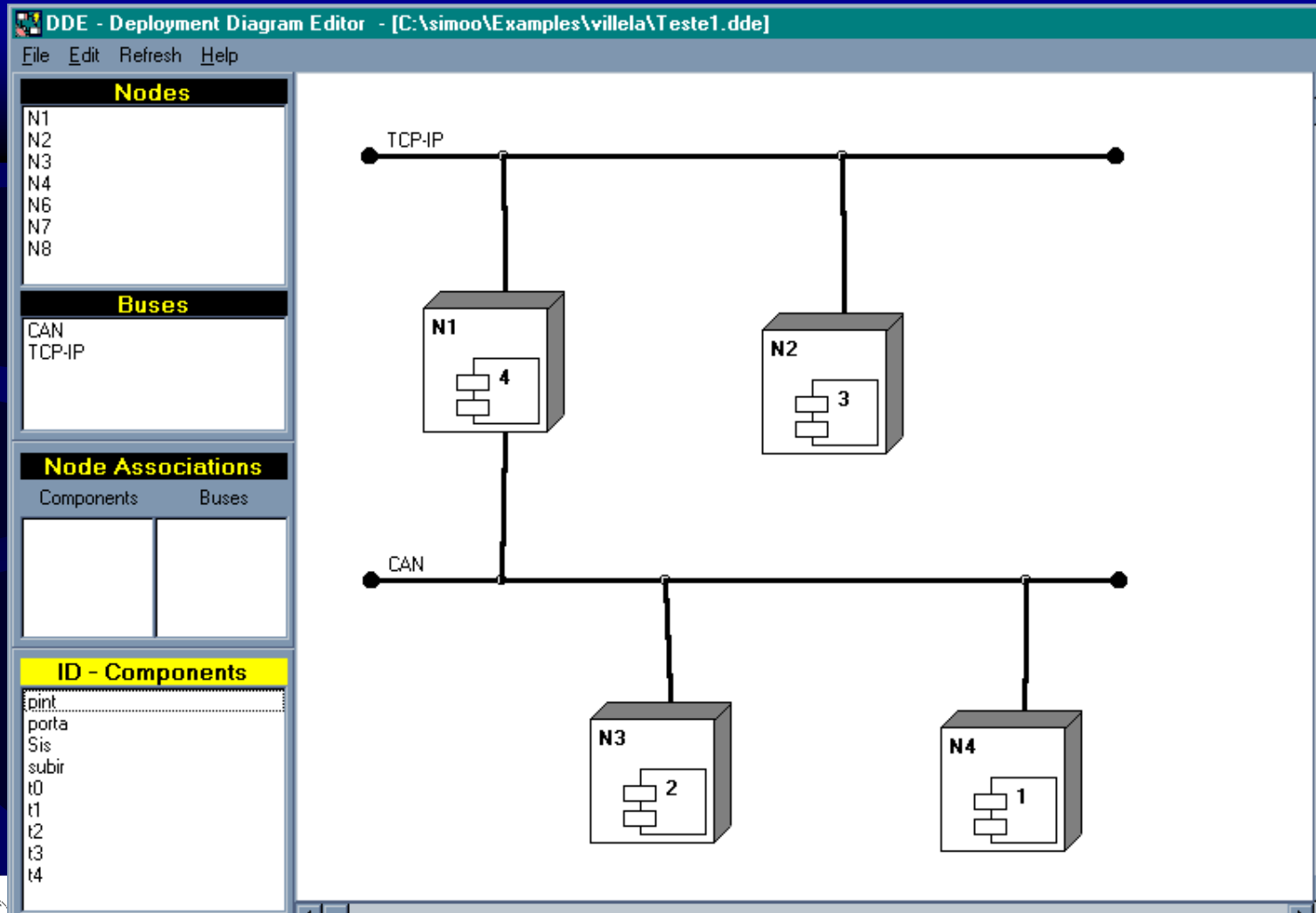
- Further extensions to SIMOO-RT
 - Instrumentation: Gergeleit (ISORC 99 + special issue)
 - Use cases + EDFDs: Automation Object Identification (ISORC 00 + AARTC 00 best paper)
 - Supervisory systems (WORDS 99)
 - Deployment Diagram



Mapping Objects to Processors



Deployment diagram



Current work

- Further extensions to SIMOO-RT
 - Instrumentation: Gergeleit (ISORC 99 + special issue)
 - Use cases + EDFDs: Automation Object Identification (ISORC 00 + AARTC 00 best paper)
 - Supervisory systems (WORDS 99)
 - Deployment Diagram
 - Code generator:
 - Linux/TCP, mLinux/CAN
 - Future work: TAO (RT-CORBA), RT-JAVA



Current work

- Further extensions to SIMOO-RT
 - Instrumentation: Gergeleit (ISORC 99 + special issue)
 - Use cases + EDFDs: Automation Object Identification (ISORC 00 + AARTC 00 best paper)
 - Supervisory systems (WORDS 99)
 - Deployment Diagram
 - Code generator:
 - Linux/TCP, mLinux/CAN
 - Future work: TAO (RT-CORBA), RT-JAVA
 - Mapping timing requirements to QoS specifications



Current work

- Further extensions to SIMOO-RT
 - Instrumentation: Gergeleit (ISORC 99 + special issue)
 - Use cases + EDFDs: Automation Object Identification (ISORC 00 + AARTC 00 best paper)
 - Supervisory systems (WORDS 99)
 - Deployment Diagram
 - Code generator:
 - Linux/TCP, mLinux/CAN
 - Future work: TAO (RT-CORBA), RT-JAVA
 - Mapping timing requirements to QoS specifications
 - From QoS specs to Integrated Scheduling of Processes and Messages => Mode-change, adaptive schedule (based on sound RT scheduling theory)



Current work

- HW-Support
 - Low-cost board (microcontroller-based) for running distributed objects (RT-mLinux, CAN)
 - Scheduler co-processor (handling of event-based and time-based activation of tasks, scheduling, measurement of runtime execution times)



Final Remarks

- Drawbacks identified in 90s were only partly solved and remain a challenge
- RT-DOC has increased interest of research community (this workshop is a good example)
 - Good: more people and more funding
 - Bad: proliferation of non technical papers

