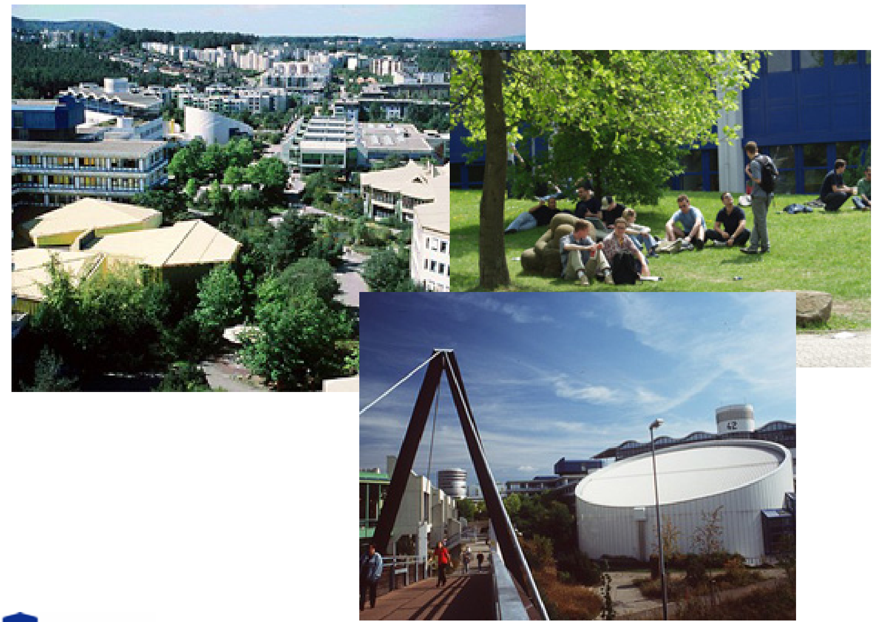
Design of Distributed Automation Systems using UML and IEC61499

Georg Frey

University of Kaiserslautern

UnB



2007-05-25
© Georg Frey

2007-05-25
© Georg Frey

- **Design of Distributed Automation Systems**
  - **Development Processes based on UML**
  - **Object-Oriented Automation O²A**
  - **Design based on IEC 61499**
  - **Implementation on Networked Devices**
  - **Object Oriented Simulation of Automation Systems (SIL, HIL)**
  - **Functionality Based Design using Automation Objects**
- Analysis of Networked Automation Systems
  - Simulation
    - discrete
    - continuous
  - Model Checking
    - timed
    - probabilistic

2007-05-25
© Georg Frey

Outline

- What's the Problem with Distributed Automation anyway?
  - ➢ Why not use the known approaches to do it!
- There is a new Standard?
  - ➢ What it describes and especially what it not describes!
- Is it useful?
  - ➢ Yes, but there are some problems!
- Is it complete?
  - ➢ No, but it can be made complete!
- Where is the connection to UML?
  - ➢ Presentation of a design approach

What's the Problem with Distributed Automation anyway?

Centralized automation means scan-based execution

- All input data is scanned at the same time
- All algorithms are executed in a sequence
- All output data is written at the same time
- Main Features:
  1. The scan-cycle assures that all the algorithms work on the same process data
  2. The sequential execution is used to assure that algorithms work on current data

In a distributed system (networked intelligent devices) there are several asynchronous scan-cycles

- Input data may be read at different instants in time
- Algorithms may be executed concurrently
- Output data may be written at different instants in time
- Resulting Problems
  1. Several algorithms may work on different samples of the process data
  2. The execution order of algorithms is no longer clear. Algorithms may work on old results of other algorithms.

Why not use the known Approaches to do it? UnB

Problems can be fixed by message exchange between algorithms

- Algorithm A informs Algorithm B that he has now finished calculating the data B needs
- Now the execution order is no longer relevant for the result of the calculation (but still for efficiency)

Problems with inconsistent I/O is fixed by an additional I/O-image

- Input reading and output writing is modeled as an algorithm (additional internal I/O image)
- Based on message exchange (as above) validity is assured

Why not implement this in IEC 61131

- Large overhead because it is not a feature of the standard
- Error-prone if built manually

There is a new standard? UnB

- IEC 61499 is an approach to solve exactly this problem

  - Definition of a new type of **Function Block**
  - Definition of **Events** to control execution and to indicate validity of data
  - Definition of **Service Interface Function Blocks** to actively communicate with the process (and the network)

  - Definition of **Composite Function Blocks** to allow hierarchical structuring
  - Definition of a surrounding model of system/device/resource
  - Definition of management functions
  - Definition of standard libraries …

  - NO definition of new programming language (IEC 61131-3 is re-used)
  - NO definition of a development process
  - NO definition of function to system mapping

  - NO Object-Oriented Paradigm for Automation

Object-Orientation in IEC 61499

- A Function Block according to IEC 61499 in an Object
  - Instantiation from a Function Block Type (Class)
  - Encapsulation of Data
  - Encapsulation of Algorithms (Procedures)
- IEC 61499 as a whole is NOT Object Oriented
  - No Inheritance
  - No OO development process
- IEC 61499 combines concepts from OO with the FB concept already accepted in engineering (IEC 61131, Simulink, …)
- Problem: The IEC 61499 FBs follow an execution model not readily known and accepted by people familiar with other FB concepts
  - EVENT-BASED EXECUTION

What is an Event?

UnB

- Signals discrete or continuous are defined at all time
- This is also true for time-discrete representations
- An Event occurs spontaneous
- (In Theory) has no duration
- Events are “consumed” by processes

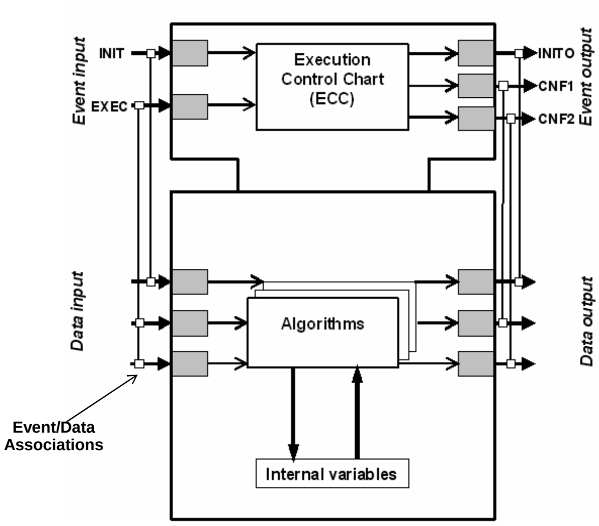
→ You must not miss an event

- Signals (changes thereof) may be transformed to events
- Events may be transformed to signals
- Most naturally: Binary signal B1 is converted to a set of two events: E1=Change to Zero, E2 = Change to One
- Not so naturally: Analogue signals

→ Events fit well in manufacturing but less in process applications

2007-05-25
© Georg Frey

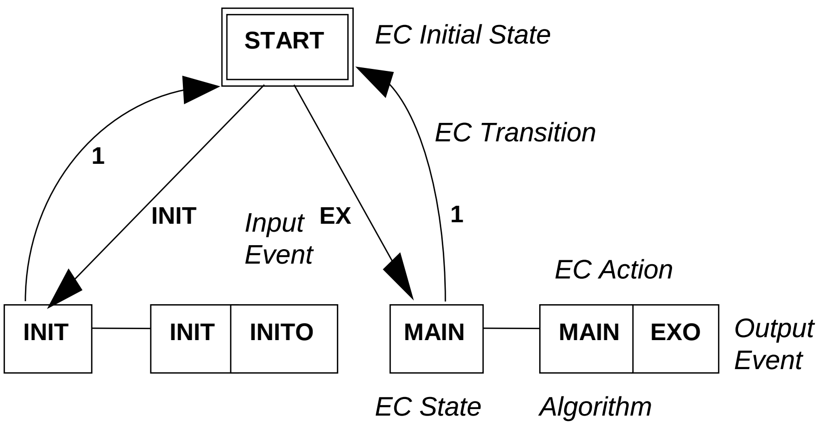
Basic Function Block



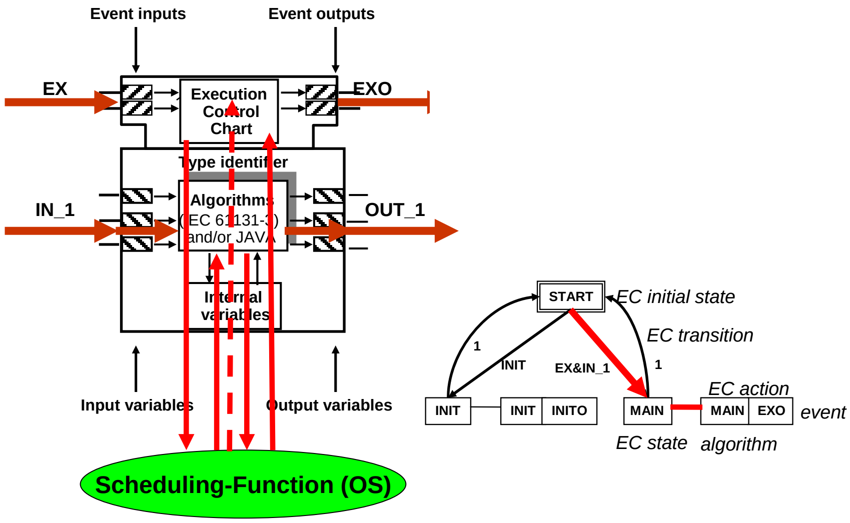
- Events may be associated with data
- Data is read by the block when the associated event occurs

Execution Control Chart (ECC)

- Separation of Control Flow (Events) and Data Flow
- Local Execution Control in each FB
- Algorithms are implemented using IEC 61131-3 languages or others like C or JAVA

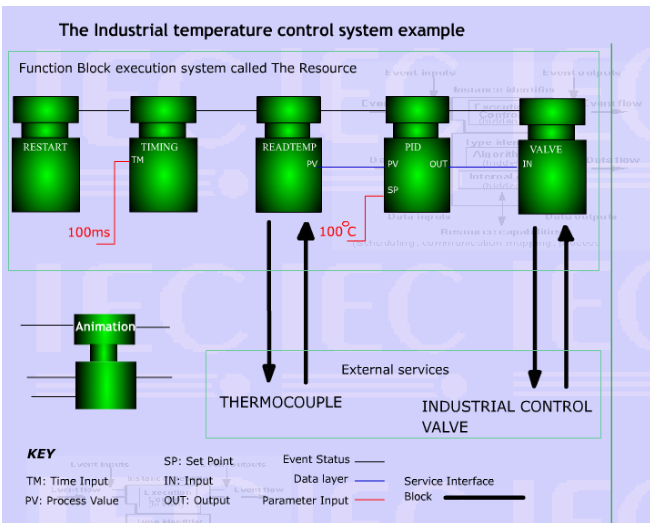


Execution of an IEC 61499 FB



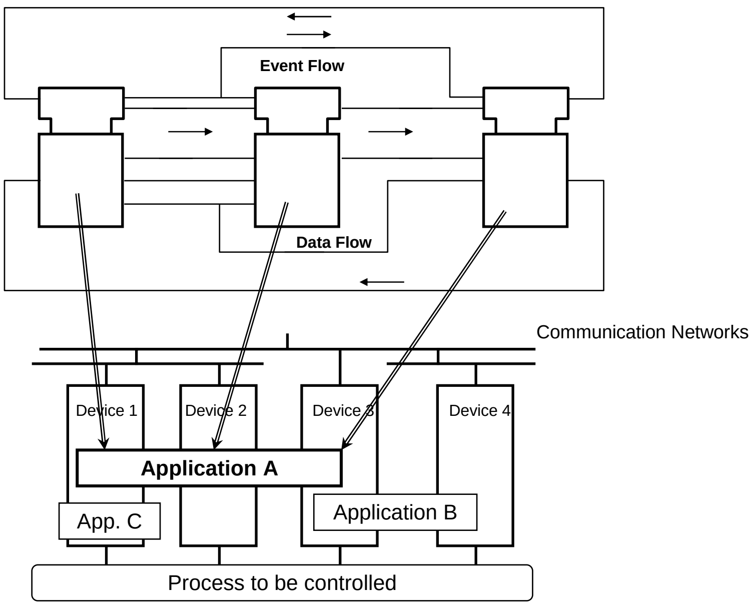
Scan-based (time-triggered, cyclic) Execution in IEC 61499

- Scan-based execution is a special case of event-based execution



Distributed Applications and the IEC 61499 System Model

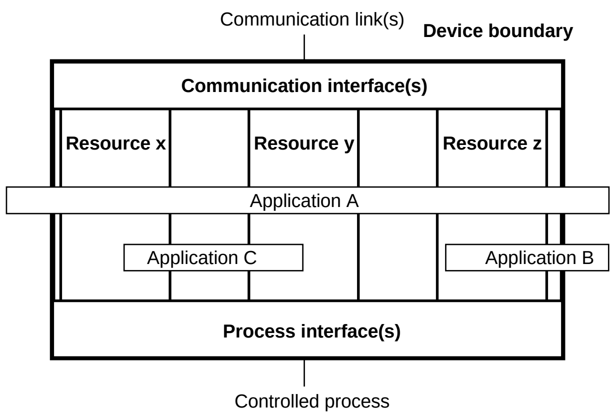
UnB



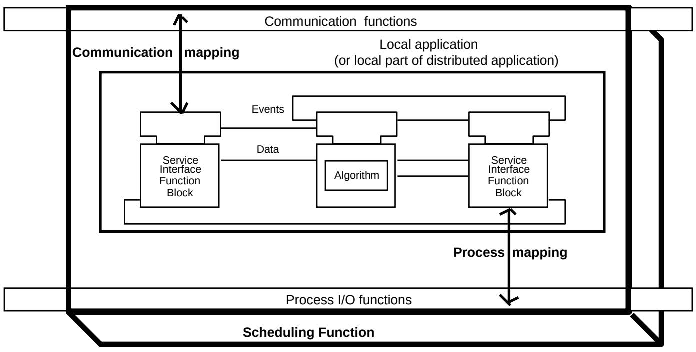
2007-05-25 © Georg Frey

IEC 61499 Device Model

UnB



2007-05-25 © Georg Frey



2007-05-25
© Georg Frey

YES, it solves the main problem of distributed systems

- Consistency of data
- Explicit execution model

However there are open Problems

- Unclear definition of
  - Execution model
  - event-handling
  - data-handling
- Not an implementation standard (no tests defined)

Several compliant tools will present different behavior

2007-05-25
© Georg Frey

Execution of a FB-Network

UnB

- Sequential:
  For FBs running in the same resource under a single-task (possible PLC-like) execution model.
  Note: even this simple case has an additional problem: unlike for example in the FB Diagram of IEC 61131, 61499 does not define an order of the constituent FBs in a diagram. Hence here is another weak point that could lead to different interpretations.

- Synchronous:
  For FBs running in one resource under a multi-tasking-system that realizes task switching times very short compared to the execution times in the application. In this case, it could be safely assumed that the algorithms run in parallel.

- Asynchronous:
  For FBs running on different resources where it is not possible to make an assumption like in the synchronous case.

Event and Data Handling

UnB

- How to implement Events
  - Messages
  - Shared Variables (same resource)

  - Technical Problem could be solved

- How to handle Events
  - Event occurs while FB is still processing the same type of event (unsafe state)
  - Different FBs are waiting for Events from each other FBs (blocking)
  - Different routes in the network (hazards)

  - DES Theory can solve (analyze) these properties iff the model is clear (1st point)

- When is data actually read
  - Occurrence of the Event at the input-port
  - Consummation of the Event by the ECC

  - Could be solved by encapsulating data and event in one message

Solution: Definition of Compliance Profiles

UnB

2007-05-25
© Georg Frey

Is it complete?

UnB

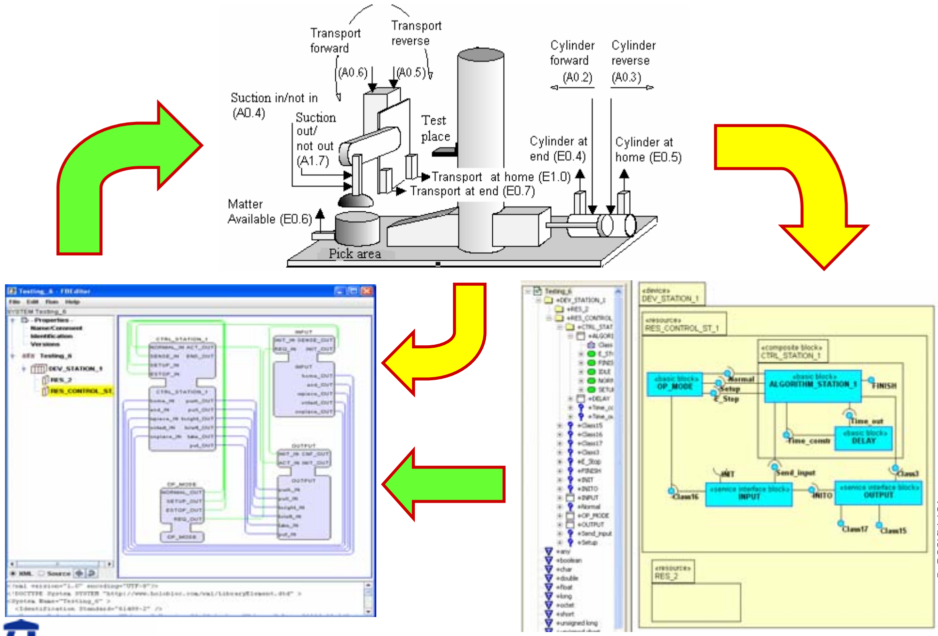
NO

- Mapping problem (slide 8) is not solved
  - at least two groups are working at this
  - Component models are needed for HW and SW
  - Metrics have to be defined
  - The rest is optimization. However we will need rules to reduce search space
- Development Process is not defined
  - FBs are not suitable for all stages in a development process
  - UML seams to be the solution
  - several approaches are already published and will be further investigated

2007-05-25
© Georg Frey

Development using UML

UnB



FBC-S³ Approach

UnB

Functionality based Control (FBC) features

- Using functional units based on fundamental functionalities in manufacturing
- Fully adopt OO-paradigm
- Provides operation mode handling

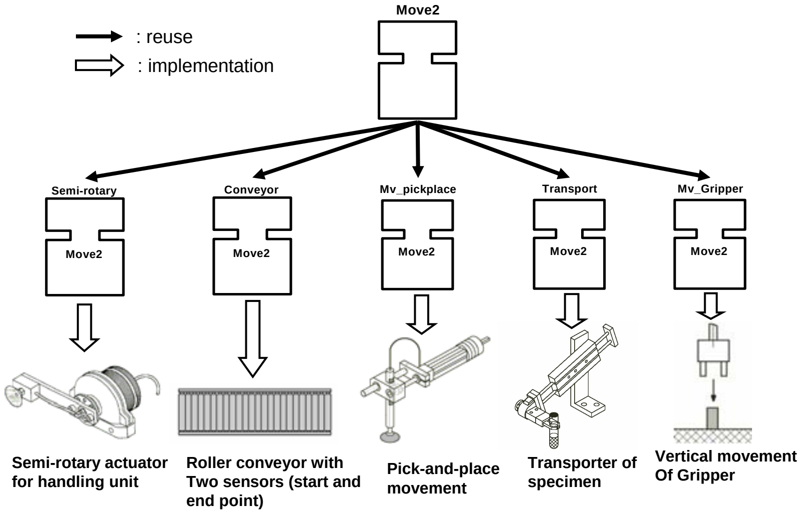
Scheduler, Selector, and Synchronizer (S³) features

- *Scheduler* contains the schedules' concept of component execution
- *Selector* gets the task from Scheduler and executes the component(s)
- *Synchronizer* resumes the task process and confirms the successful of task

Reuse of FBC-based Components

UnB

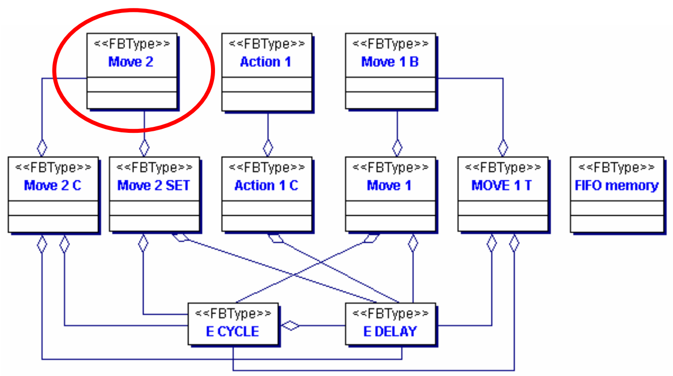
Improves reusability using FBC-based component:



2007-05-25 © Georg Frey

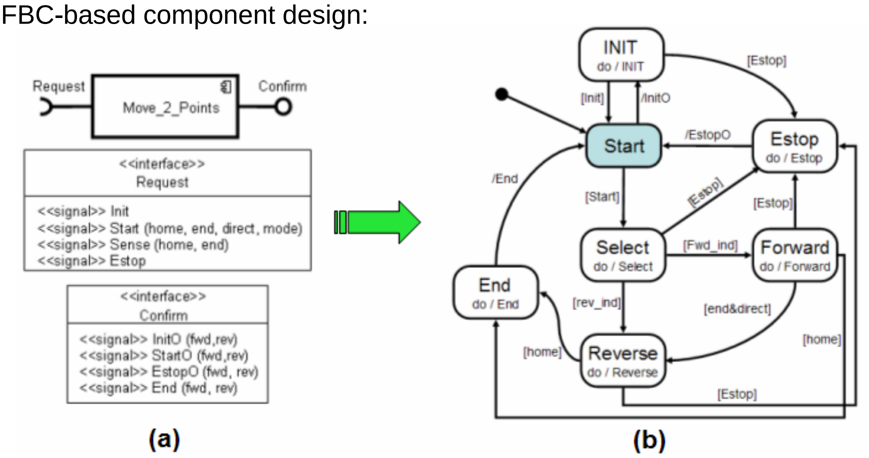
The use of UML for FBC Component design

UnB



2007-05-25 © Georg Frey

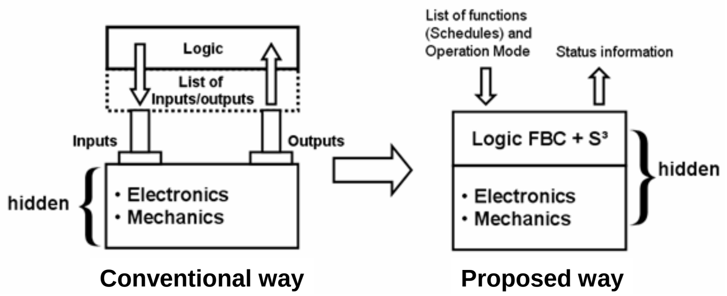
FBC-based component design:



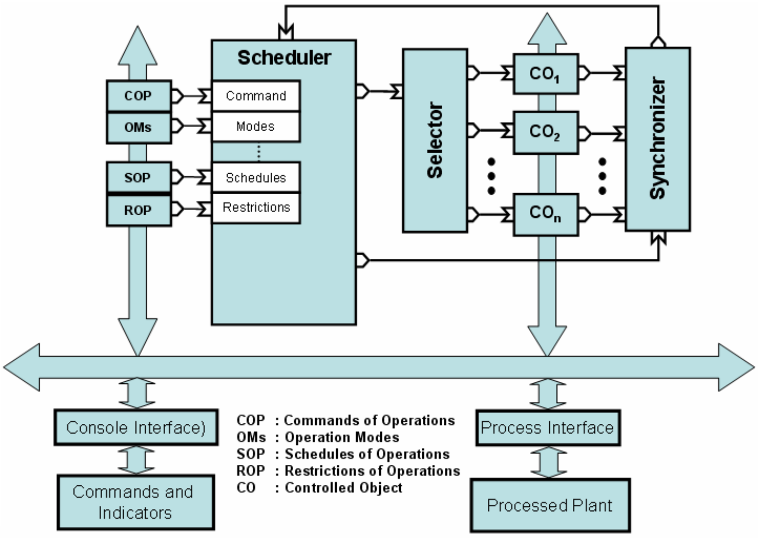
Move_2_Points model: (a) Structural using UML Class Diagram and Component, (b) Behaviour model using UML State diagram

FBC (Functionality based control) is an approach which considers the common Functionalities on automation system to design and build the software controller

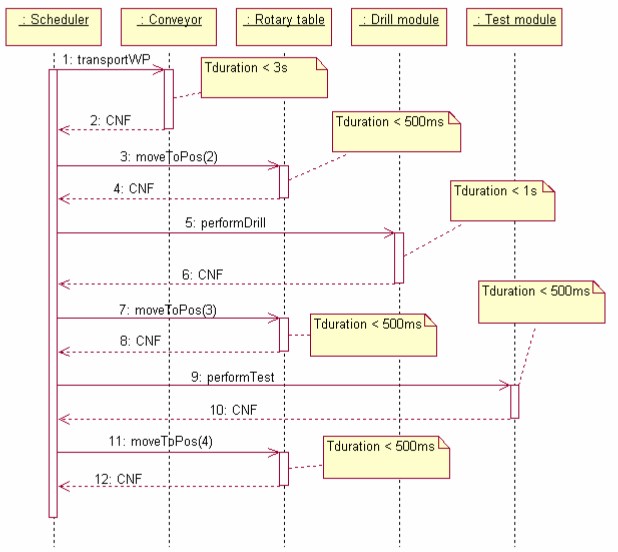
Migration of control strategy



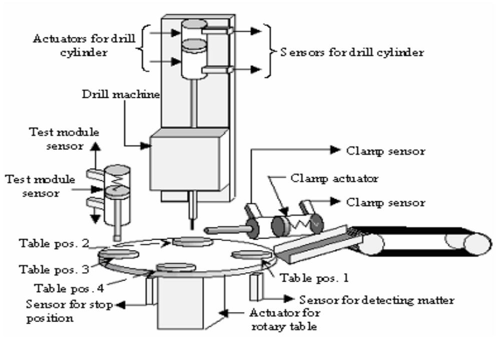
S³ (Scheduler-selector-synchronizer) based control utilizing OMs-based scenario (COP, OMs, SOP, ROP) and FBC-based component (CO).



Sequence diagram for normal production in normal operation mode:



Example: Calculation the number of Controlled objects (COs) UnB

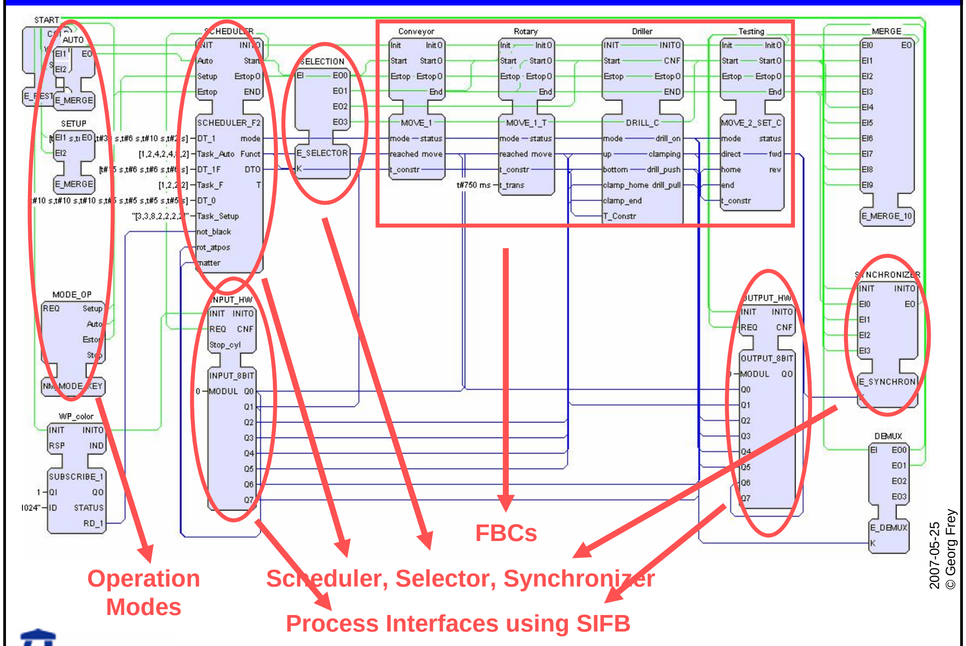


Calculating the required FBC-based components

The selected functional units that will be implemented

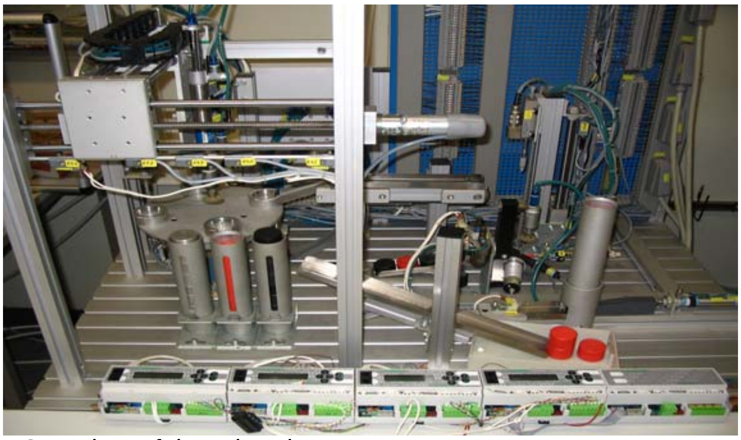
FB	Functionality	Selected FBCs
Conveyor	Switching on/off the concerned motor	Move 1 point
Rotary table	Switching on/off motor ensuring at certain position	Move 1 point
Testing module	Push/pull cylinder	Move 2 points
	Drill Module (Composite FB)	
Clamp	Push/pull the clamp cylinder	Move 2 points
Drill cylinder	Push/pull the drill cylinder	Move 2 points
Drill machine	Switch on/off drill motor	Action_1

Example: Implementation using IEC 61499 UnB



Lab Plant

UnB



Overview of the migration

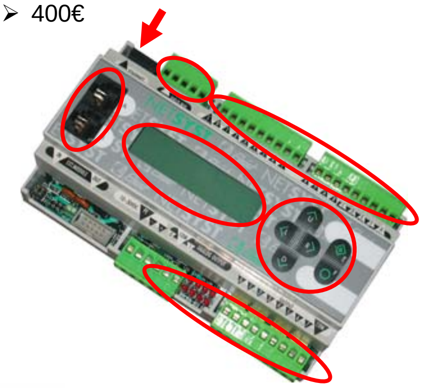
Modeling/Analysis	Implementation	Hardware platform
SIPN	Siemens Step5	PLC
IEC 61499	Java	4 NETMASTER

2007-05-25
© Georg Frey

Aside on Network-enabled controllers

UnB

- **Features of network-enabled controllers:**
  - Support one or more than one Ethernet protocols.
  - Mostly programmable with high-level languages (C, Java, C++ etc.).
  - Interfaces to standard fieldbuses.
  - Interfaces to digital and/or analog I/O.
  - 400€



- **Digital and Analog I/Os**
- **LCD Display and Keys**
- **CAN controller and ports**
- **RS 232 serial ports**
- **Ethernet port**
- **1-wire peripherals**
- **I²C extension possibilities**

2007-05-25
© Georg Frey

Summary

IEC 61499

- IEC 61499 is a promising step towards distributed automation
- Abstracts from implementation (controller, network)
- Technical problems in the standard could be resolved
- Standard could be integrated into an OO development process

Development Process

- Development process for DCS based on UML and IEC 61499
- Re-use and modularity based on FBC
- OM-handling and reconfiguration using S^3

Discussion

Thank You