Design of Distributed Automation Systems using UML and IEC61499

Georg Frey

University of Kaiserslautern
JPA² Group at University of Kaiserslautern: People

The Juniorprofessorship (JPA²) at the University of Kaiserslautern is part of the Department of Electrical and Computer Engineering. The focus of our research is the development of methods for the design of autonomic (distributed) automation systems. In addition to the research and development processes, we also study the functional and performance aspects of these systems.

JPA² Group at University of Kaiserslautern: Research

- 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
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?

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?

- 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?

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

Execution Control Chart (ECC)

START

EC Initial State

INIT

EC Transition

Input

Event

EX

1

1

Output

Event

EC Action

EC State

Algorithm

Init

Init

InitO

Main

Main

Ex0
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

Event Flow

Data Flow

Communication Networks

Device 1

Device 2

Device 3

Device 4

Application A

App. C

Application B

Process to be controlled

IEC 61499 Device Model

Communication link(s)

Device boundary

Communication interface(s)

Resource x

Resource y

Resource z

Application A

Application C

Application B

Process interface(s)

Controlled process
Is it useful?

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
Execution of a FB-Network

- **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

- **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 if 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
Is it complete?

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
FBC-S³ Approach

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

Improves reusability using FBC-based component:

- Semi-rotary actuator for handling unit
- Roller conveyor with two sensors (start and end point)
- Vertical movement of Gripper
- Transporter of specimen
- Pick-and-place movement
- Pick-and-place movement of Gripper

The use of UML for FBC Component design
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.
S³ (Scheduler-selector-synchronizer) based control utilizing OMs-based scenario (COP, OMs, SOP, ROP) and FBC-based component (CO).

Example: Design of SOP for normal production

Sequence diagram for normal production in normal operation mode:
Example: Calculation the number of Controlled objects (COs)

Calculating the required FBC-based components

<table>
<thead>
<tr>
<th>FB</th>
<th>Functionality</th>
<th>Selected FBCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conveyor</td>
<td>Switching on/off the concerned motor</td>
<td>Move 1 point</td>
</tr>
<tr>
<td>Rotary table</td>
<td>Switching on/off motor ensuring at certain position</td>
<td>Move 1 point</td>
</tr>
<tr>
<td>Testing module</td>
<td>Push/pull cylinder</td>
<td>Move 2 points</td>
</tr>
<tr>
<td>Clamp</td>
<td>Push/pull the clamp cylinder</td>
<td>Move 2 points</td>
</tr>
<tr>
<td>Drill cylinder</td>
<td>Push/pull the drill cylinder</td>
<td>Move 2 points</td>
</tr>
<tr>
<td>Drill machine</td>
<td>Switch on/off drill motor</td>
<td>Action 1</td>
</tr>
</tbody>
</table>

The selected functional units that will be implemented

Example: Implementation using IEC 61499

Operation Modes
Scheduler, Selector, Synchronizer
Process Interfaces using SIFB
Overview of the migration

<table>
<thead>
<tr>
<th>Modeling/Analysis</th>
<th>Implementation</th>
<th>Hardware platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPN</td>
<td>Siemens Step5</td>
<td>PLC</td>
</tr>
<tr>
<td>IEC 61499</td>
<td>Java</td>
<td>4 NETMASTER</td>
</tr>
</tbody>
</table>

Aside on Network-enabled controllers

- 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
- PC extension possibilities
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 S³

Discussion

Thank You