A Profile for Integrating Function Blocks into the Unified Modeling Language

Torsten Heverhagen, Rudolf Tracht
Automation and Control, Dept. of Mechanical Engineering,
University of Duisburg-Essen, Germany

Robert Hirschfeld
DoCoMo Euro-Labs, Future Networking Lab, Germany
Overview

• Introduction and Motivation
  – Drilling Example

• Function block oriented languages
  – Matlab/Simulink
  – IEC 61131-3
  – IEC 61499

• Integration of function blocks into UML
  – Comparison of FBs and Classes
  – Profile “Functionblockadapters”

• Summary and Outlook
Drilling Example

• Four steps
  – Moving to drilling position
  – Drilling
  – Cleaning the hole
  – Moving back
Mathematical Modeling of the Plant

\[
\begin{align*}
\dot{\omega}(t) &= -a\omega(t) + bu(t) \\
\phi(t) &= \omega(t) \\
y(t) &= \alpha\phi(t)
\end{align*}
\]
Control Loop Design

- Common structure of feedback control loops
- Controller is real-time software
  - implemented in microcontrollers, PLCs, ...
- Plant is real world, but for simulation and testing it is also software
Simulation of Continuous Time Controllers

• For example using Matlab/Simulink™:

behavior of such blocks can also be described using

– z-transfer functions,
– statecharts,
– ...
Function Block Oriented Languages

• Matlab/Simulink

\[
\begin{align*}
        &x = Ax + Bu \\
        &y = Cx + Du \\
        &\frac{1}{s+1} \\
        &\frac{1}{z+0.5}
\end{align*}
\]

State-Space Transfer Func. Discrete Transfer Func.

• IEC 61131-3 (Languages for programmable controllers)
  – 5 languages

• IEC 61499 (Function Blocks for Industrial-Process Measurement and Control Systems)
  – Execution Control Chart (ECC)
  – action language from 61131-3
A General Function Block Model

- Our working model for the integration of existing FB-languages into UML
  - interface: input, output, through variables (all static)
  - internal: static data structures + behavior
  - type (class) and instance
Example for Protocols of Function Blocks

PLCopen

Standardization in Industrial Control Programming

Technical Specification
PLCopen - Technical Committee 2 – Task Force

Function blocks for motion control

Version 1.0

<table>
<thead>
<tr>
<th>Axis Move Absolute</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXIS_REF</td>
<td>Axis</td>
</tr>
<tr>
<td>BOOL</td>
<td>Execute</td>
</tr>
<tr>
<td>REAL</td>
<td>Position</td>
</tr>
<tr>
<td>REAL</td>
<td>Velocity</td>
</tr>
<tr>
<td>REAL</td>
<td>Acceleration</td>
</tr>
<tr>
<td>REAL</td>
<td>Deceleration</td>
</tr>
<tr>
<td>REAL</td>
<td>Jerk</td>
</tr>
<tr>
<td>MC_Direction</td>
<td>Direction</td>
</tr>
<tr>
<td>AXIS_REF</td>
<td>Done</td>
</tr>
<tr>
<td>BOOL</td>
<td>CommandAborted</td>
</tr>
<tr>
<td>BOOL</td>
<td>Error</td>
</tr>
<tr>
<td>WORD</td>
<td>ErrorID</td>
</tr>
</tbody>
</table>

- **Input Data**
  - AXIS_REF
  - BOOL
  - REAL
  - REAL
  - REAL
  - REAL
  - MC_Direction

- **Output Data**
  - AXIS_REF
  - BOOL
  - REAL
  - REAL
  - WORD

- **Error**
  - ErrorID
  - Execute
  - Done or CA
  - input data
  - output data
Example for UML Protocols

mc_client / MC_Client

wait

1: NewPosition

ffwd

1.1: Ended

2: NewPosition

fwd

2.1: Ended

dwell

3: NewPosition

rev

3.1: Ended

wait

/MC_MoveAbsolute

ffwd

fwd

dwell

rev

S0

Error

NewPosition

S1

Error

Ended or Timeout

\[\text{SVERTS '03}
\]
Torsten Heverhagen, Torsten.Heverhagen@web.de
Function Block Adapters

- Work as programmable protocol adapters between UML protocols and function block protocols
- Platform independent
- Provided as a UML profile
Summary and Outlook

• High importance of function block oriented languages in industrial control modeling and programming
• Combination of UML and function blocks is promising
• Function block adapters provide a framework and design patterns for integrating UML and function blocks
• Previous work was based on IEC 61131-3
• Current work:
  – profile Functionblockadapters
  – extension to function block oriented languages
• Outlook: functionblocks.org
Comparison of Function Blocks and Classes

- Function Blocks
  - type
  - instance
  - internal variables
  - internal behavior
  - interface variables

- UML
  - class
  - instance
  - attributes
  - one operation/statechart
  - no direct match
    - this is discussed in the next slide
Discussion of Interface Variables

• Comparison to parameters
  – both have a direction kind (in, inout, out)
  – but: should be independent of behavior execution

• Comparison to attributes
  – both are independent of internal behavior execution
  – but: direction kind is missing

• Comparison to pins of activities
  – both are connectable and have a direction kind
  – but: should be independent of activity execution

• Comparison to ports
  – ports are mightier than interface variables 😊
Using Ports as Interface to Function Blocks

- **Interface** (from Interfaces)
- **Port** (from Ports)

- **FBInPort**
- **FBOutPort**

- **FBInterface**

May only contain one signal called *ValueChanged*

May only require one *FBInterface*

May only provide one *FBInterface*