Re-Engineering Approach for PLC Programs based on Formal Methods

Mohammed Bani Younis
Contents

• Introduction
• Re-Engineering of PLC Programs
• Formalization of PLC Programs
• Visualization of the formalized PLC Programs
• Re-Implementation of PLC Programs
• SW-Quality
• Case Studies
• Summary
• Programmable Logic Controllers (PLCs)
  – Special type of computers used in industrial and safety applications
  – System controlled by PLC programs vary in complexity

• Programming Languages (IEC 61131-3):
  – Ladder Diagram (LD)
  – Instruction List (IL)
  – Function Block Diagram (FBD)
  – Structured Text (ST)
  – Sequential Function Chart (SFC)
  – However, also vendor-specific languages
What is Re-Engineering?

[Chikofsky and Cross 1990]
Why Re-Engineering of PLC Programs?

Introduction
Re-Engineering
Formalization
Visualization
Re-Impl. SW-Quality
Case Stud.
Summary

• Longevity of PLC programs (often more than 30 years)
  → Problems with HW
  – code is HW-specific
  – replacement to a new HW problematic
  – replacement of the supplier problematic
  ✤ e.g.: Siemens S5 is no more produced, Siemens S7 can not process S5 programs

Goals

• Code is continuously adjusted → documentation problems
  – no formal description at the beginning
  – undocumented adjustments
  ✤ Need for visualization

• New Technologies hold move in the area of Automation
  – better SW-Engineering methods
  – short HW-Life cycles
  ✤ Formal model allows adjustments on new HW-SW environment
Industrial Re-Engineering

- Convert STEP 5 \(\rightarrow\) STEP 7 (*Siemens Automation and Drives TIA*)
  - Not all Program Constructs (e.g. Standard functions)
  - Often with simplifications are used
  - Delete non Compatible Blocks and invocations
  - These should be re-programmed in STEP 7
  - Programs of normal instruction are converted easily and complete Addressing

  \(\rightarrow\) Logical dynamic is not converted

- STEP 5 \(\rightarrow\) IEC 61131-3 (*3S CodeSys*)
  - Import .SYM
  - standard.lib to the project
  - SEQ-file as Global Variables of the IEC 61131-3
  - The Address is matched to the IEC 61131-3
  - Non-Valid Characters and Functions are comment out

  \(\rightarrow\) only Instruction mapping (no Logic)
<table>
<thead>
<tr>
<th>Reference</th>
<th>Source</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lang.</td>
<td>Additional</td>
</tr>
<tr>
<td>[Treseler et al., 2000]</td>
<td>IL</td>
<td>Plant</td>
</tr>
<tr>
<td>[Bornot et al., 2000 (b)]</td>
<td>SFC</td>
<td>Without</td>
</tr>
<tr>
<td>[Willems, 1999]</td>
<td>IL</td>
<td>Plant</td>
</tr>
<tr>
<td>[Mader and Wupper, 1999]</td>
<td>IL</td>
<td>Without</td>
</tr>
<tr>
<td>[Brinksma and Mader, 2000]</td>
<td>SFC</td>
<td>Plant</td>
</tr>
<tr>
<td>[Kowalewski et al., 1999]</td>
<td>SFC</td>
<td>Plant</td>
</tr>
<tr>
<td>[Bornot et al., 2000 (a)]</td>
<td>IL</td>
<td>Without</td>
</tr>
<tr>
<td>[Canet et al., 2000]</td>
<td>IL</td>
<td>Without</td>
</tr>
<tr>
<td>[Roussel and Lesage, 1996]</td>
<td>SFC</td>
<td>Without</td>
</tr>
<tr>
<td>[Lampérière-Couffin et al., 1999]</td>
<td>SFC&amp;LD</td>
<td>Without</td>
</tr>
<tr>
<td>[Mertke and Menzel, 2000]</td>
<td>IL</td>
<td>Plant</td>
</tr>
<tr>
<td>[Hassapis et al., 1998]</td>
<td>SFC</td>
<td>Plant</td>
</tr>
<tr>
<td>[Rossi and Schnoebelen, 2000]</td>
<td>LD</td>
<td>Without</td>
</tr>
<tr>
<td>[Baresi et al., 2000]</td>
<td>FBD</td>
<td>Without</td>
</tr>
<tr>
<td>[Hatono and Baba, 1996]</td>
<td>LD</td>
<td>Without</td>
</tr>
<tr>
<td>[Vyatkin and Hanisch, 2000]</td>
<td>FBD</td>
<td>Plant</td>
</tr>
<tr>
<td>[Canet, 2001]</td>
<td>ST</td>
<td>Without</td>
</tr>
</tbody>
</table>
- New trend in controller design
- Majority of the works are Forward Engineering
- Evaluation of OOP delegated through Unified Modeling Language (UML)
- Use of UML as modeling environment
- Use of Internet Technologies (XML, HTML, XSL, etc…)

→ Research against Industry
Compound Re-Engineering

- Compound of OO, Internet, and formal methods

Introduction
Re-Engineering
Formalization
Visualization
Re-Implement.
SW-Quality
Case Stud.
Summary

- Classification and Conversion algorithms
- Verification and Validation
- Analysis and Simulation
- Visualization and Formalization
- Visualized Formal Description
- Re-Implementation
- SW Quality
- SW Metrics

PLC-Code Bosch
PLC-Code S5
PLC-Code IEC

XML

Done
PLC and STEP5

- **STEP 5 in a hierarchy-like form:**
  - **OB:** Organization Module
  - **PB:** Program Module
  - **DB:** Data Module
  - **FB:** Function Module

- **Timers and Counters**
- **Polling mode operation**
- **PLC Memory**

![PLC Cycle Diagram]

![PLC Memory Diagram]
PLC as Discrete Event System

- $\text{PLC}_{\text{system}}/\rho \rightarrow \text{closed loop}$
- plant as a FSM: $\rho = \langle S, \sum, X_0, X_f, \delta \rangle$
- $\text{PLC}_{\text{system}}$ as a tuple $\langle\text{PLC}_{\text{SW}}, \text{PLC}_{\text{HW}}, \text{PLC}_{\text{Cycle}}\rangle$
- $\text{PLC}_{\text{SW}}$ which denotes the PLC program as tuple: $\langle \text{PAE}, \text{PAA}, I, A_{\text{PAE}}, \text{PLC}_{\rho r}, x_0, x_f \rangle$
- $\text{PLC}_M$ module or block as a stand alone is a tuple: $\langle S, \sum, Y, \delta, \lambda, s_0, s_f \rangle$
- $S$ set of states
- $Y = \alpha(PAA)$ output alphabet
- $\delta : S \times \sum \rightarrow S$ transition function
- $\sum = \alpha(PAE)$ input alphabet
- $\lambda : S \times \sum \rightarrow Y$ output function
PLC Program as Communicating Automata

- \( \text{PLC}_{SW} \) is a two subsets \( \text{PLC}_u \) and \( \text{PLC}_{SYS} \)
- \( \text{PLC}_u \) is re-engineering relevant
- \( \text{PLC}_u \) is a model of CFSM \( \text{PLC}_{M1} \ldots \text{PLC}_{Mn} \) of \( \langle S_i, \sum_i, Y_i, \delta_i, \lambda_i, s_{0,i} \rangle \)
- The model \( \text{PLC}_{Mi} \) \( \forall i \in \{1, \ldots, n\} \) \( \text{PLC}_{M1} \otimes \text{PLC}_{M2} \otimes \ldots \otimes \text{PLC}_{Mn} \) builds the automaton \( \text{PLC}_u := \langle S, \sum, Y, \delta, \lambda, s_0 \rangle \)

General feed-forward composition

\[ Y_i = Y_{i1} \times Y_{i2} \]
\[ \sum_i = \sum_{i1} \times \sum_{i2} \]
PLC Program and Cycle

- $PLC_{\text{Cycle}}$ as CFSM with the CFSMs of $PLC_u$
- Example $PLC_{\text{OB1}} \otimes PLC_{\text{PB1}} \otimes PLC_{\text{PB3}}$

Next step is how to formalize PLC blocks?
General Consideration

- State definition through the influence of single operations
- Investigation of the operations on the Status Word
  - Status Word
  - CR (VKE in German)

Different possibilities were examined

1. All possibilities of a single operation are concerned $\rightarrow$ State = $f$ (VKE, PC, internal variables)
2. The conversion of the program according to University of Cachan $\rightarrow$ Q is the set of states and is a tuple (V, a, m), V: variables, a: accumulator, m: program counter
3. Optimization of 2, operation of the same type are merged to form a coherent segment
4. IF-THEN-ELSE transformation $\rightarrow$ State = $f$ (PC, variables)
5. Conversion to Moore machine
6. Based on 4, no need for state contents
### PLC Text

**Example:**

```plaintext
Kommentar: 
Autor: 
Erstellt: 15.07.2003  Geandert am: 
BIB: 0

NETZWERK 1
0000   :U   E   38.1
0001   :U   E   18.3
0002   :U   E   20.4
0003   :SPB LAB1
0004   :U   E   20.6
0005   :S   A   14.1
0006   :SPA LAB2
0007   LAB1 :U   E   20.7
0008   :=   A   15.0
0009   LAB2 :O   E   21.0
000A   :=   A   16.0
000B   :BE
```
Example:

Alternative No. 1
Conversion of PLC Programs

Alternative No. 2
Conversion of PLC Programs

Alternative No. 3
Conversion of PLC Programs

Alternative No. 5

Juniorprofessorship Agentbased Automation
Mohammed Bani Younis
### Binary Example

#### PLC Text

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Content</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>U</td>
<td>E</td>
</tr>
<tr>
<td>0002</td>
<td>U</td>
<td>E</td>
</tr>
<tr>
<td>0004</td>
<td>U</td>
<td>E</td>
</tr>
<tr>
<td>0006</td>
<td>SPB</td>
<td>LAB1</td>
</tr>
<tr>
<td>0008</td>
<td>U</td>
<td>E</td>
</tr>
<tr>
<td>000A</td>
<td>S</td>
<td>A</td>
</tr>
<tr>
<td>000B</td>
<td>SPA</td>
<td>LAB2</td>
</tr>
<tr>
<td>000C</td>
<td>LAB1</td>
<td>U</td>
</tr>
<tr>
<td>000D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000E</td>
<td>LAB2</td>
<td>O</td>
</tr>
<tr>
<td>000F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td>BE</td>
</tr>
</tbody>
</table>

#### Comparison of the transformations

<table>
<thead>
<tr>
<th>Transformation</th>
<th>No. of states</th>
<th>No. of Transitions</th>
<th>State Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternative no. 1 (Single Operations)</td>
<td>28 states</td>
<td>29</td>
<td>CR, PC, Variables, output</td>
</tr>
<tr>
<td>Alternative no. 2 (University of Cachan)</td>
<td>14 states</td>
<td>13</td>
<td>CR, PC, Variables, output</td>
</tr>
<tr>
<td>Alternative no. 3 (Optimization of Alt. 2)</td>
<td>9 states</td>
<td>8</td>
<td>CR, PC, Variables, output</td>
</tr>
<tr>
<td>Alternative no. 4 (Abstraction)</td>
<td>6 states</td>
<td>9</td>
<td>PC, Variables</td>
</tr>
<tr>
<td>Alternative no. 5 (Moore)</td>
<td>9 states</td>
<td>14</td>
<td>CR, PC, Variables output</td>
</tr>
<tr>
<td>Alternative no. 6 (Alt. 4 no contents)</td>
<td>6 states</td>
<td>9</td>
<td>No State contents</td>
</tr>
</tbody>
</table>
## Classification of binary operations

<table>
<thead>
<tr>
<th>Typ 1</th>
<th>Typ 2</th>
<th>Typ 3</th>
<th>Typ 4</th>
<th>Typ 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>1b</td>
<td>1c</td>
<td>2a</td>
<td>2b</td>
</tr>
<tr>
<td>S</td>
<td>SI</td>
<td>SA</td>
<td>U</td>
<td>SPB</td>
</tr>
<tr>
<td>R</td>
<td>SV</td>
<td></td>
<td>UN</td>
<td>BAB</td>
</tr>
<tr>
<td>SPB</td>
<td>SE</td>
<td></td>
<td>O</td>
<td>BEB</td>
</tr>
<tr>
<td>BAB</td>
<td>SA</td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>BEB</td>
<td>ZV</td>
<td></td>
<td>U(</td>
<td></td>
</tr>
<tr>
<td>ZR</td>
<td></td>
<td></td>
<td>O(</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
- Typ 1, Typ 2, Typ 3, Typ 4, and Typ 5 are different types of binary operations.
- The operations are classified based on specific rules indicated in the table.
- The symbols S, R, SPB, BAB, BEB, SI, SV, SE, SA, ZV, ZR, O, U, U(, O(, A, A, X, E, EX are used to denote different operations and conditions.
Fourth and Sixth alternative

Flow Diagram for the abstraction of binary programs
Alternative Nr. 6

### Introduction

- **Re-Engineering**
- **Formalization**
- **Visualization**
- **Re-Implem.**
- **SW-Quality**
- **Case Stud.**
- **Summary**

### Definition

- **Juniorprofessorship Agentbased Automation**
- **Mohammed Bani Younis**

### Merge of diverse instructions to build up a coherent segment

- Elimination of the PC and CR due to the abstraction
- Optimization using IF-THEN-ELSE to minimize the number of the states

### IF-ELSE

- **IF** \( U \ E 38.1 \ U \ E 18.3 \ U \ E 20.4 = 1 \)
- **THEN** Jump to LAB1
- **IF** \( U \ E 20.6 = 1 \)
- **THEN** A 14.1=1
- **Jump to LAB2**

### LAB1

- **IF** \( U \ E 20.7 = 1 \)
- **THEN** A 15.0=1
- **ELSE** A 15.0=0

### LAB2

- **IF** \( O \ E 21.0 = 1 \)
- **THEN** A 16.0=1
- **ELSE** A 16.0=0
- **BE**
Overview

- Visualization concept in a compound re-eng.
- Use of XML as an intermediate step
Formalization of Counters and Timers

- Need for counters and counters
- Counting range 000 up to 999
- 16 bit word for a counter word consists of:
  - State bits → to process the counter
  - Counting Value → real value of the counter

- Set a counter:

```
ZV Z5 IF "VKE"=1 AND Z5,Bit13=0
  THEN Z5,Bit0...9=(Z5,Bit0...9)+1
      AND Z5,Bit13=1
  ELSE IF "VKE"=0
  THEN Z5, Bit 13=0
```

Function block of a counter (Z5)

Need to ask for the state of the Counter using U Z5
### Digital or Non-Binary Programs

- Non-Binary Programs extends Binary to allow other types of controls Data Handling, Numerical Logic, and Lists.
- Abstraction of digital operations to IF-THEN-ELSE Algorithms

<table>
<thead>
<tr>
<th>Type</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Load operation</td>
</tr>
<tr>
<td>2</td>
<td>Transfer operation</td>
</tr>
<tr>
<td>3</td>
<td>Arithmetic operation</td>
</tr>
<tr>
<td>4</td>
<td>Compare operation</td>
</tr>
<tr>
<td>5</td>
<td>Digital logical operation</td>
</tr>
<tr>
<td>6</td>
<td>1s complement operation</td>
</tr>
<tr>
<td>7</td>
<td>2s complement operation</td>
</tr>
<tr>
<td>8</td>
<td>Shift and rotate operation</td>
</tr>
<tr>
<td>9</td>
<td>Jump operation</td>
</tr>
<tr>
<td>10</td>
<td>Other operation</td>
</tr>
</tbody>
</table>

- Transformation of IF-THEN-ELSE Algorithms into Mealy FSM
- Optimization of the Abstraction according to optimization algorithm
Digital or Non-Binary Programs

Example

PLC Code

0001 :L KB0
0002 :T PW138
0003 :L KM0000000010011
0004 :OW
0005 :T PY128
0006 :L KB85
0007 M0 :L KB1
0008 :F
0009 :SPZ= M0
000A M2 :L PY28
000B :T MB225
000C :UN M225.7
000D :SPB= M2
000E :BE

PW138 = KB0
AKKU 1 = KM0000000010011OW KB0
ANZ0 = 0 AND OV = 0
IF AKKU 1 = 0 THEN ANZ1 = 0 ELSE ANZ1 = 1
PY128 = AKKU 1
AKKU 1 = KB 85
M0 AKKU 2 = AKKU 1
AKKU 1 = KB1
AKKU1 = AKKU 2-AKKU 1
IF AKKU1 <= -1
THEN IF AKKU1 < -32768
THEN ANZ1 = 1 AND ANZ0 = 0 AND OV = 1
ELSE ANZ1 = 0 AND ANZ0 = 1 AND OV = 0
ELSE IF AKKU1 >= 1
THEN IF AKKU1 > 32768
THEN ANZ1 = 0 AND ANZ0 = 1 AND OV = 1
ELSE ANZ1 = 1 AND ANZ0 = 0 AND OV = 0
ELSE ANZ1 = 0 AND ANZ0 = 0 AND OV = 0
IF ANZ1 = 0 AND ANZ0 = 0 THEN Jump to M0
M2 MB225 = PY28
IF (N M225.7) = 1 THEN Jump To M2
BE
Steps toward the conversion

→ State Charts as a visualization alternative
Internet Technologies

Fields of Application

- eCommerce
- asset management
- remote engineering
- remote control
- remote maintenance

Languages
- JavaScript
- C#
- XHTML
- Java OS
- SMTP
- WML
- IP
- NNTP
- WML
- COM
- CGI

Protocols
- FTP
- JNI
- RTF
- XPath
- DNS
- HTTP
- TCP
- SMTP
- SOAP
- CORBA
- JAVA

Models
- WWW
- SAX
- DOM
- RMI
- NEWS
- JAVA

Programming Languages
- HTML
- XML
- XSL
- DTD
- XLink
- JNI

Mark-up Languages
- eCl@ss
- eBusiness

Services
- Email
- News
- Email
- Computers

Systems
- WWW
- FTP
- CGI
- CGI

Operating Systems
- JVM
- CORBA
- JAVA

Applications
- eCommerce
- Asset Management
- Remote Engineering
- Remote Control
- Remote Maintenance
XML as a tool for visualization

- **XML** (eXtensible Markup Language)
- XML and HTML
- XML to exchange information across platforms and applications.
- **How to apply XML?**

**Conventional:**
- Lexical Specification
- Grammar + Code for object
- Net generator
  - yacc, bison, cup
  - lex, flex, jflex
  - expat, Xerces

**XML:**
- DTD/ XML-Schema
- (Not) Obligatory
- Object hierarchy
- »DOM«
- W3C Standard
- Scanner
- Parser
- expat, Xerces

- Application-Specific Object Net.
XML as a tool for visualization

- **XSL** *(stylesheet language for XML)* and **XSLT** *(XSL transformation)*
- **XSLT** functions in two steps
  - structural transformation XML $\rightarrow$ structure that reflects the desired output
  - formatting the new structure into the required format, such as HTML or PDF

---

**Diagram:**

1. **XML**
2. **XSL**
3. **XSL-Processor**
4. **DTD/XML Schema**
5. **XML-Editor**
6. **XML-Document**
7. **XSL**
8. **Transformation**
9. **XML-Editor**
10. **XSL-Processor**
11. **DOM (Document Object Model)**
12. **DTD (Document Type Definition)** or **XML-Schema**
13. **HTML File**
14. **PDF File**
15. **Application**
16. **HTML**
17. **XML**
18. **PDF**
XML-Schema for IL (graphical)
XSL for Instruction Identification of STP5

Introduction

Re-Engineering

Formalization

Visualization

Re-Implem.

SW-Quality

Case Stud.

Summary

Outlook
PLC Example

Introduction
Re-Engineering
Formalization
Visualization
Re-Implem.
SW-Quality
Case Stud
Summary
Outlook

**OB 1**

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>SPA PB 1</td>
<td>Jump Absolute to PB 1</td>
</tr>
<tr>
<td>0002</td>
<td>BE</td>
<td></td>
</tr>
</tbody>
</table>

**PB 1**

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>E38.1</td>
<td>AND Operation</td>
</tr>
<tr>
<td>0002</td>
<td>E38.2</td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>E38.1</td>
<td>OR Operation</td>
</tr>
<tr>
<td>0008</td>
<td>E38.3</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>:=</td>
<td>M100.0 at least two Fans running</td>
</tr>
<tr>
<td>0012</td>
<td>UN E38.1</td>
<td>ANDN Operation</td>
</tr>
<tr>
<td>0014</td>
<td>UN E38.2</td>
<td></td>
</tr>
<tr>
<td>0016</td>
<td>UN E38.3</td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td>:=</td>
<td>M100.1 no running Fan</td>
</tr>
<tr>
<td>001A</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>001C</td>
<td>:= O</td>
<td>M100.0 Continuous Light</td>
</tr>
<tr>
<td>001E</td>
<td>:= O</td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>:= M100.1</td>
<td></td>
</tr>
<tr>
<td>0022</td>
<td>:= M99.1</td>
<td>Flashing with 2 Hz</td>
</tr>
<tr>
<td>0024</td>
<td>:=</td>
<td></td>
</tr>
<tr>
<td>0026</td>
<td>:= M100.0</td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td>:= M100.1</td>
<td></td>
</tr>
<tr>
<td>002A</td>
<td>:= M99.2</td>
<td>Flashing with 0.5 Hz</td>
</tr>
<tr>
<td>002C</td>
<td>:=</td>
<td></td>
</tr>
<tr>
<td>002E</td>
<td>:= A42.4</td>
<td>“Active“</td>
</tr>
<tr>
<td>0030</td>
<td>:= A51.7</td>
<td>LCD lamp</td>
</tr>
<tr>
<td>0032</td>
<td>:=</td>
<td>BE</td>
</tr>
</tbody>
</table>
### Visualization Example: HTML Table

| U | Logical Operator |
| U | Logical Operator |
| O | Logical Operator |
| U | Logical Operator |
| U | Logical Operator |
| O | Logical Operator |
| U | Logical Operator |
| U | Logical Operator |

| = | Assignment |
| UN | Logical Operator |
| UN | Logical Operator |
| UN | Logical Operator |

| = | Assignment |
| U( | Logical Operator |
| O | Logical Operator |
| O | Logical Operator |
| U | Logical Operator |

| BE | Special Operation |
**UML - OMG's Unified Modeling Language** - is a graphical language that expresses application requirements analysis and program design in a standard way. Methodology-independent, UML is used by dozens of analysis and design (A&D) tools on the market, making it OMG's most widely used specification.

UML standardizes four types of structural diagrams:
- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

Also five types of behavioral diagrams:
- Use Case diagram
- Sequence diagram
- Collaboration diagram
- Statechart diagram
- Activity diagram

And three types of model management diagrams:
- Package diagram
- Model diagram
- Subsystem diagram

Standardization allows design tools to interchange models using XMI
**XMI**, XML-eXtensible Markup Language, a W3C standard

is an international industry-standard defined by the Object Management Group OMG

is a stream format for interchange of metadata including the UML models that you create during your analysis and design activities

It's useful for transferring the model from one step to the next as your design and coding progress or for transferring from one design tool to another.

because XMI streams models into XML datasets, it also serves as a mapping from UML to XML

SW tools available made it possible to integrate XMI to UML (by import project form XMI or export project to XMI)
XMI_UML Example

<XML timestamp="2000-10-09T17:00:00" verified="true" xmi.version="1.1">
  <XML.header>
    <XML.model xmi.name="SimpleClassModel"/>
    <XML.metamodel xmi.name="UML" xmi.version="1.3"/>
  </XML.header>
</XML>
XMI_UML Example

```
<XMI.content>
  <UML:Class name="Person" xmi.id="Person">
    <UML:Classifier.feature>
      <UML:Attribute name="Name" type="string"/>
    </UML:Classifier.feature>
  </UML:Class>
</XMI.content>
```
<UML:Class name="Firma" xmi.id="Firma">
    <UML:Classifier.feature>
        <UML:Attribute name="Name" type="string"/>
    </UML:Classifier.feature>
</UML:Class>
XMI_UML Example

```
&lt;UML:Association&gt;
  &lt;UML:Association.connection&gt;
    &lt;UML:AssociationEnd name="Arbeitnehmer" type="Person"/&gt;
    &lt;UML:AssociationEnd name="Arbeitgeber" type="Firma"/&gt;
  &lt;/UML:Association.connection&gt;
&lt;/UML:Association&gt;
```
XMI_UML Example

```xml
<uml:AssociationClass name="Arbeitsverhältnis">
  <uml:classifier.feature>
    <uml:Attribute name="Gehalt" multiplicity="1..1" type="money"/>
  </uml:Classifier.feature>
</uml:AssociationClass>
```
UML of OB 1 imported in Together
### Example: Abstraction into IF-THEN-ELSE

<table>
<thead>
<tr>
<th>PB 1</th>
<th>NETZWERK 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000:U</td>
<td>E38.1 //AND Operation</td>
</tr>
<tr>
<td>0002:U</td>
<td>E38.2</td>
</tr>
<tr>
<td>0004:O</td>
<td>//OR Operation</td>
</tr>
<tr>
<td>0006:U</td>
<td>E38.1</td>
</tr>
<tr>
<td>0008:U</td>
<td>E38.3</td>
</tr>
<tr>
<td>00A :</td>
<td>O</td>
</tr>
<tr>
<td>00C :</td>
<td>U E38.2</td>
</tr>
<tr>
<td>00E :</td>
<td>U E38.3</td>
</tr>
<tr>
<td>010:=</td>
<td>M100.0 at least two Fans running</td>
</tr>
<tr>
<td>012:UN</td>
<td>E38.1 // ANDN Operation</td>
</tr>
<tr>
<td>014:UN</td>
<td>E38.2</td>
</tr>
<tr>
<td>016:UN</td>
<td>E38.3</td>
</tr>
<tr>
<td>018:=</td>
<td>M100.1</td>
</tr>
<tr>
<td>01A :</td>
<td>U</td>
</tr>
<tr>
<td>01C :</td>
<td>O</td>
</tr>
<tr>
<td>01E :</td>
<td>O</td>
</tr>
<tr>
<td>020:U</td>
<td>M1</td>
</tr>
<tr>
<td>022:U</td>
<td>M9</td>
</tr>
<tr>
<td>024:O</td>
<td></td>
</tr>
<tr>
<td>026:UN</td>
<td>M1</td>
</tr>
<tr>
<td>028:UN</td>
<td>M1</td>
</tr>
<tr>
<td>02A :</td>
<td>U</td>
</tr>
<tr>
<td>02C :</td>
<td></td>
</tr>
<tr>
<td>02E :</td>
<td>U</td>
</tr>
<tr>
<td>030:=</td>
<td>A5</td>
</tr>
<tr>
<td>032:BE</td>
<td></td>
</tr>
</tbody>
</table>

**IF**

(E 38.1 AND E 38.2) OR (E 38.1 AND E 38.3) OR (E 38.2 AND E 38.3) =1

**THEN**

M 100.0 =1

**ELSE**

M 100.0 =0

**IF**

NOT E 38.1 ANDN E 38.2 ANDN E 38.3 =1

**THEN**

M 100.1 =1

**ELSE**

M 100.1 =0

**IF**

(M 100.0) OR (M 100.1 AND M 99.1) OR (NOT M 100.0 ANDN M 100.1 AND M 99.2) AND A 42.4 =1

**THEN**

A 51.7 =1

**ELSE**

A 51.7 =0

**BE**

Search Instruction ID and Build IF-THEN-ELSE Statements

Build the Automaton (SVG)

SVG of the Automaton
Example: Visualization through XML

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<fsm name="PB001">
  <state name="Si">
    <transition action="NULL" input="?Call PB001" next="S0" />
  </state>
  <state name="S0">
    <transition action="M 100.0=1" input="E38.1 AND E38.2 OR E38.1 AND E38.3 OR E38.2 AND E38.3" next="S1" />
    <transition action="M 100.0=0" input="~ ( E38.1 AND E38.2 OR E38.1 AND E38.3 OR E38.2 AND E38.3 )" next="S1" />
  </state>
  <state name="S1">
    <transition action="M 100.1=1" input="~ E38.1 ANDN E38.2 ANDN E38.3" next="S2" />
    <transition action="M 100.1=0" input="~ ( ~ E38.1 ANDN E38.2 ANDN E38.3 )" next="S2" />
  </state>
  <state name="S2">
    <transition action="A 51.7=1" input="( M100.0 OR M100.1 AND M99.1 OR ~ M100.0 ANDN M100.1 AND M99.2 ) AND A42.4" next="SBE" />
    <transition action="A 51.7=0" input="~ ( M100.0 OR M100.1 AND M99.1 OR ~ M100.0 ANDN M100.1 AND M99.2 ) AND A42.4" next="SBE" />
  </state>
  <state name="SBE">
    <transition action="!Ret PB001" input="NULL" next="Si" />
  </state>
</fsm>
```
Example: Visualization through SVG

**IF**

(E 38.1 AND E 38.2) OR
(E 38.1 AND E 38.3) OR
(E 38.2 AND E 38.3) = 1

**THEN**

M 100.0 = 1

**ELSE**

M 100.0 = 0

**IF**

NOT E 38.1 ANDN E 38.2
ANDN E 38.3 = 1

**THEN**

M 100.1 = 1

**ELSE**

M 100.1 = 0

**IF**

(M 100.0) OR (M 100.1 AND M 99.1) OR (NOT M 100.0 ANDN M 100.1 AND M 99.2)
AND A 42.4 = 1

**THEN**

A 51.7 = 1

**ELSE**

A 51.7 = 0

BE
Example: Visualization through SC

**IF**

(E 38.1 AND E 38.2) OR
(E 38.1 AND E 38.3) OR
(E 38.2 AND E 38.3) =1

**THEN**

M 100.0 =1

**ELSE**

M 100.0 =0

**IF**

NOT E 38.1 ANDN E 38.2
ANDN E 38.3 =1

**THEN**

M 100.1 =1

**ELSE**

M 100.1 =0

**IF**

(M 100.0) OR (M 100.1 AND
M 99.1) OR (NOT M 100.0
ANDN M 100.1 AND M 99.2)
AND A 42.4 =1

**THEN**

A 51.7 =1

**ELSE**

A 51.7 =0

BE
Visualization’s Concluding Remarks

- XML allow the Visualization of the Formalization
- SVG to draw the FSMs
- Extraction of the PLC structure through XMI
- SC as an alternative for the Visualization
- CFSM in XML as a Basis for the Re-Implementation
- XML transformation for deriving the SW-Quality
Overview

Introduction
Re-Engineering
Formalization
Visualization
Re-Implem.
SW-Quality
Case Stud.
Summary
Concept of Re-Implementation

- OB1 → Program in the new PLC
- PB and FB → Function Blocks
- Other OBs → Programs or Function Blocks
- Data Blocks → Array in IEC 61131-3
- Symbol Table → global addressed variables of the inputs, outputs and internal variables
- Other elements in the STEP 5
SW Quality Definition

Introduction
Re-Engineering
Formalization
Visualization
Re-Implem.
SW-Quality
Case Stud.
Summary

ISO 9126

Functionality
- Suitability
- Accurateness
- Interoperability
- Compliance
- Security

Reliability
- Maturity
- Fault Tolerance
- Recoverability

Usability
- Understandability
- Learnability
- Operability

Efficiency
- Time Behavior
- Resource Behavior

Maintainability
- Analyzability
- Changeability
- Stability
- Testability

Portability
- Adaptability
- Installability
- Conformance
- Replaceability
<table>
<thead>
<tr>
<th>Criterion</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>Attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy a stated or implied need.</td>
</tr>
<tr>
<td>Reliability</td>
<td>Set of attributes, that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.</td>
</tr>
<tr>
<td>Usability</td>
<td>Attributes that bear on the effort needed for use, and on the individual evaluation of such use, by a stated or implied set of users.</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Attributes that bear on the relationship between the level of the performance of the software and the amount of resources used, under stated conditions</td>
</tr>
<tr>
<td>Portability</td>
<td>Attributes that bear on the ability of software to be transformed from one environment to another.</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Attributes that bear on the effort needed to make specified modifications</td>
</tr>
</tbody>
</table>
## Evaluation of the Metrics

<table>
<thead>
<tr>
<th>Name</th>
<th>practicability in Software</th>
<th>Usability to IL</th>
<th>Diagnosability Explanatory</th>
<th>Later use for the Diagnosis (online)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>++</td>
<td>++</td>
<td>-(Serves for the coarse appraisal)</td>
<td>--</td>
</tr>
<tr>
<td>Halstead</td>
<td>++</td>
<td>++</td>
<td>0 (Overview about operators and operands)</td>
<td>--</td>
</tr>
<tr>
<td>McCabe Cyclomatic Complexity</td>
<td>+</td>
<td>0</td>
<td>-(Information about conditioned jumps)</td>
<td>--</td>
</tr>
<tr>
<td>Information flow</td>
<td>0</td>
<td>-</td>
<td>0 (Variables flow bet. Blocks)</td>
<td>0</td>
</tr>
<tr>
<td>Tree Impurity</td>
<td>0</td>
<td>+</td>
<td>0 (Graph is necessary)</td>
<td>+</td>
</tr>
<tr>
<td>Coupling</td>
<td>-</td>
<td>-</td>
<td>0 (Variables flow bet. Blocks)</td>
<td>--</td>
</tr>
</tbody>
</table>
Condition Equation Construction

- Basic Principles for the implemented metrics
  1. Search after assignment instructions (S; R; = )
  2. Record the assignment instruction with its relating Variable
  3. Record all Elements that exist between the current and the last assignment instruction
  4. Setting up the Condition Equation and showing it
Condition Equation Construction

U E 12.1
U(
U M 33.1
S M 33.3
U M 33.3
)
U E 13.6
S M 33.4
Implementierung McCabe

\[
\begin{align*}
\text{M33.3} & \quad \text{S} & \quad \text{U} & \quad \text{M33.1} & \quad \text{M33.3} & \quad \text{R} & \quad \text{U} & \quad \text{M32.0} & \quad \text{M33.4} & \quad \text{=} & \quad \text{U} & \quad \text{E12.1} & \quad \text{U} & \quad \text{M33.3}
\end{align*}
\]
\[ m(G) = 0 \]

\[ \Rightarrow \text{Pure Tree Structure; easy Graph} \]
Halstead-Measure

- \( \mu_1 \): number of distinct Operators
- \( \mu_2 \): number distinct Operands
- \( N_1 \): total number of Operator occurrences
- \( N_2 \): total number of Operands occurrences

\[ \mu = \mu_1 + \mu_2 \] : size of the vocabulary
\[ N = N_1 + N_2 \] : implementing length

Volume of the program:
\[ V = N \log_2 \mu \]

\[ \Rightarrow \text{difficulty: } D = \frac{\mu_1}{2} \cdot \frac{N_2}{\mu_2} \]

Effort:
\[ E = V \cdot D \]
McCabe Cyclomatic Complexity

- Flow graphs with $e$ edges and $n$ nodes:
  \[ v(G) = e - n + 2; \]
- Measure for linearly independent paths in $G$: $v(G) = d + 1$
- $d$: number of decisions in $G$
- Change on each operand from state $K$ before the processing of the module to $K+1$ after processing
- Evaluation of the module after:

<table>
<thead>
<tr>
<th>Value</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>An easy program, low risk</td>
</tr>
<tr>
<td>11-20</td>
<td>Complex program, endurable risk</td>
</tr>
<tr>
<td>21-50</td>
<td>Very Complex program, high risk</td>
</tr>
<tr>
<td>&gt;50</td>
<td>Non testable program, extremely high risk</td>
</tr>
</tbody>
</table>

This metric shows how easy/hard to test or maintain a given program or a module
Tree Impurity Implementation

- Complexity determination of the calls between the blocks or modules
  1. Graph formation of the jumps
     - Starting point: current Block
     - End point: unconditional jump (SPA) or conditional jump (SPB) in current Block
     - Conditions for conditional jump are shown on the transitions
  2. Count edges \( n \) and nodes \( e \) of the Graph
  3. Calculate the Tree Impurity:
     \[
     m(G) = \frac{2(e - n + 1)}{(n - 1)(n - 2)} \quad 0 \leq m(G) \leq 1
     \]
  4. If the value tends to Zero this implies it is an easy graph
Didactic Case Study

Description of the MPS (FESTO):
The task of this MPS is Sorting, processing, and Lifting of Cylindrical Pieces of different Materials.
Conversion Process 1

Introduction

Re-Engineering

Formalization

Visualization

Re-Implem.

SW-Quality

Case Stud.

Summary

Converting Platform into IEC 61131:
Symbol Table → Global variables

OB 1 → Program (Main)

PBs → Function Block FBs

FBs → Function Block FBs with instance index

Blocks contains:
- Binary Operations
- Timer and Counters
- Non-Binary
Conversion Process 2

Program structure in UML
### Symbol Table

<table>
<thead>
<tr>
<th>Operand</th>
<th>Symbol</th>
<th>Kommentar</th>
</tr>
</thead>
<tbody>
<tr>
<td>E 0.0</td>
<td>1K1</td>
<td>STEUERUNG AUS/EIN</td>
</tr>
<tr>
<td>E 0.2</td>
<td>1S10</td>
<td>LAMPENTEST</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>A 0.0</td>
<td>1H5</td>
<td>STEUERUNG</td>
</tr>
<tr>
<td>A 0.2</td>
<td>1H7</td>
<td>EINRICHTEN</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>M 0.0</td>
<td>M0.0</td>
<td>VKE = 0 FUER BCD WANDLUNG + VORZEICHEN</td>
</tr>
<tr>
<td>M 0.2</td>
<td>M0.2</td>
<td>RESET STOERMELDUNGEN</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>MB 120</td>
<td>MB120</td>
<td>STOERUNGEN S.WALZE FUER TEXTANZEIGE</td>
</tr>
</tbody>
</table>
S5 to IEC Keywords

<table>
<thead>
<tr>
<th>S5 in FSM</th>
<th>IEC 61131-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>AND</td>
</tr>
<tr>
<td>ANDN</td>
<td>ANDN</td>
</tr>
<tr>
<td>OR</td>
<td>OR</td>
</tr>
<tr>
<td>ORN</td>
<td>ORN</td>
</tr>
<tr>
<td>&lt;, &lt;=</td>
<td>LT, LE</td>
</tr>
<tr>
<td>&gt;, &gt;=</td>
<td>GT, GE</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>NE</td>
</tr>
<tr>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>+</td>
<td>ADD</td>
</tr>
<tr>
<td>-</td>
<td>SUB</td>
</tr>
<tr>
<td>*</td>
<td>MUL</td>
</tr>
<tr>
<td>/</td>
<td>DIV</td>
</tr>
</tbody>
</table>

S5 Keywords in FSM
AND, ANDN, OR, ORN.
<, <=, >, >=, ><, =.
+,-,*,/.
Jump to, Call to.
S5 to IEC 61131 Cycle

STEP5

Start-up Program (OB 21)

Init_PLC := TRUE

IEC 61131

Init_PLC := FLASE

S5 to IEC 61131 Cycle

cyclic Program (OB1)

Read Inputs

Execute User Program

Write Outputs

cyclic Program (OB1)

Read Inputs

Execute User Program

Write Outputs

Start-UP Program (OB 21)
Re-Implementation of Binary PLC programs

- Binary Instructions
  - Direct Converting of FSM (Mealy) into IEC 61131 (Logic und dynamic)
  - Operanden are declared as Global Variables in IEC
  - Variables from Symbol Table
  - FSM Keywords \(\rightarrow\) IEC Instructions

- Example

```xml
<?xml version="1.0" ?>
<fsm name="PBonly_5b">
  <states>
    <state name="S0">
      <transition input="null" next="S1" action="M 100.0= E 38.1 AND E 38.2 OR E 38.1 AND E 38.3 OR E 38.2 AND E 38.3" />
    </state>
  </states>
</fsm>
```

**Example**

```plaintext
VAR
  M100_0: BOOL;
END_VAR
VAR_GLOBAL
  E38_2 AT IX38.2: BOOL;
  E38_1 AT IX38.1: BOOL;
  E38_3 AT IX38.3: BOOL;
END_VAR

S0:
  LD E38_1
  AND E38_2
  OR (True
    AND E38_1
    AND E38_3
  )
  OR (True
    AND E38_2
    AND E38_3
  )
  ST M100_0
  JMP S1

S1:
```

**Implementation of Binary PLC programs**

- Binary Instructions
  - Direct Converting of FSM (Mealy) into IEC 61131 (Logic und dynamic)
  - Operanden are declared as Global Variables in IEC
  - Variables from Symbol Table
  - FSM Keywords \(\rightarrow\) IEC Instructions

**Example**

```xml
<?xml version="1.0" ?>
<fsm name="PBonly_5b">
  <states>
    <state name="S0">
      <transition input="null" next="S1" action="M 100.0= E 38.1 AND E 38.2 OR E 38.1 AND E 38.3 OR E 38.2 AND E 38.3" />
    </state>
  </states>
</fsm>
```
Timer and Counters

- Timer (T) and Counter (C) are
  - Not taken from FSMs (Logic)
  - Treated separately

SV → TP
SE → TON
SA → TOF
Reprogram other T and C types

Example: STEP 5

Richten blinkt

```
[3] UN M 0.1
UN M 0.2
U M 80.0
UN A 0.0
L KT 003.2
SV T 2
NOP 0
NOP 0
NOP 0
U T 2
= A 0.1
***
```

IEC 61131

```
VAR
  M0_1: BOOL;
  M0_2: BOOL;
  M80_0: BOOL;
  T2: TP;
END_VAR

VAR_IN_OUT
  A0_0: BOOL;
END_VAR

S0: LDN M0_1
    ANDN M0_2
    AND M80_0
    A0_0
    ST T2.IN
    CAL T2(PT := T#3000ms)
    JMP S1
S1: LD T2.Q
    ST A0_1
```

Mohammed Bani Younis
Non-Binary PLC programs

- From the FSM (Logik)
- A new Function or function block is generated
- Non-binary instruction in a new Function block according to FSM

Example: STEP 5

```plaintext
[1
NAME: ADD
BEZ : Z1 EW
BEZ : Z2 EW
BEZ : Z3 AW
L =Z1
L KF +800
+F
T =Z3
]
```

Other Variables in the FSM are declared as Global Variables

```
VAR_EXTERNAL
AKKU1 : DINT;
END_VAR

VAR_INPUT
Z2 : DINT;
Z1 : DINT;
END_VAR

VAR_OUTPUT
Z3 : DINT;
END_VAR

VAR
ADDS5 :ADD_S5;
END_VAR
```
Data Blocks

- Transferred to an Array in the main as Global in the first use (Call)
- The corresponding Array index is changed in the after

Example: STEP 5 (PB 1)

IEC 61131

PROGRAM MAIN
VAR_GLOBAL
DB2 :ARRAY [0..255] OF DINT := [0, 128, 130, 1, .......

FUNCTION_BLOCK PB_1
VAR_EXTERNAL
DB2 : ARRAY [0..255] OF DINT;
MW10 : DINT;
END_VAR
VAR
FB2_1 : FB_2;
END_VAR
S0:
(*Call to DB2*)
S1:
CAL
S1FB2:
JMP S2
S2:
LD DB2[5]
ST MW10
SBE:
END_FUNCTION_BLOCK

Introduction
Re-Engineering
Formalization
Visualization
Re-Implem.
SW-Quality
Case Stud.
Summary
Outlook
## Introduction

Re-Engineering

Formalization

Visualization

Re-Implem.

SW-Quality

Case Stud.

Summary

### SW Quality (LOC)

<table>
<thead>
<tr>
<th>Block</th>
<th>LOC (NCSS)</th>
<th>Block</th>
<th>LOC (NCSS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB122</td>
<td>8</td>
<td>PB3</td>
<td>20</td>
</tr>
<tr>
<td>OB1</td>
<td>8</td>
<td>PB4</td>
<td>56</td>
</tr>
<tr>
<td>OB21</td>
<td>177</td>
<td>PB40</td>
<td>9</td>
</tr>
<tr>
<td>PB1</td>
<td>72</td>
<td>PB5</td>
<td>6</td>
</tr>
<tr>
<td>PB10</td>
<td>6</td>
<td>PB50</td>
<td>209</td>
</tr>
<tr>
<td>PB11</td>
<td>5</td>
<td>PB55</td>
<td>99</td>
</tr>
<tr>
<td>PB12</td>
<td>6</td>
<td>PB6</td>
<td>97</td>
</tr>
<tr>
<td>PB15</td>
<td>6</td>
<td>PB7</td>
<td>70</td>
</tr>
<tr>
<td>PB2</td>
<td>104</td>
<td>PB8</td>
<td>261</td>
</tr>
<tr>
<td>PB20</td>
<td>40</td>
<td>PB9</td>
<td>56</td>
</tr>
<tr>
<td>PB21</td>
<td>81</td>
<td><strong>total NCSS:</strong> 1403</td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>Volume</td>
<td>Difficulty</td>
<td>Effort</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>------------</td>
<td>--------</td>
</tr>
<tr>
<td>FB122</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OB1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OB21</td>
<td>2864</td>
<td>1</td>
<td>2864</td>
</tr>
<tr>
<td>PB1</td>
<td>828</td>
<td>8</td>
<td>6624</td>
</tr>
<tr>
<td>PB10</td>
<td>32</td>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>PB11</td>
<td>22</td>
<td>2</td>
<td>44</td>
</tr>
<tr>
<td>PB12</td>
<td>32</td>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>PB15</td>
<td>32</td>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>PB2</td>
<td>998</td>
<td>15</td>
<td>14970</td>
</tr>
<tr>
<td>PB20</td>
<td>357</td>
<td>2</td>
<td>714</td>
</tr>
<tr>
<td>PB21</td>
<td>1237</td>
<td>8</td>
<td>9896</td>
</tr>
<tr>
<td>PB3</td>
<td>104</td>
<td>1</td>
<td>104</td>
</tr>
<tr>
<td>PB4</td>
<td>495</td>
<td>9</td>
<td>4455</td>
</tr>
<tr>
<td>PB40</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PB5</td>
<td>32</td>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>PB50</td>
<td>3281</td>
<td>40</td>
<td>131240</td>
</tr>
<tr>
<td>PB55</td>
<td>862</td>
<td>15</td>
<td>12930</td>
</tr>
<tr>
<td>PB6</td>
<td>1261</td>
<td>12</td>
<td>15132</td>
</tr>
<tr>
<td>PB7</td>
<td>673</td>
<td>6</td>
<td>4038</td>
</tr>
<tr>
<td>PB8</td>
<td>3363</td>
<td>20</td>
<td>67260</td>
</tr>
<tr>
<td>PB9</td>
<td>390</td>
<td>4</td>
<td>1560</td>
</tr>
<tr>
<td>Average</td>
<td>937</td>
<td>8</td>
<td>15115</td>
</tr>
</tbody>
</table>
SW Quality (McCabe)

Introduction
Re-Engineering
Formalization
Visualization
Re-Implem.
SW-Quality
Case Stud.
Summary

OB 21 Segment
SW Quality (Tree Impurity)

Tree Impurity = 0.023391813
Freudenberg (PK14)

Main goal was the diagnosability
Conversion Process 1

Program structure in UML
Conversion Process 2

Introduction
Re-Engineering
Formalization
Visualization
Re-Implen.
SW-Quality
Case Stud.
Summary

Conversion of OB1 segment
## SW Quality (LOC)

<table>
<thead>
<tr>
<th>Block</th>
<th>NCSS</th>
<th>Block</th>
<th>NCSS</th>
<th>Block</th>
<th>NCSS</th>
<th>Block</th>
<th>NCSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB100</td>
<td>3</td>
<td>FB101</td>
<td>3</td>
<td>PB113</td>
<td>15</td>
<td>PB122</td>
<td>131</td>
</tr>
<tr>
<td>FB140</td>
<td>10</td>
<td>FB141</td>
<td>10</td>
<td>PB123</td>
<td>46</td>
<td>PB133</td>
<td>55</td>
</tr>
<tr>
<td>FB150</td>
<td>10</td>
<td>FB151</td>
<td>6</td>
<td>PB14</td>
<td>386</td>
<td>PB141</td>
<td>76</td>
</tr>
<tr>
<td>FB152</td>
<td>6</td>
<td>FB153</td>
<td>63</td>
<td>PB15</td>
<td>57</td>
<td>PB150</td>
<td>62</td>
</tr>
<tr>
<td>FB154</td>
<td>21</td>
<td>FB240</td>
<td>2</td>
<td>PB151</td>
<td>30</td>
<td>PB16</td>
<td>314</td>
</tr>
<tr>
<td>FB241</td>
<td>2</td>
<td>FB242</td>
<td>2</td>
<td>PB18</td>
<td>173</td>
<td>PB19</td>
<td>1</td>
</tr>
<tr>
<td>FB243</td>
<td>2</td>
<td>FB244</td>
<td>2</td>
<td>PB2</td>
<td>50</td>
<td>PB20</td>
<td>93</td>
</tr>
<tr>
<td>FB245</td>
<td>2</td>
<td>FB246</td>
<td>2</td>
<td>PB21</td>
<td>179</td>
<td>PB22</td>
<td>79</td>
</tr>
<tr>
<td>FB247</td>
<td>2</td>
<td>FB248</td>
<td>2</td>
<td>PB250</td>
<td>479</td>
<td>PB251</td>
<td>13</td>
</tr>
<tr>
<td>FB249</td>
<td>2</td>
<td>FB250</td>
<td>2</td>
<td>PB252</td>
<td>5</td>
<td>PB3</td>
<td>43</td>
</tr>
<tr>
<td>FB251</td>
<td>2</td>
<td>FB252</td>
<td>27</td>
<td>PB31</td>
<td>113</td>
<td>PB33</td>
<td>34</td>
</tr>
<tr>
<td>FB253</td>
<td>14</td>
<td>FB254</td>
<td>29</td>
<td>PB4</td>
<td>17</td>
<td>PB43</td>
<td>39</td>
</tr>
<tr>
<td>FB255</td>
<td>30</td>
<td>OB1</td>
<td>39</td>
<td>PB44</td>
<td>39</td>
<td>PB51</td>
<td>18</td>
</tr>
<tr>
<td>OB13</td>
<td>2</td>
<td>OB21</td>
<td>3</td>
<td>PB53</td>
<td>8</td>
<td>PB63</td>
<td>80</td>
</tr>
<tr>
<td>OB22</td>
<td>3</td>
<td>OB31</td>
<td>2</td>
<td>PB71</td>
<td>20</td>
<td>PB72</td>
<td>24</td>
</tr>
<tr>
<td>PB1</td>
<td>36</td>
<td>PB10</td>
<td>209</td>
<td>PB73</td>
<td>63</td>
<td>PB80</td>
<td>62</td>
</tr>
<tr>
<td>PB103</td>
<td>43</td>
<td>PB11</td>
<td>23</td>
<td>PB82</td>
<td>20</td>
<td>PB83</td>
<td>34</td>
</tr>
<tr>
<td>PB93</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**total NCSS**: 3493
### SW Quality (Halstead)

<table>
<thead>
<tr>
<th>Block</th>
<th>Volume</th>
<th>Difficulty</th>
<th>Effort</th>
<th>Block</th>
<th>Volume</th>
<th>Difficulty</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB140</td>
<td>36.0</td>
<td>6.0</td>
<td>216.0</td>
<td>PB16</td>
<td>3847.0</td>
<td>18.0</td>
<td>69246.0</td>
</tr>
<tr>
<td>FB141</td>
<td>36.0</td>
<td>6.0</td>
<td>216.0</td>
<td>PB18</td>
<td>1939.0</td>
<td>15.0</td>
<td>29085.0</td>
</tr>
<tr>
<td>PB1</td>
<td>337.0</td>
<td>5.0</td>
<td>1685.0</td>
<td>PB2</td>
<td>456.0</td>
<td>4.0</td>
<td>1824.0</td>
</tr>
<tr>
<td>PB10</td>
<td>2246.0</td>
<td>16.0</td>
<td>35936.0</td>
<td>PB20</td>
<td>801.0</td>
<td>10.0</td>
<td>8010.0</td>
</tr>
<tr>
<td>PB103</td>
<td>342.0</td>
<td>2.0</td>
<td>684.0</td>
<td>PB21</td>
<td>2077.0</td>
<td>9.0</td>
<td>18693.0</td>
</tr>
<tr>
<td>PB11</td>
<td>172.0</td>
<td>4.0</td>
<td>688.0</td>
<td>PB22</td>
<td>656.0</td>
<td>2.0</td>
<td>1312.0</td>
</tr>
<tr>
<td>PB113</td>
<td>91.0</td>
<td>2.0</td>
<td>182.0</td>
<td>PB250</td>
<td>6648.0</td>
<td>14.0</td>
<td>93072.0</td>
</tr>
<tr>
<td>PB122</td>
<td>1381.0</td>
<td>15.0</td>
<td>20715.0</td>
<td>PB3</td>
<td>321.0</td>
<td>3.0</td>
<td>963.0</td>
</tr>
<tr>
<td>PB123</td>
<td>407.0</td>
<td>3.0</td>
<td>1221.0</td>
<td>PB31</td>
<td>899.0</td>
<td>7.0</td>
<td>6293.0</td>
</tr>
<tr>
<td>PB133</td>
<td>677.0</td>
<td>3.0</td>
<td>2031.0</td>
<td>PB33</td>
<td>221.0</td>
<td>6.0</td>
<td>1326.0</td>
</tr>
<tr>
<td>PB14</td>
<td>4398.0</td>
<td>30.0</td>
<td>131940.0</td>
<td>PB34</td>
<td>143.0</td>
<td>3.0</td>
<td>429.0</td>
</tr>
<tr>
<td>PB141</td>
<td>657.0</td>
<td>1.0</td>
<td>657.0</td>
<td>PB43</td>
<td>296.0</td>
<td>5.0</td>
<td>1480.0</td>
</tr>
<tr>
<td>PB15</td>
<td>549.0</td>
<td>8.0</td>
<td>4392.0</td>
<td>PB44</td>
<td>296.0</td>
<td>5.0</td>
<td>1480.0</td>
</tr>
<tr>
<td>PB150</td>
<td>540.0</td>
<td>10.0</td>
<td>5400.0</td>
<td>PB51</td>
<td>135.0</td>
<td>8.0</td>
<td>1080.0</td>
</tr>
<tr>
<td>PB151</td>
<td>253.0</td>
<td>4.0</td>
<td>1012.0</td>
<td>PB53</td>
<td>42.0</td>
<td>3.0</td>
<td>126.0</td>
</tr>
<tr>
<td>PB80</td>
<td>564.0</td>
<td>12.0</td>
<td>6768.0</td>
<td>PB63</td>
<td>792.0</td>
<td>5.0</td>
<td>3960.0</td>
</tr>
<tr>
<td>PB82</td>
<td>154.0</td>
<td>3.0</td>
<td>462.0</td>
<td>PB71</td>
<td>167.0</td>
<td>3.0</td>
<td>501.0</td>
</tr>
<tr>
<td>PB83</td>
<td>265.0</td>
<td>4.0</td>
<td>1060.0</td>
<td>PB72</td>
<td>188.0</td>
<td>3.0</td>
<td>564.0</td>
</tr>
<tr>
<td>PB93</td>
<td>131.0</td>
<td>2.0</td>
<td>262.0</td>
<td>PB73</td>
<td>585.0</td>
<td>4.0</td>
<td>2340.0</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>888.03</td>
<td>6.92</td>
<td>12034.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# SW Quality (McCabe)

<table>
<thead>
<tr>
<th>Block</th>
<th>Value</th>
<th>Risk</th>
<th>Block</th>
<th>Value</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB140</td>
<td>2</td>
<td>easy Program, low Risk</td>
<td>PB16</td>
<td>60</td>
<td>untestable Program, extremely high Risk</td>
</tr>
<tr>
<td>PB10</td>
<td>38</td>
<td>Very Complex Program, High Risk</td>
<td>PB18</td>
<td>44</td>
<td>Very Complex Program, High Risk</td>
</tr>
<tr>
<td>PB44</td>
<td>9</td>
<td>easy Program, low Risk</td>
<td>PB2</td>
<td>14</td>
<td>Complex Program, endurable Risk</td>
</tr>
<tr>
<td>PB4</td>
<td>8</td>
<td>easy Program, low Risk</td>
<td>PB20</td>
<td>11</td>
<td>Complex Program, endurable Risk</td>
</tr>
<tr>
<td>PB103</td>
<td>2</td>
<td>easy Program, low Risk</td>
<td>PB21</td>
<td>6</td>
<td>easy Program, low Risk</td>
</tr>
<tr>
<td>PB11</td>
<td>9</td>
<td>easy Program, low Risk</td>
<td>PB22</td>
<td>2</td>
<td>easy Program, low Risk</td>
</tr>
<tr>
<td>PB113</td>
<td>2</td>
<td>easy Program, low Risk</td>
<td>PB3</td>
<td>2</td>
<td>easy Program, low Risk</td>
</tr>
<tr>
<td>PB122</td>
<td>20</td>
<td>Very Complex Program, High Risk</td>
<td>PB250</td>
<td>132</td>
<td>untestable Program, extremely high Risk</td>
</tr>
<tr>
<td>PB123</td>
<td>2</td>
<td>easy Program, low Risk</td>
<td>PB31</td>
<td>18</td>
<td>Complex Program, endurable Risk</td>
</tr>
<tr>
<td>PB133</td>
<td>6</td>
<td>easy Program, low Risk</td>
<td>PB33</td>
<td>15</td>
<td>Complex Program, endurable Risk</td>
</tr>
<tr>
<td>PB14</td>
<td>55</td>
<td>untestable Program, extremely high Risk</td>
<td>PB141</td>
<td>64</td>
<td>untestable Program, extremely high Risk</td>
</tr>
<tr>
<td>FB141</td>
<td>2</td>
<td>easy Program, low Risk</td>
<td>PB43</td>
<td>9</td>
<td>easy Program, low Risk</td>
</tr>
<tr>
<td>PB15</td>
<td>12</td>
<td>Complex Program, endurable Risk</td>
<td>PB80</td>
<td>14</td>
<td>Complex Program, endurable Risk</td>
</tr>
<tr>
<td>PB150</td>
<td>3</td>
<td>easy Program, low Risk</td>
<td>PB51</td>
<td>8</td>
<td>easy Program, low Risk</td>
</tr>
<tr>
<td>PB151</td>
<td>5</td>
<td>easy Program, low Risk</td>
<td>PB53</td>
<td>2</td>
<td>easy Program, low Risk</td>
</tr>
<tr>
<td>PB1</td>
<td>10</td>
<td>Complex Program, endurable Risk</td>
<td>PB63</td>
<td>10</td>
<td>Complex Program, endurable Risk</td>
</tr>
<tr>
<td>PB82</td>
<td>5</td>
<td>easy Program, low Risk</td>
<td>PB71</td>
<td>5</td>
<td>easy Program, low Risk</td>
</tr>
<tr>
<td>PB83</td>
<td>5</td>
<td>easy Program, low Risk</td>
<td>PB72</td>
<td>7</td>
<td>easy Program, low Risk</td>
</tr>
<tr>
<td>PB93</td>
<td>2</td>
<td>easy Program, low Risk</td>
<td>PB73</td>
<td>5</td>
<td>easy Program, low Risk</td>
</tr>
</tbody>
</table>
Tree Impurity $= -6.4935064 \times 10^{-4}$

$\rightarrow$ The negative value makes it clear that the graph of the tree impurity consists of more than one tree structure.

Tree Impurity segment of PK14
Re-Eng. of PLC programs requires a Model

PLC program were modeled as CFSMs

PLC code was transformed to FSM after the optimization using IF-THEN-ELSE

UML and XML made it possible to get a model of the PLC

Re-implementation of the existing PLCs from the existing structure and formal description \text{\(\rightarrow\)} IEC 61131

SW Quality derivation of the PLC program
Discussion
Oops clicked firmly 😊
Additional Slides
PLC as Discrete Event System

• $\text{PLC}_{\text{system}}/\rho \rightarrow \text{Closed loop}$

plant as a FSM $\rho = \langle X, \Sigma, X_0, X_f, \delta \rangle$ where

• $X$ Finite set of states
• $X_0 \subseteq X$ set of initial states
• $X_f \subseteq X$ set of final states also marked or accepted) states of $\rho$
• $\Sigma$ finite alphabet of $\rho$
• $\delta$ partial transition function mapping $X \times \Sigma$ to $X$

$\rightarrow \delta(x_i, e)$ an event $e \in \Sigma$ leads $x_i \in X$ to state $x_j \in X$
PLC as Discrete Event System

• **PLC**$_{\text{system}}$ as a tuple $\langle \text{PLC}_{\text{SW}}, \text{PLC}_{\text{HW}}, \text{PLC}_{\text{Cycle}} \rangle$

• **PLC**$_{\text{SW}}$ PLC program as tuple

  $\langle \text{PAE}, \text{PAA}, I, A_{\text{PAE}}, \text{PLC}_{\text{pr}}, x_0, x_f \rangle$

• **PAE** non empty finite ordered set of binary inputs

• **PAA** non empty finite ordered set of binary outputs

• **I** non empty finite ordered set of binary internal variables of PLC

• $\alpha(I)$ as Cartesian product $\{0,1\}^{\mid I\mid}$ which is the alphabet generated by the nonempty ordered set of variables $I$

• **PLC**$_{\text{pr}}$ is the PLC program described as a partial function

  \[\text{PLC}_{\text{pr}}(x, e) : \alpha(I) \times \alpha(PAA) \times A_{\text{PAE}} \rightarrow \alpha(I) \times \alpha(PAA)\]

  where $x \in \alpha(A_{\text{PAE}}) \times \alpha(PAA)$ and $e \in A_{\text{PAE}}$ and $x_0$, an initial state of the PLC program such that $x_0 \in \alpha(I) \times \alpha(PAA) = \{0,1\}^{\mid I\mid + \mid PAA\mid}$

• $A_{\text{PAE}} \subseteq \subseteq \text{(PAE)}$ of recognized inputs
PLC Blocks as FSM

Introduction

- **PLC**_M Module or Block as a stand alone is a tuple \( \langle S, \sum, Y, \delta, \lambda, s_0, s_f \rangle \)
- **S** set of states
- **\( \sum = \alpha(\text{PAE}) \)** input alphabet

- **\( b \in \text{Binary} \)** range over binary variables
- **\( bexpr \)** is derived which ranges over Boolean expressions
- **\( bexpr \in \text{Bexpr}^+ \)** where \( \text{Bexpr}^+ \) is the language generated by \( \text{Gexpr} \) grammar

\[
\text{Gexpr} = 1 | 0 | b | \sim b | \sim \sim \sim (\text{Gexpr}) \\
\text{Gexpr} \equiv \text{Gexpr} \equiv \text{Gexpr} \equiv \text{Gexpr}
\]

\( \rightarrow \text{Gexpr} \) is also an alphabet since \( \Delta \text{\&} \equiv \text{\&} \) are called closed binary on \( \text{pow}(\cup) \) where \( \cup \) is the universe
• \( PLC_{SW} \) is a two subsets \( PLC_u \) and \( PLC_{SYS} \)
• \( PLC_u \) is Re-engineering relevant
• \( PLC_u \) is a model of CFSM \( PLC_{M1} \ldots PLC_{Mn} \) of \( \langle S_i, \sum_i, Y_i, \delta_i, \lambda_i, s_{0,i} \rangle \)
• The model \( PLC_{Mi} \ \forall i \in \{1, \ldots, n\} \) \( PLC_{M1} \otimes PLC_{M2} \ldots \otimes PLC_{Mn} \) builds the automaton \( PLC_u := \uparrow S, \bullet, Y, \delta, \lambda, s_0 \uparrow \) such that in case of no Sync.

General feed-forward composition

\[
Y_i = Y_{i1} \times Y_{i2}
\]

\[
\sum_i = \sum_{i1} \times \sum_{i2}
\]
Industrielle Fallstudie

Einführung

Re-Engineering

Formalisierung

Visualisierung

Re-Implementierung

SW-Qualität

Fallstudien

Zusammenfassung
Implementation of the UML Activity diagrams

Application of the method to other PLC proprietary languages

Re-Implementation into new Systems (IEC 61499)

Extension of the SWQ to the dynamic of the PLC program
Structure and Complexity Metrics

• Size Metric
  – Lines of Code (LOC)
  – Non-Commented Source Statements (NCSS)

• Halstead-Measure
  – Calculation through operands and operators of:
    - implementing length
    - size of the vocabulary
    - Volume of the program
    - Difficulty and Effort

• McCabe Cyclomatic Complexity Measure
  – Calculation through Flow graph with e edges and n nodes:
    \[ v(G) = e - n + 2; \]

• Tree Impurity:
  \[ m(G) = \frac{2(e - n + 1)}{(n - 1)(n - 2)} \quad 0 \leq m(G) \leq 1 \]

★★ If the value tends to zero, this implies it is an easy graph