TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Re-Engineering Approach for PLC Programs based on Formal Methods

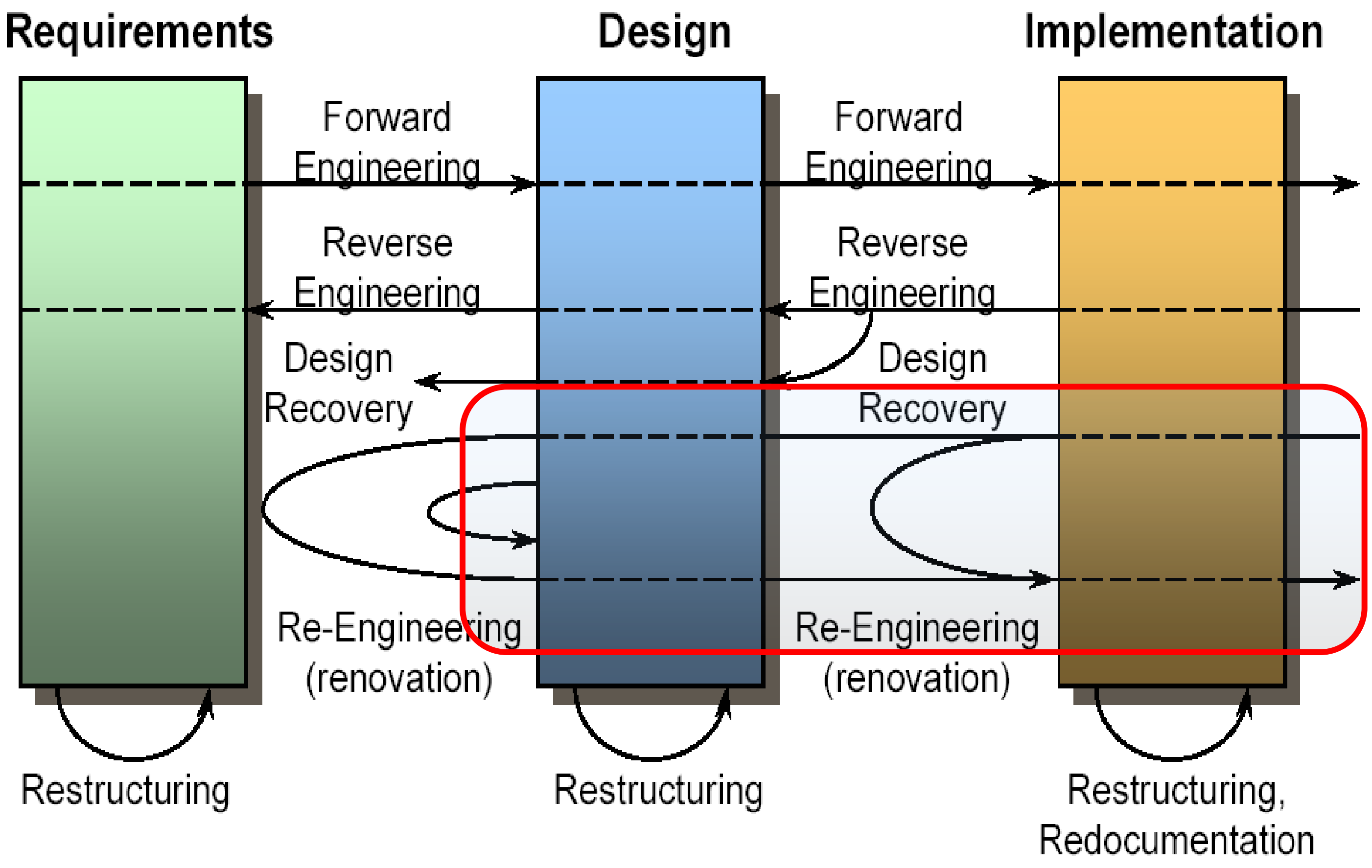
Mohammed Bani Younis

Contents

PLC Programs

- 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?

- 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 → 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

→Logical dynamic is not converted

- STEP 5 → 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

→only Instruction mapping (no Logic)

Re-Engineering based on Formal Methods

Reference	Classification				
	Source		Level	Aim	Model
	Lang.	Additional			
[Storr and Kraneis, 1997]	IL	Plant	Program	Re-Eng.	Automaton
[Treseler et al., 2000]	IL	Plant	Program	Verification	Automaton
[Bornot et al., 2000 (b)]	SFC	Without	Program	Verification	SMV Input Code
[Willems, 1999]	IL	Plant	Program	Verification	Timed Automaton
[Mader and Wupper, 1999]	IL	Without	Algorithm	Verification	Timed Automaton
[Brinksma and Mader, 2000]	SFC	Plant	Program	Verification	SPIN model
[Kowalewski et al., 1999]	SFC	Plant	Program	Static analysis	Automaton
[Bornot et al., 2000 (a)]	IL	Without	Program	Verification	No model
[Canet et al., 2000]	IL	Without	Program	Verification	Automaton
[Roussel and Lesage, 1996]	SFC	Without	Program	Verification	FSM
[Lampérière-Couffin et al., 1999]	SFC&LD	Without	Program	Verification	Automaton
[Mertke and Menzel, 2000]	IL	Plant	Program	Verification	PN
[Hassapis et al., 1998]	SFC	Plant	Program	Verification	hybrid Automaton
[Rossi and Schnoebelen, 2000]	LD	Without	Program	Verification	FSM
[Baresi et al., 2000]	FBD	Without	Algorithm	Verification	PN
[Hatono and Baba, 1996]	LD	Without	Program	Verification	PN
[Vyatkin and Hanisch, 2000]	FBD	Plant	Program	Verification	SNS
[Canet, 2001]	ST	Without	Algorithm	Verification	Automata

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Re-Engineering based on OOP and Internet

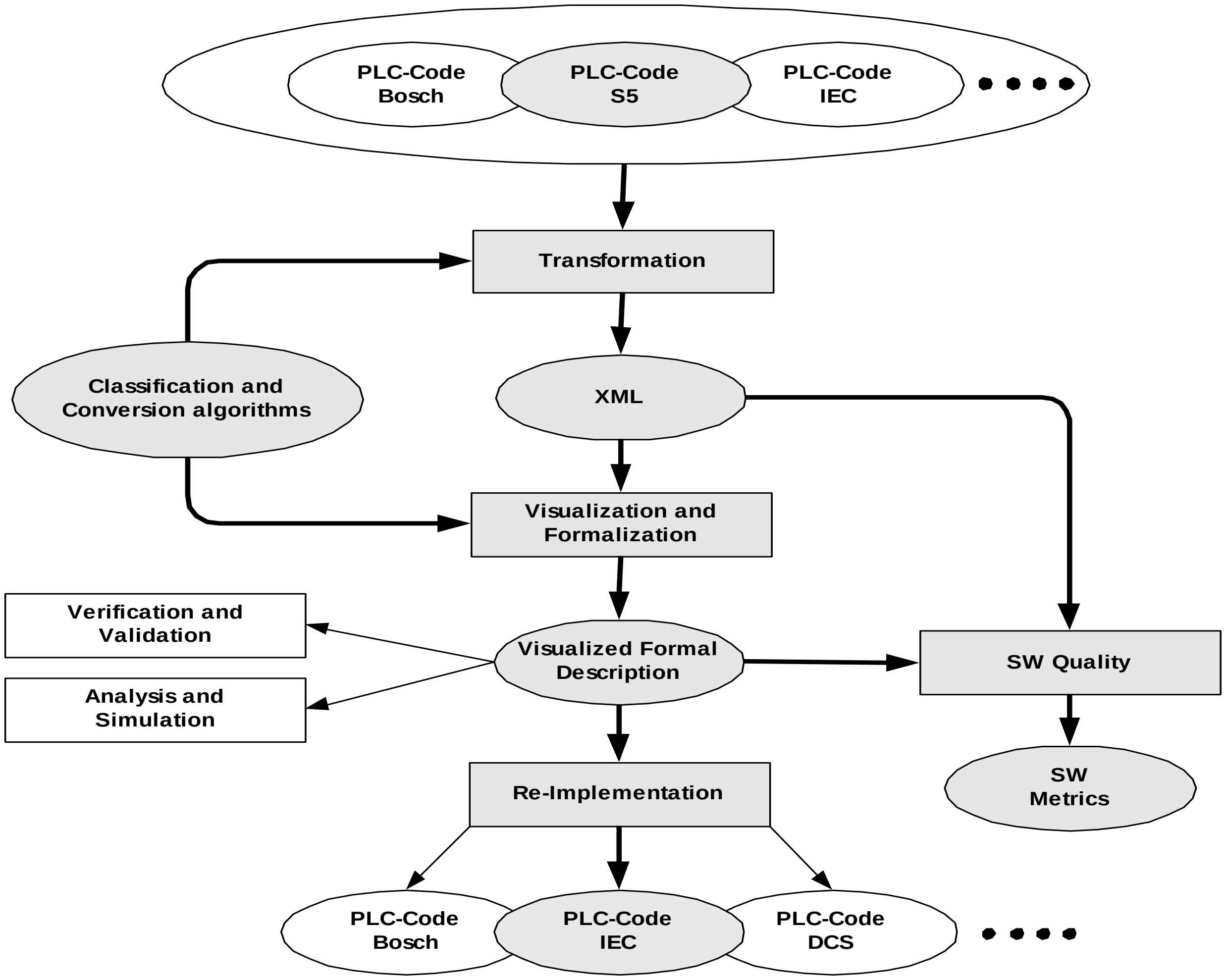
- **Internet and OO Re-Engineering**
  - New trend in controller design
  - Majority of the works are Forward Engineering
  - Evaluation of OOP delegated through Unified Modeling Language (UML)
  - Use of UML as modeling environment
  - Use of Internet Technologies (XML, HTML, XSL, etc…)

→ Research against Industry

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Compound Re-Engineering

- Compound of OO, Internet, and formal methods

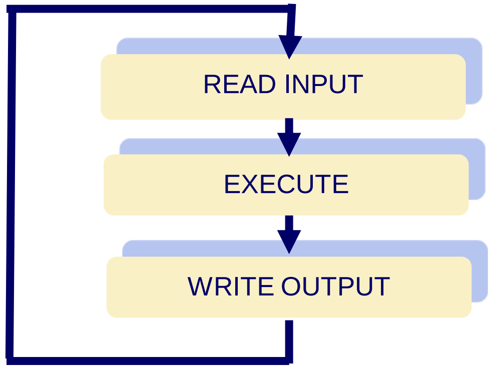


Done

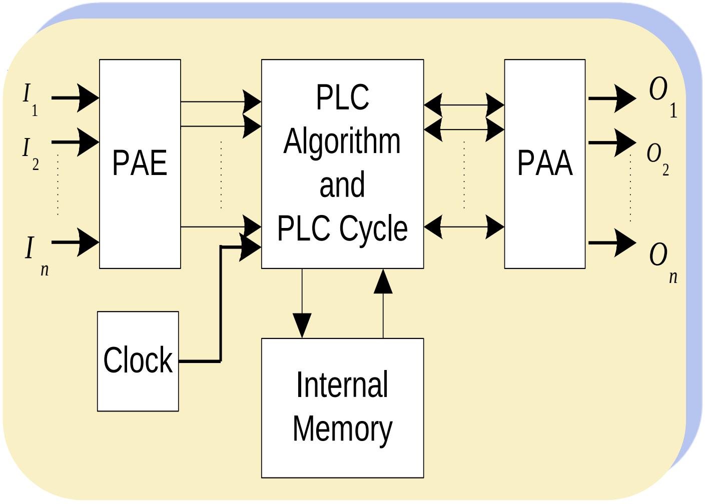
PLC and STEP5

- STEP 5 in a hierarchy-like form:
  - o *OB:* Organization Module
  - o *PB:* Program Module
  - o *DB:* Data Module
  - o *FB:* Function Module
- Timers and Counters
- Polling mode operation
- PLC Memory

PLC Cycle



PLC Memory

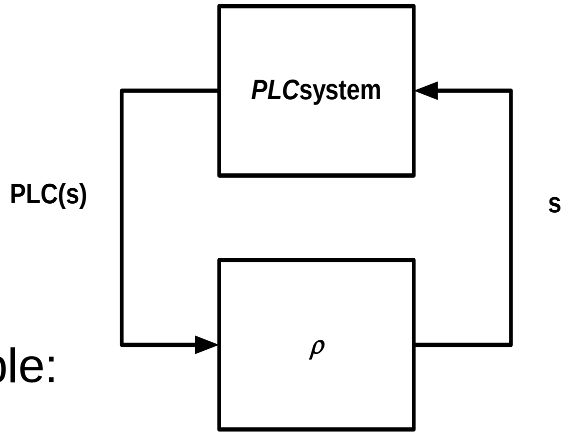


PLC as Discrete Event System

- *PLC_{system}/ρ → closed loop*
- plant as a FSM: $\rho = \langle S, \Sigma, X_0, X_f, \delta \rangle$
- *PLC_{system}* as a tuple $\langle PLC_{\text{SW}}, PLC_{\text{HW}}, PLC_{\text{Cycle}} \rangle$
- *PLC_{SW}* which denotes the PLC program as tuple:

$$\langle PAE, PAA, I, A_{PAE}, PLC_{\rho r}, x_0, x_f \rangle$$

- *PLC_M* module or block as a stand alone is a tuple: $\langle S, \Sigma, Y, \delta, \lambda, s_0, s_f \rangle$
- *S* set of states
- $Y = \alpha(PAA)$ output alphabet
- $\delta : S \times \Sigma \to S$ transition function
- $\Sigma = \alpha(PAE)$ input alphabet
- $\lambda : S \times \Sigma \to Y$ output function

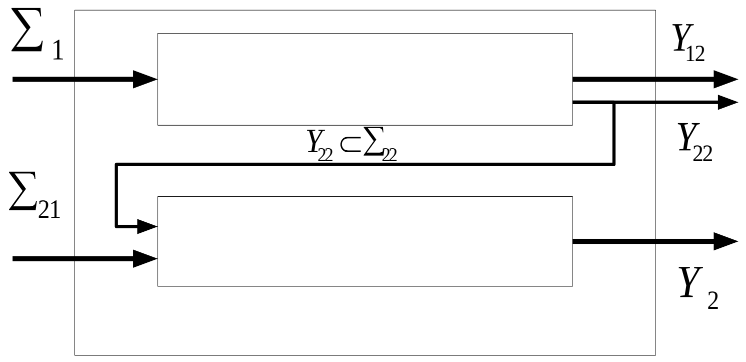


PLC Program as Communicating Automata

- *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*$_{M1}$…..*PLC*$_{Mn}$ of $\langle S_i, \sum_i, Y_i, \delta_i, \lambda_i, s_{0,i} \rangle$
- The model *PLC*$_{Mi}$ $\forall i \in \{1,....,n\}$ PLC$_{M1}$⊗PLC$_{M2}$ ⊗…⊗PLC$_{Mn}$ builds the automaton PLC$_u$:= $\langle S, \sum, Y, \delta, \lambda, s_0 \rangle$

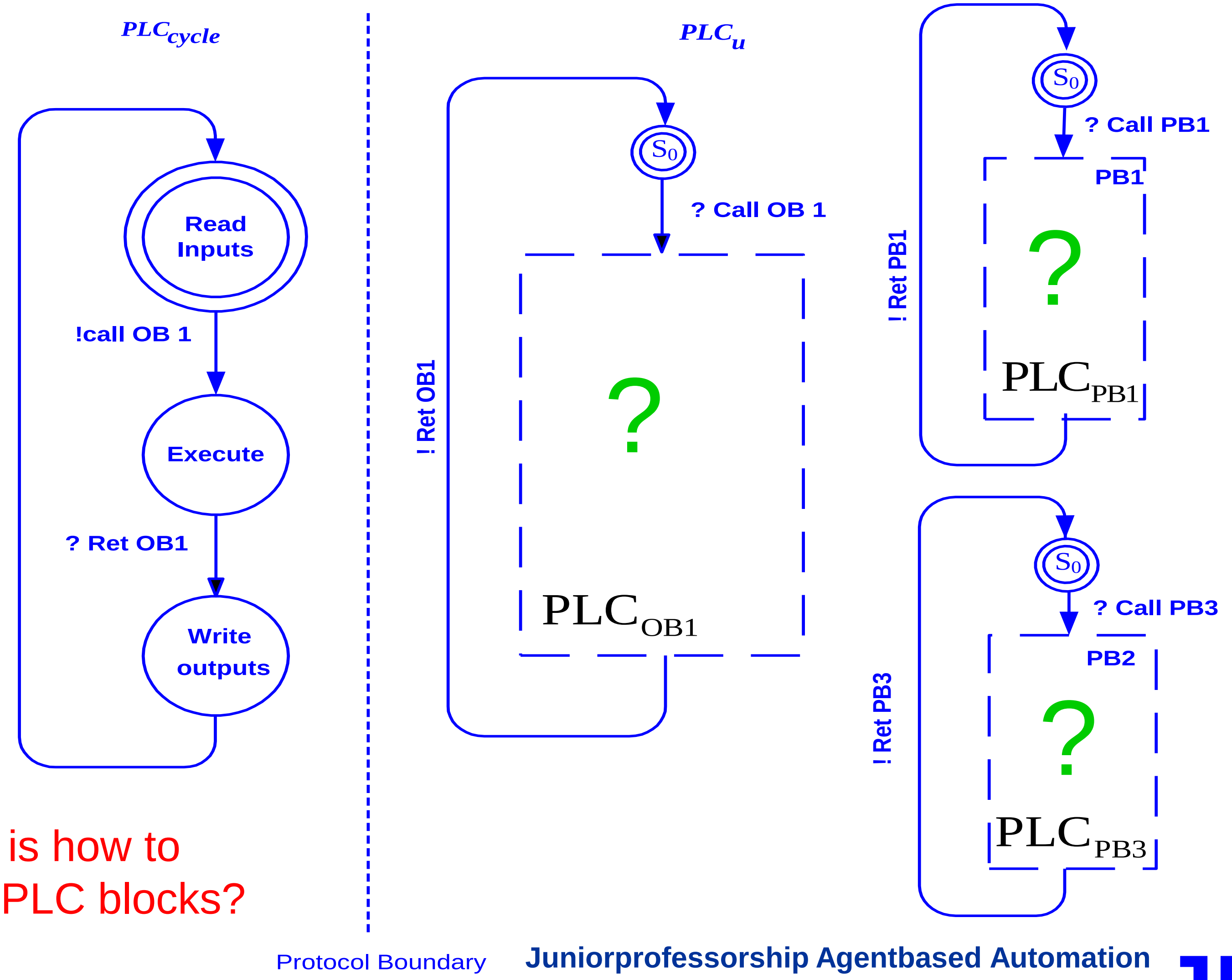
General feed-forward composition



$$Y_i = Y_{i1} \times Y_{i2}$$
$$\sum_i = \sum_{i1} \times \sum_{i2}$$

PLC Program and Cycle

- *PLC*$_{\text{Cycle}}$ as CFSM with the CFSMs of *PLC*$_{u}$
- Example $PLC_{OB1} \otimes PLC_{PB1} \otimes PLC_{PB3}$



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 → State = f (VKE, PC, internal variables)
2. the conversion of the program according to University of Cachan → 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 → State = f (PC, variables)
5. Conversion to Moore machine
6. Based on 4, no need for state contents

Conversion of PLC Programs

PLC Text

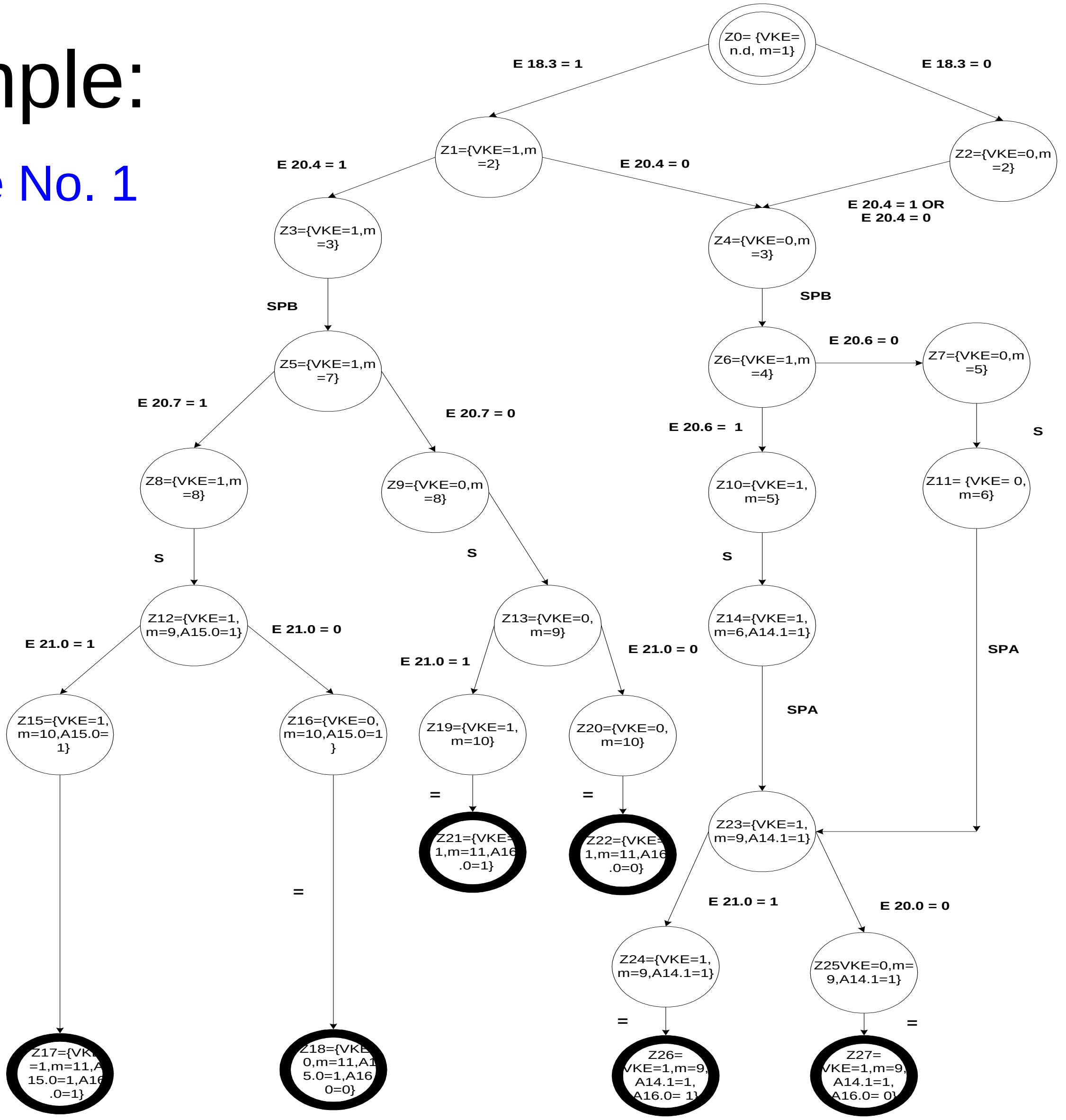
Example:

```
Kommentar :
Autor     :
Erstellt  :15.07.2003  Geaendert 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
```

Conversion of PLC Programs

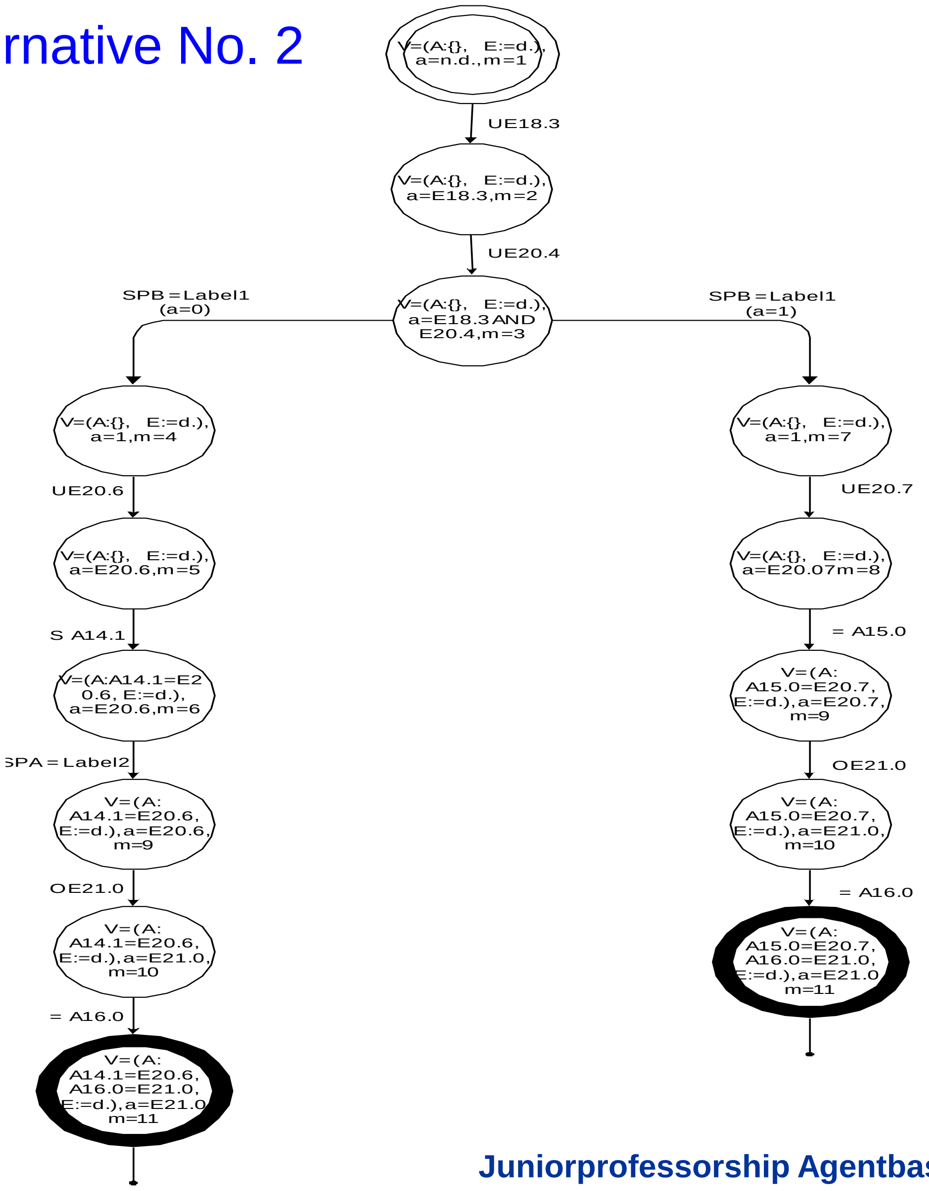
Example:

Alternative No. 1



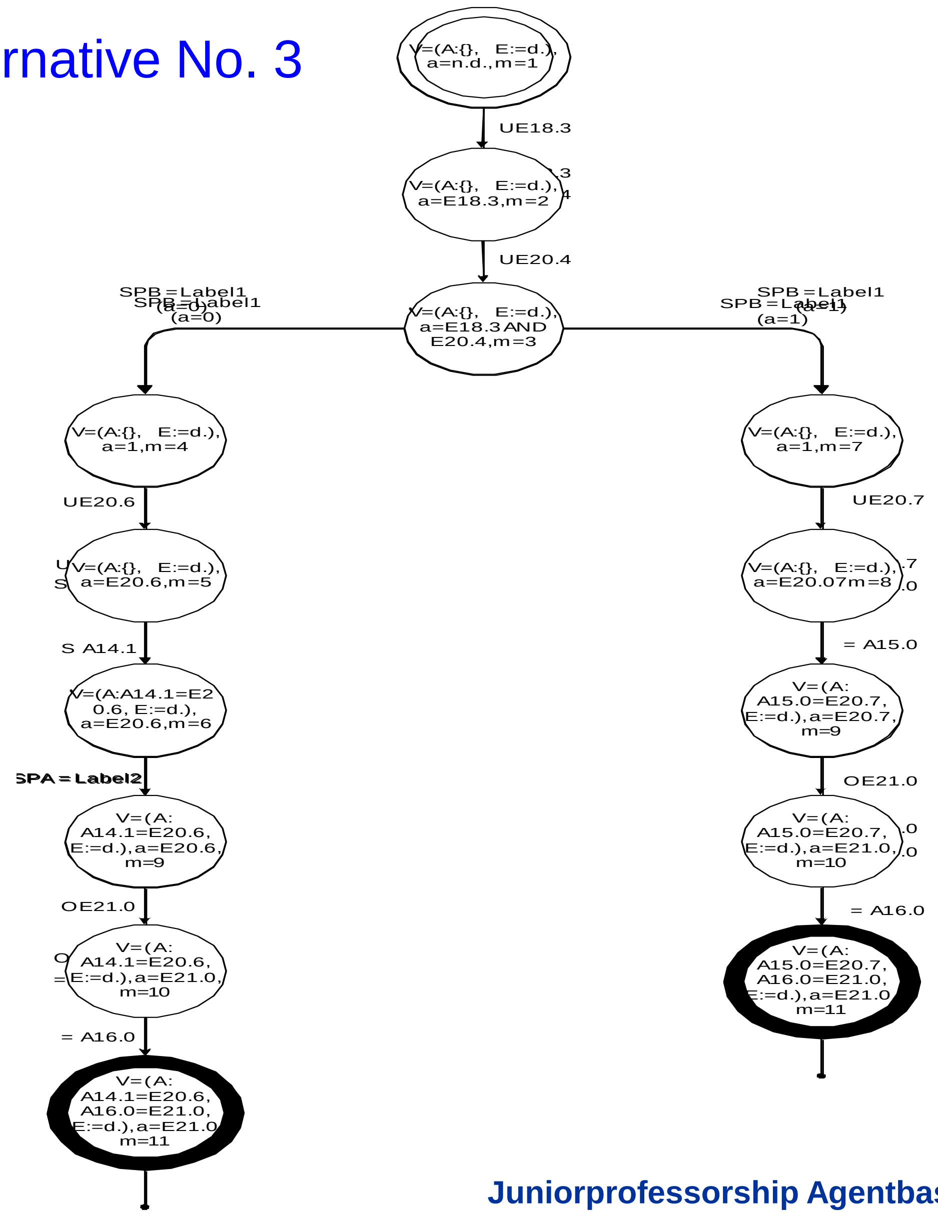
Conversion of PLC Programs

Alternative No. 2


V=(A:{}, E:=d.), a=n.d.,m=1
UE18.3
V=(A:{}, E:=d.), a=E18.3,m=2
UE20.4
V=(A:{}, E:=d.), a=E18.3 AND E20.4,m=3
SPB=Label1 (a=0)
SPB=Label1 (a=1)
V=(A:{}, E:=d.), a=1,m=4
UE20.6
V=(A:{}, E:=d.), a=E20.6,m=5
S A14.1
V=(A:A14.1=E20.6, E:=d.), a=E20.6,m=6
SPA=Label2
V=(A: A14.1=E20.6, E:=d.),a=E20.6, m=9
OE21.0
V=(A: A14.1=E20.6, E:=d.),a=E21.0, m=10
= A16.0
V=(A: A14.1=E20.6, A16.0=E21.0, E:=d.),a=E21.0 m=11
V=(A:{}, E:=d.), a=1,m=7
UE20.7
V=(A:{}, E:=d.), a=E20.07m=8
= A15.0
V=(A: A15.0=E20.7, E:=d.),a=E20.7, m=9
OE21.0
V=(A: A15.0=E20.7, E:=d.),a=E21.0, m=10
= A16.0
V=(A: A15.0=E20.7, A16.0=E21.0, E:=d.),a=E21.0 m=11

Conversion of PLC Programs

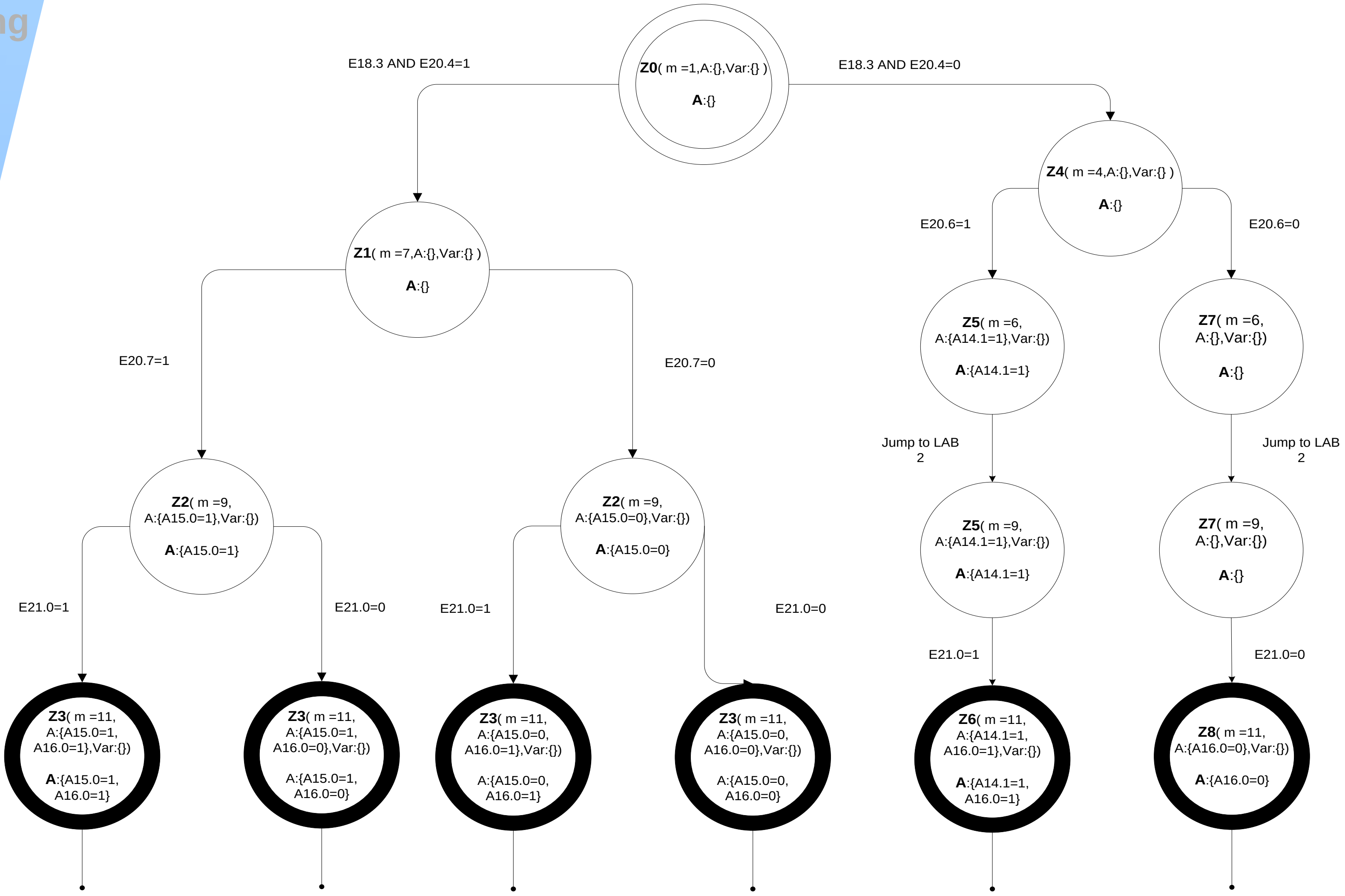
Alternative No. 3


V=(A:{}, E:=d.), a=n.d.,m=1
UE18.3
V=(A:{}, E:=d.), a=E18.3,m=2
UE20.4
V=(A:{}, E:=d.), a=E18.3 AND E20.4,m=3
SPB=Label1 (a=0)
SPB=Label1 (a=1)
V=(A:{}, E:=d.), a=1,m=4
UE20.6
V=(A:{}, E:=d.), a=E20.6,m=5
S A14.1
V=(A:A14.1=E20.6, E:=d.), a=E20.6,m=6
SPA=Label2
V=(A: A14.1=E20.6, E:=d.),a=E20.6, m=9
OE21.0
V=(A: A14.1=E20.6, E:=d.),a=E21.0, m=10
= A16.0
V=(A: A14.1=E20.6, A16.0=E21.0, E:=d.),a=E21.0 m=11
V=(A:{}, E:=d.), a=1,m=7
UE20.7
V=(A:{}, E:=d.), a=E20.07m=8
= A15.0
V=(A: A15.0=E20.7, E:=d.),a=E20.7, m=9
OE21.0
V=(A: A15.0=E20.7, E:=d.),a=E21.0, m=10
= A16.0
V=(A: A15.0=E20.7, A16.0=E21.0, E:=d.),a=E21.0 m=11

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Conversion of PLC Programs

Alternative No. 5



Binary Example

PLC Text

```
0000          :U      E     38.1
0002          :U      E     18.3
0004          :U      E     20.4
0006          :SPB    LAB1
0008          :U      E     20.6
000A          :S      A     14.1
000B          :SPA    LAB2
000C   LAB1   :U      E     20.7
000D          :=      A     15.0
000E   LAB2   :O      E     21.0
000F          :=      A     16.0
0010          :BE
```

Transformation	No. of states	No. of Transitions	State Contents
Alternative no. 1 (Single Operations)	28 states	29	CR, PC, Variables, output
Alternative no. 2 (University of Cachan)	14 states	13	CR, PC, Variables, output
Alternative no. 3 (Optimization of Alt. 2)	9 states	8	CR, PC, Variables, output
Alternative no. 4 (Abstraction)	6 states	9	PC, Variables
Alternative no. 5 (Moore)	9 states	14	CR, PC, Variables output
Alternative no. 6 (Alt. 4 no contents)	6 states	9	No State contents

Comparison of the transformations

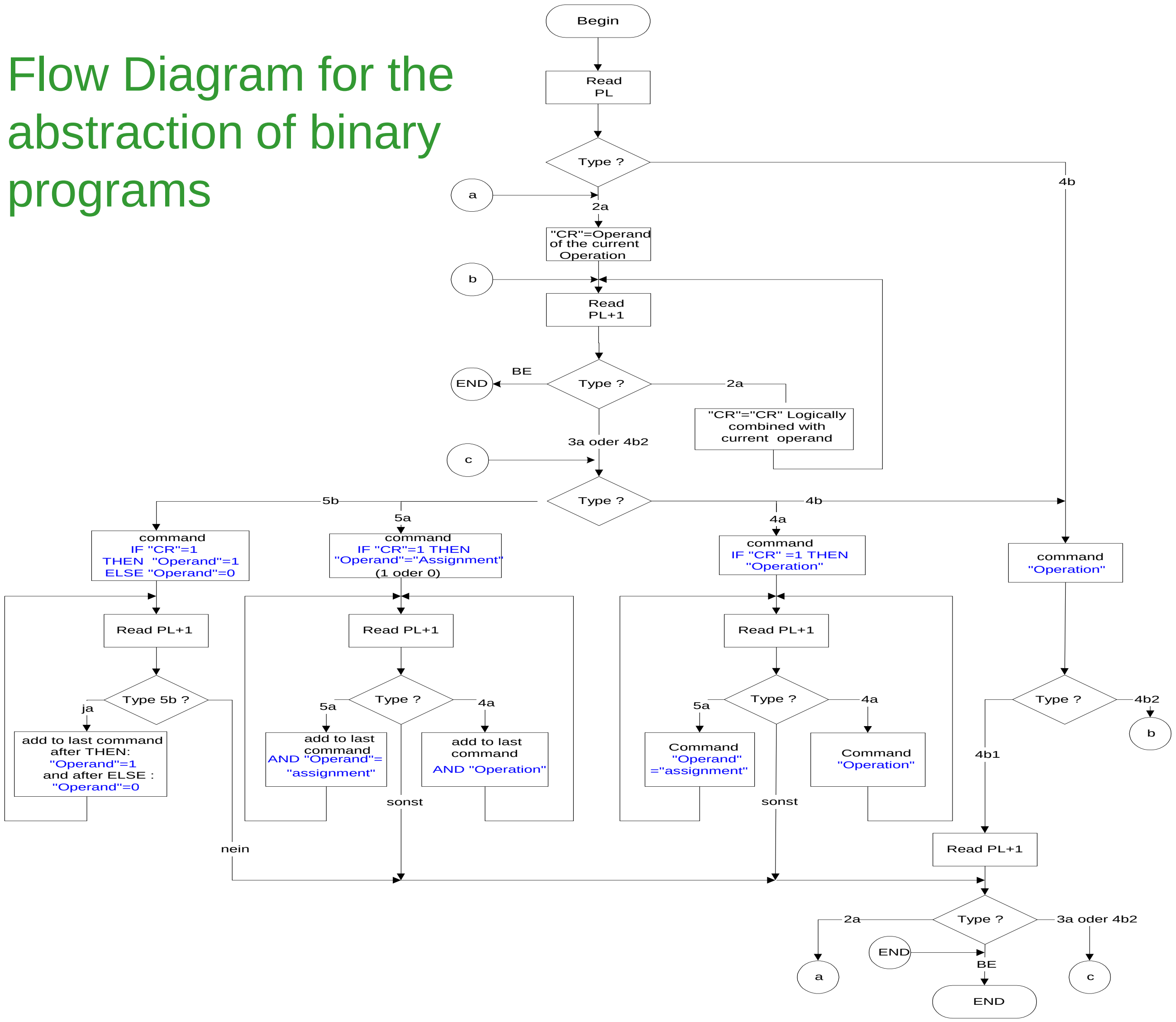
Classification of binary operations

Typ 1			Typ 2		Typ 3		Typ 4			Typ 5	
1a	1b	1c	2a	2b	3a	3b	4a	4b		5a	5b
								4b1	4b2		
S R SPB BAB BEB	SI SV SE SA ZV ZR	SA	U UN O ON O U(O( )	SPB BAB BEB	S R = SI SV SE SS SA ZV ZR SPA BA SPB BAB BE BEB BEA	O U(O(	SPB BAB BEB	SPA BA BE BEA	A,A X E,EX	S R	=

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Fourth and Sixth alternative

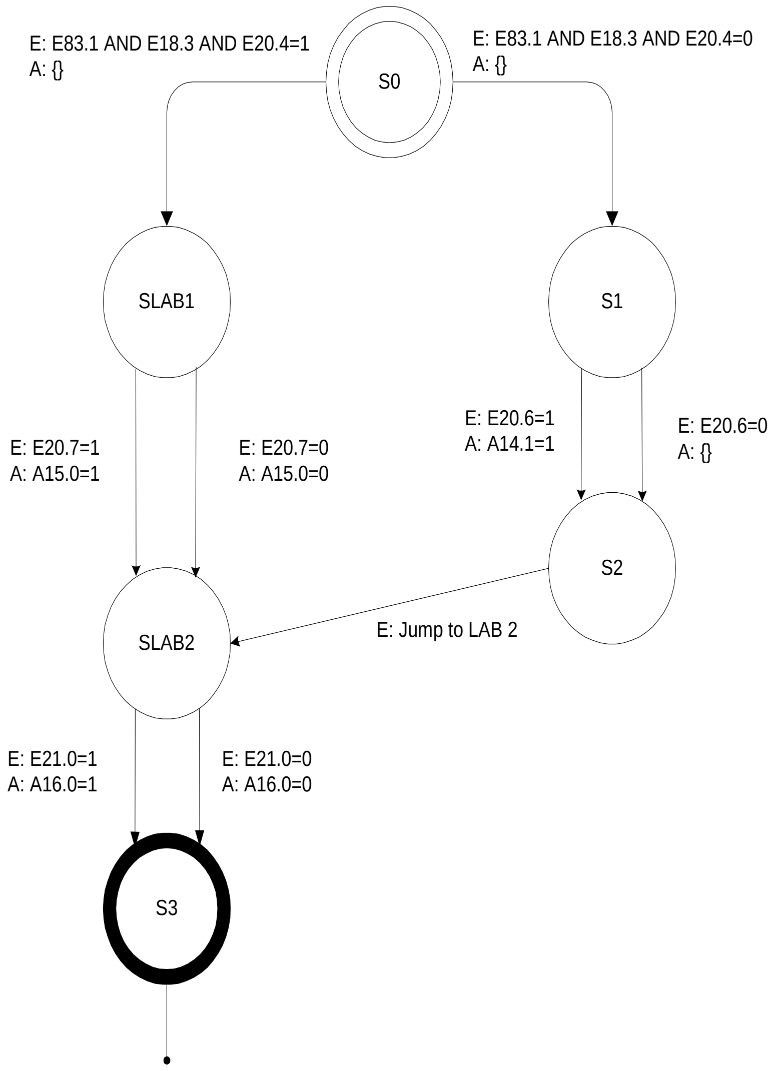
Flow Diagram for the abstraction of binary programs


Begin
Read PL
Type ?
4b
a
2a
"CR"=Operand of the current Operation
b
Read PL+1
BE
END
Type ?
2a
"CR"="CR" Logically combined with current operand
3a oder 4b2
c
Type ?
5b
5a
4b
4a
command IF "CR"=1 THEN "Operand"=1 ELSE "Operand"=0
command IF "CR"=1 THEN "Operand"="Assignment" (1 oder 0)
command IF "CR" =1 THEN "Operation"
command "Operation"
Read PL+1
Read PL+1
Read PL+1
Type 5b ?
ja
add to last command after THEN: "Operand"=1 and after ELSE : "Operand"=0
nein
Type ?
5a
4a
add to last command AND "Operand"= "assignment"
add to last command AND "Operation"
sonst
Type ?
5a
4a
Command "Operand" ="assignment"
Command "Operation"
sonst
Type ?
4b2
b
4b1
Read PL+1
Type ?
2a
3a oder 4b2
END
BE
a
c
END

Alternative Nr. 6

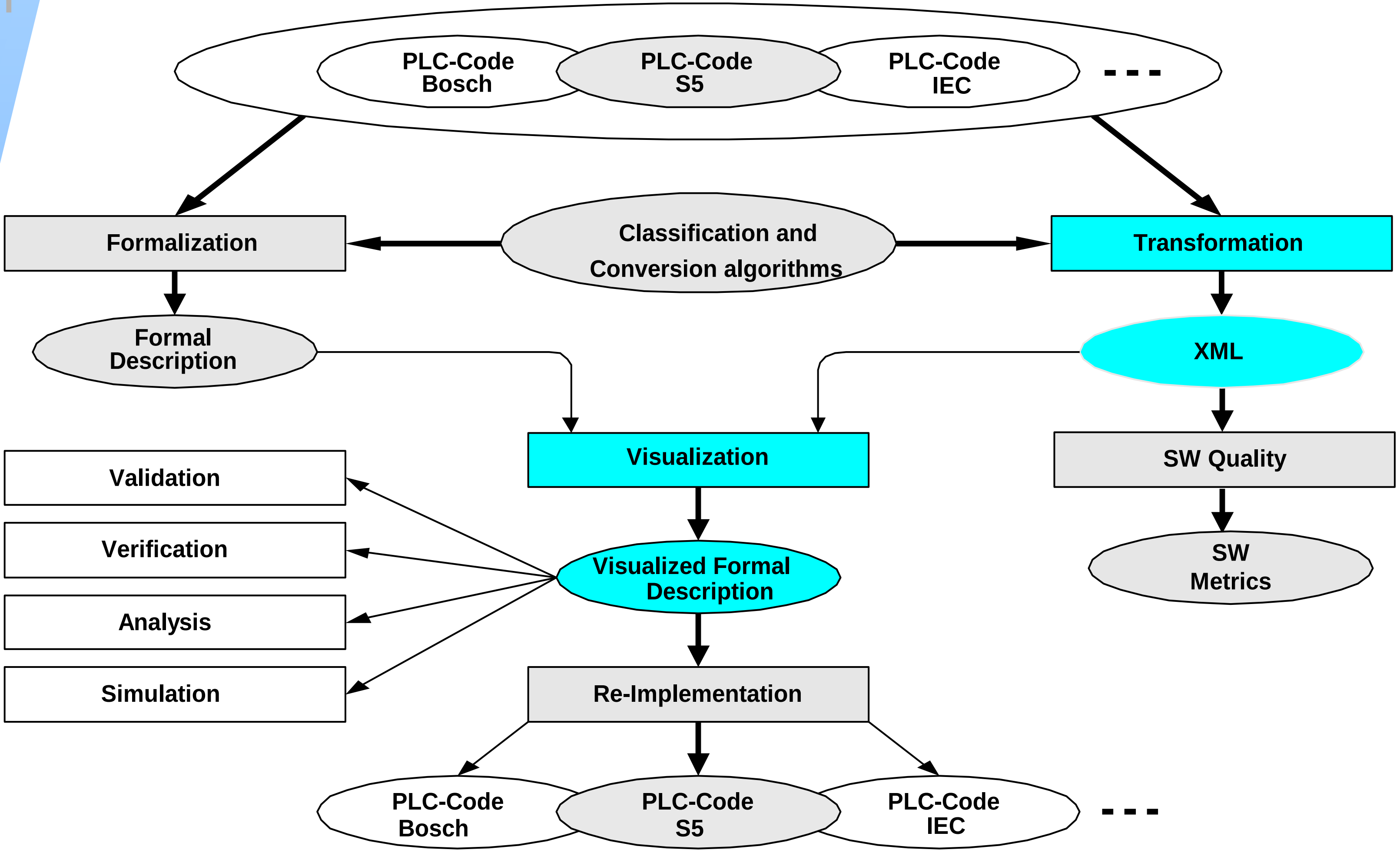
```
0000         :U      E      38.1
0002         :U      E      18.3
0004         :U      E      20.4
0006         :SPB    LAB1
0008         :U      E      20.6
000A         :S      A      14.1
000B         :SPA    LAB2
000C   LAB1  :U      E      20.7
000D         :=      A      15.0
000E   LAB2  :O      E      21.0
000F         :=      A      16.0
0010         :BE
```

```
        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
```



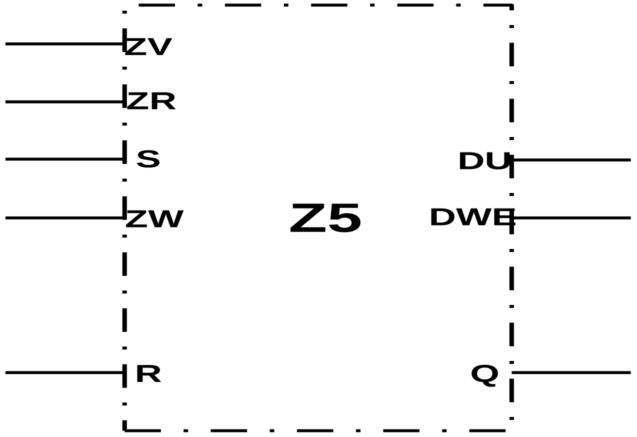
Overview

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



Formalization of Counters and Timers

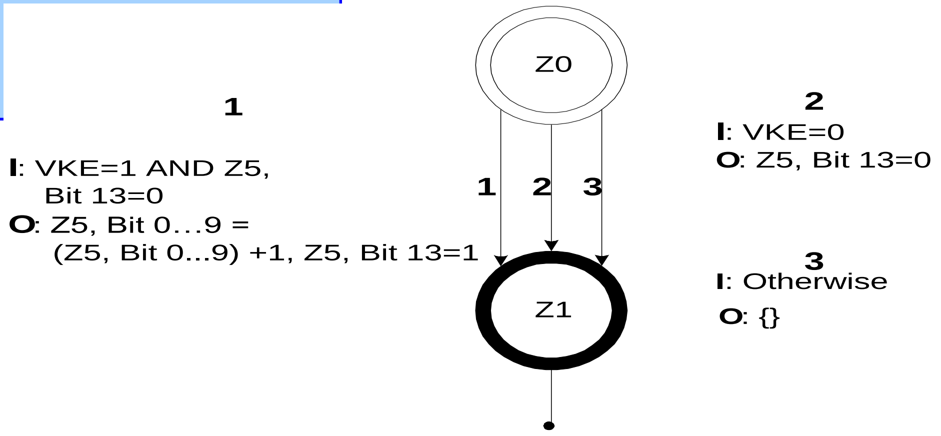
- Need for counters and counters
- Counting range 000 up to 999
- 16 bit word for a counter word consists of:
  - -State bits → to process the counter
  - -Counting Value → real value of the counter



Function block of a counter (Z5)

- 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
```


Z0
Z1
1
2
3
1
I: VKE=1 AND Z5, Bit 13=0
O: Z5, Bit 0…9 = (Z5, Bit 0…9) +1, Z5, Bit 13=1
2
I: VKE=0
O: Z5, Bit 13=0
3
I: Otherwise
O: {}

Need to ask for the state fo 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

Types of Non-Binary Operations

Type	Operations
1	Load operation
2	Transfer operation
3	Arithmetic operation
4	Compare operation
5	Digital logical operation
6	1s complement operation
7	2s complement operation
8	Shift and rotate operation
9	Jump operation
10	Other operation

- Transformation of IF-THEN- ELSE Algorithms into Mealy FSM
- Optimization of the Abstraction according to optimization algorithm

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Digital or Non-Binary Programs

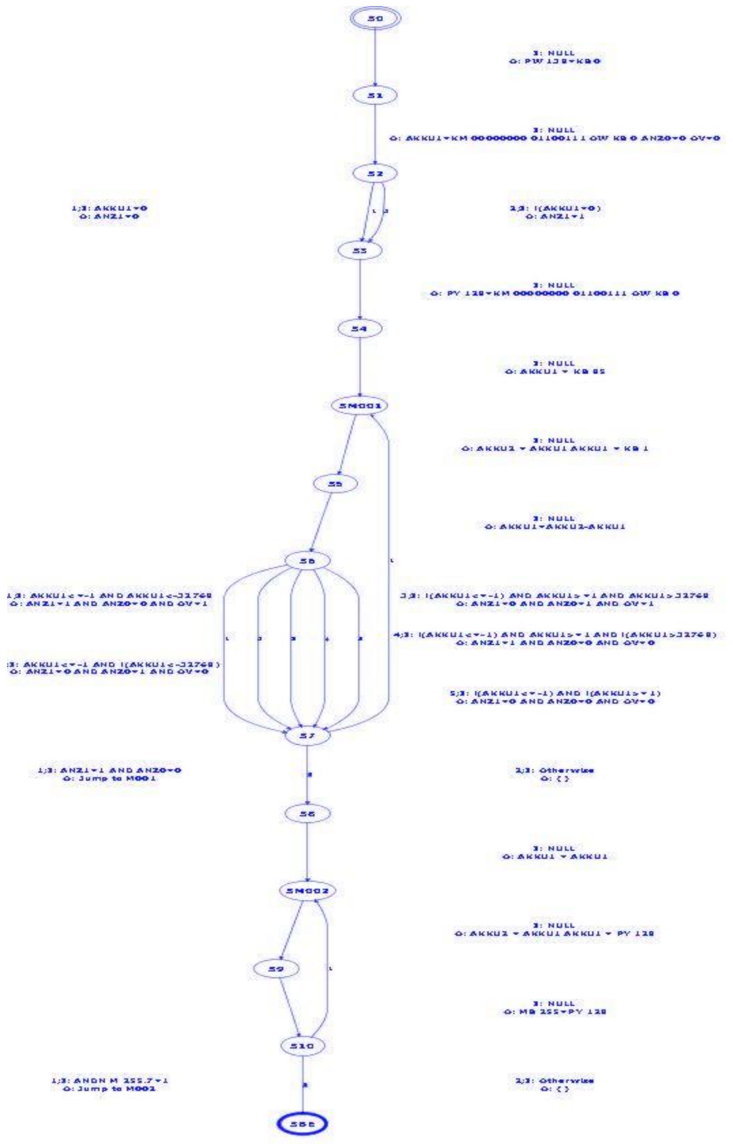
Example

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
```

IF_THEN_ELSE

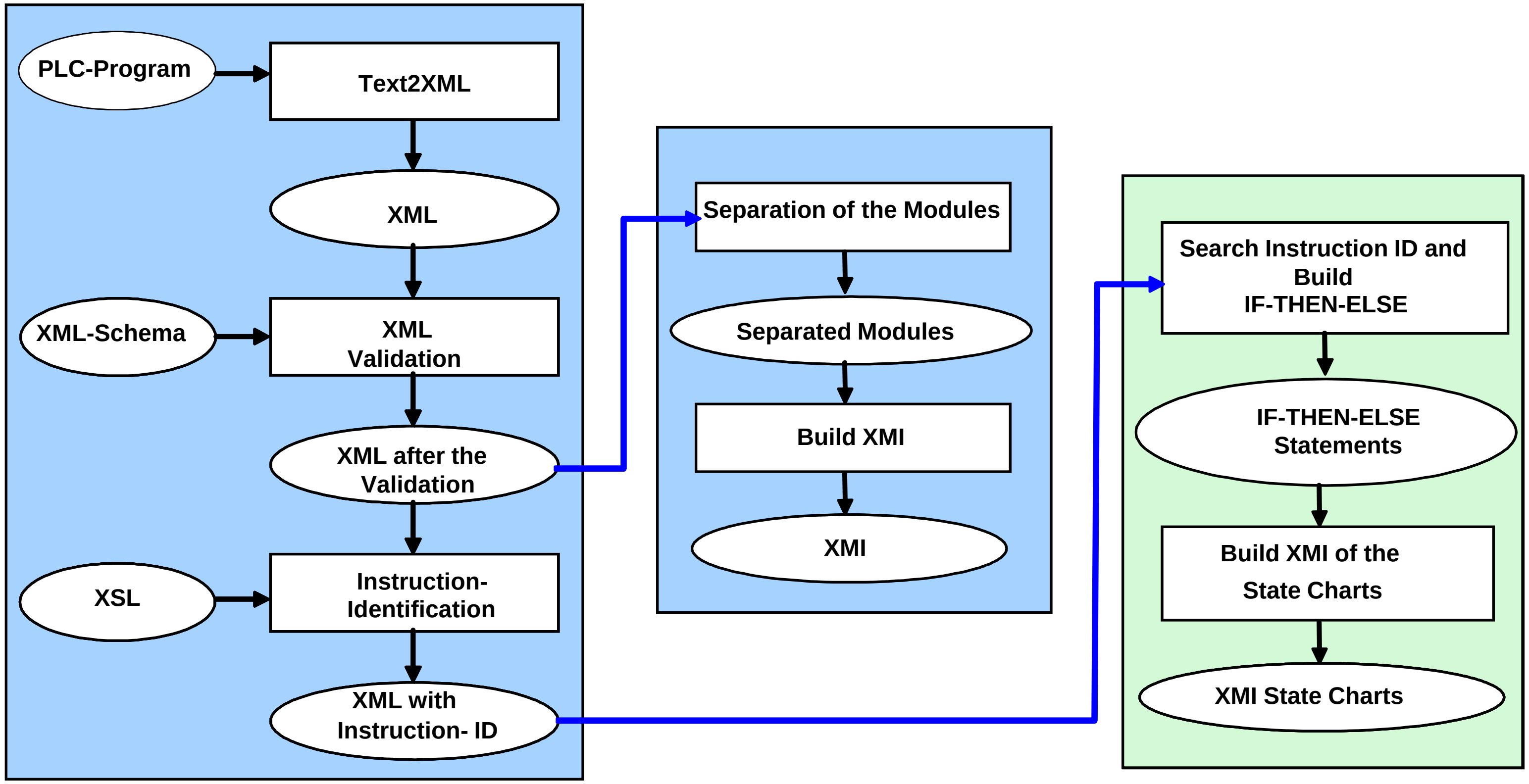
```
PW138 = KB0
AKKU 1 = KM0000000010011OW KB0
ANZ0 = 0 AND OV = 0
IF AKKU 1 = 0 THEN ANZ1 = 0 ELSE ANZ1 = 1
PY128 = AKKU 1
AKKU 1 = KB 85
M0     AKKU 2 = AKKU 1
       AKKU 1 = KB1
       AKKU1 = AKKU 2-AKKU 1
       IF AKKU1 <= -1
       THEN  IF AKKU1 < -32768
             THEN ANZ1 = 1 AND ANZ0 = 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
```


S0
I: NULL
O: PW 138=KB0
S1
I: NULL
O: AKKU1=KM 00000000 01100111 OW KB 0 ANZ0=0 OV=0
S2
1;I: AKKU1=0
O: ANZ1=0
2;I: !(AKKU1=0)
O: ANZ1=1
S3
I: NULL
O: PY 128=KM 00000000 01100111 OW KB 0
S4
I: NULL
O: AKKU1 = KB 85
SM001
I: NULL
O: AKKU2 = AKKU1 AKKU1 = KB 1
S5
I: NULL
O: AKKU1=AKKU2-AKKU1
S6
1;I: AKKU1<=-1 AND AKKU1<-32768
O: ANZ1=1 AND ANZ0=0 AND OV=1
2;I: AKKU1<=-1 AND !(AKKU1<-32768)
O: ANZ1=0 AND ANZ0=1 AND OV=0
3;I: !(AKKU1<=-1) AND AKKU1>=1 AND AKKU1>32768
O: ANZ1=0 AND ANZ0=1 AND OV=1
4;I: !(AKKU1<=-1) AND AKKU1>=1 AND !(AKKU1>32768)
O: ANZ1=1 AND ANZ0=0 AND OV=0
5;I: !(AKKU1<=-1) AND !(AKKU1>=1)
O: ANZ1=0 AND ANZ0=0 AND OV=0
S7
1;I: ANZ1=1 AND ANZ0=0
O: Jump to M001
2;I: Otherwise
O: { }
S8
I: NULL
O: AKKU1 = AKKU1
SM002
I: NULL
O: AKKU2 = AKKU1 AKKU1 = PY 128
S9
I: NULL
O: MB 255=PY 128
S10
1;I: ANDN M 255.7=1
O: Jump to M002
2;I: Otherwise
O: { }
SBE

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Visualization Concept

Steps toward the conversion



→State Charts as a visualization alternative

Internet Technologies

CGI JavaScript C# XHTML SMTP WML COM IP Java OS FTP JNI DOM RTF XSL XML XPath DNS HTTP Email B2B eCl@ss TCP eCommerce SOAP XLink Corba WWW RMI Perl HTML News NNTP Java SAX DTD JVM

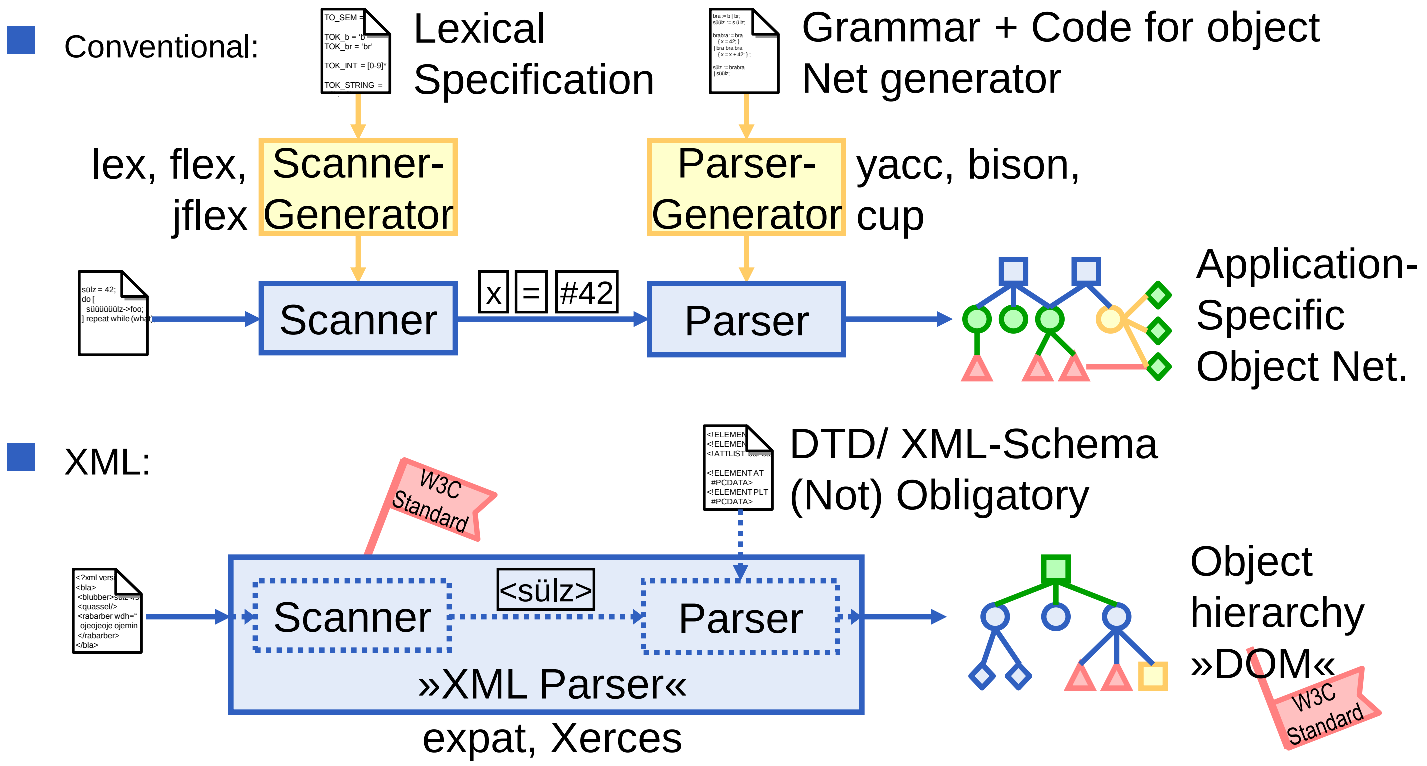
services
programming languages
protocols
mark-up languages
models
operating systems
interfaces
schemes

Fields of Application

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

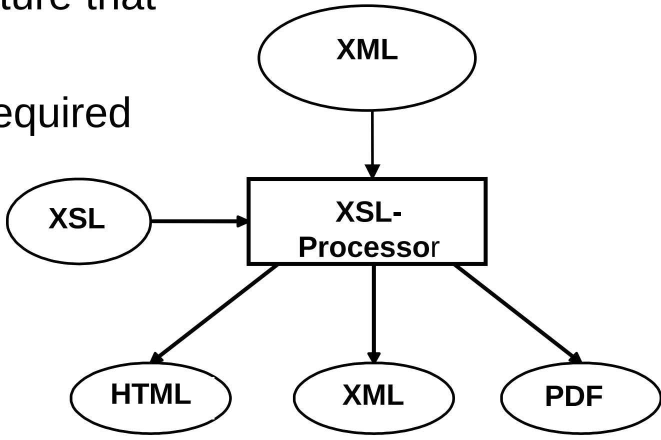
XML as a tool for visualization

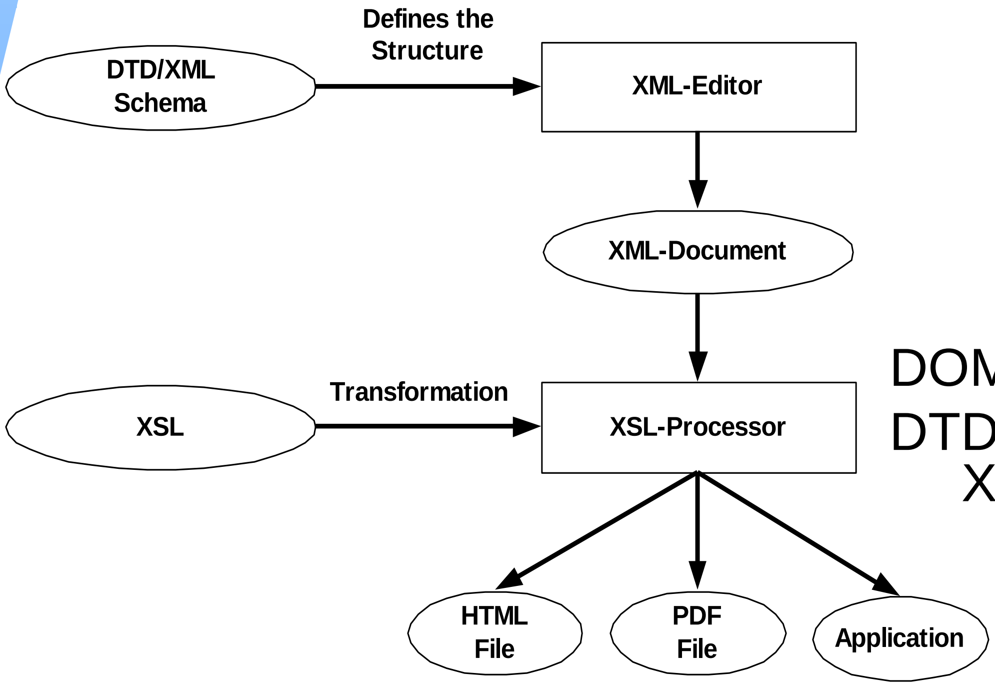
- XML (eXtensible Markup Language)
- XML and HTML
- XML to exchange information across platforms and applications.
- How to apply XML?



XML as a tool for visualization

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



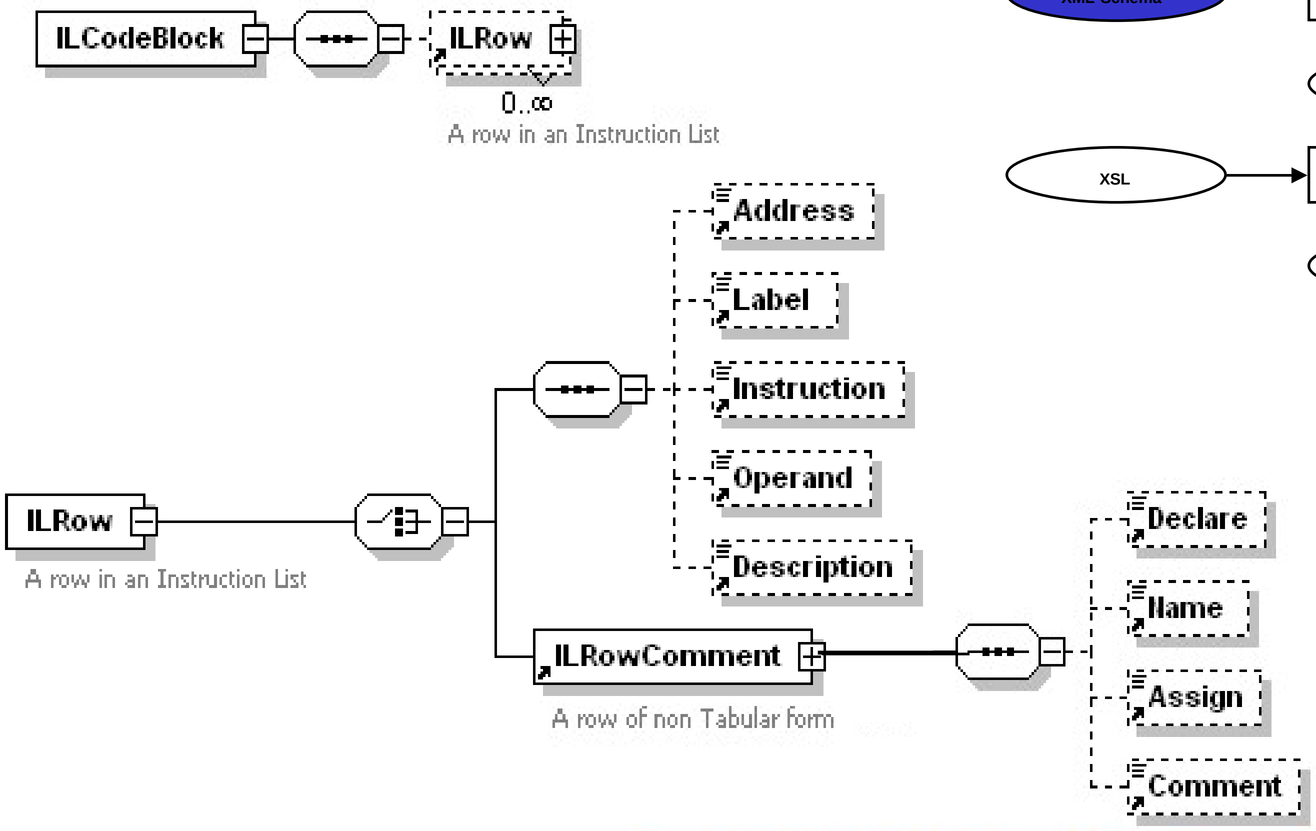


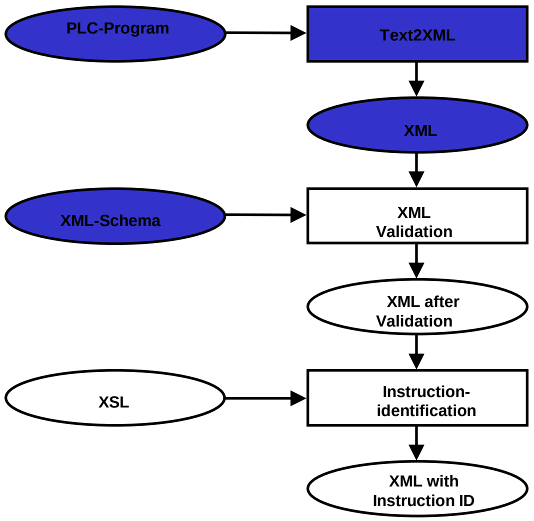
DOM (Document Object Model)

DTD (Document Type Definition) or XML-Schema

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

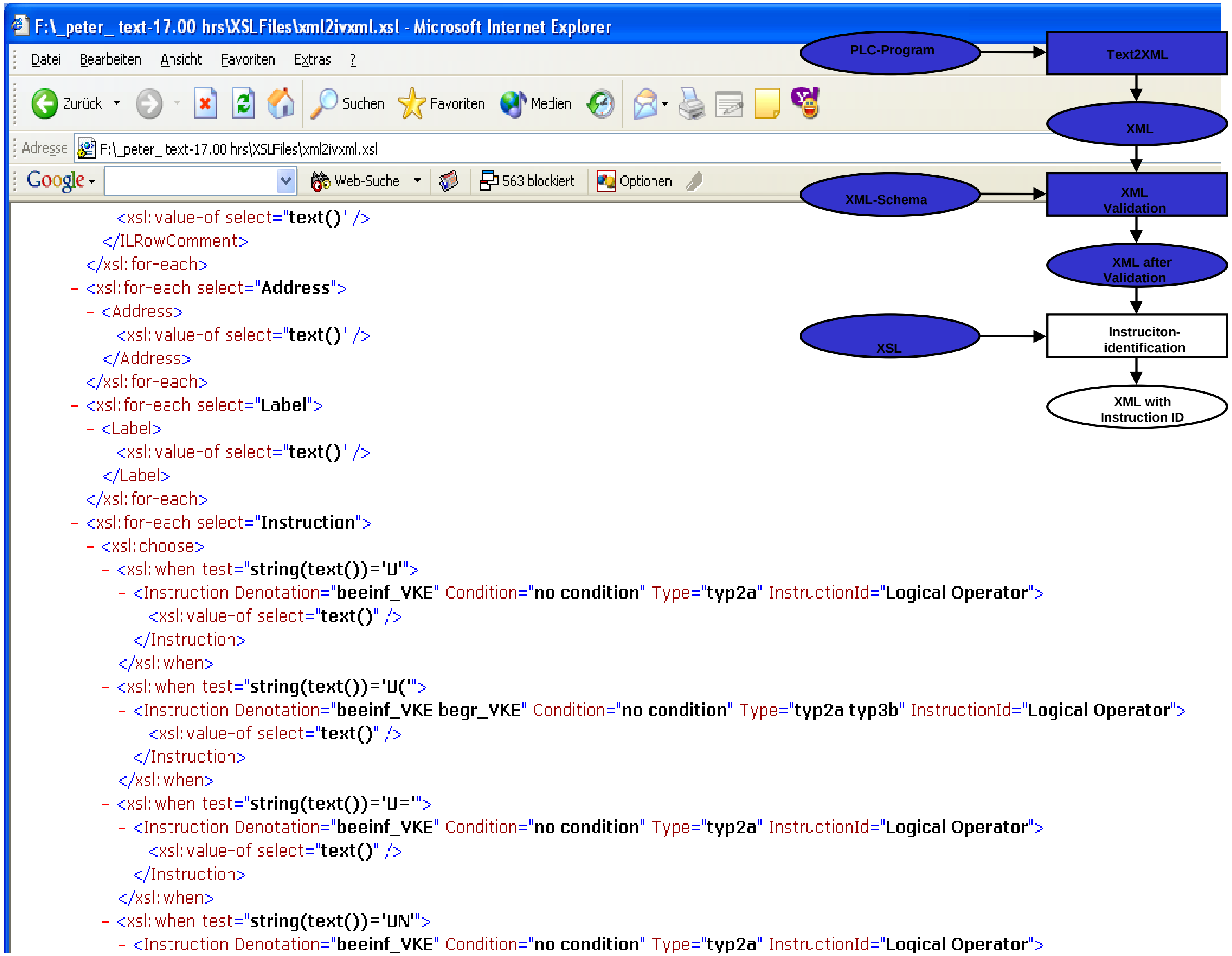
XML-Schema for IL (graphical)



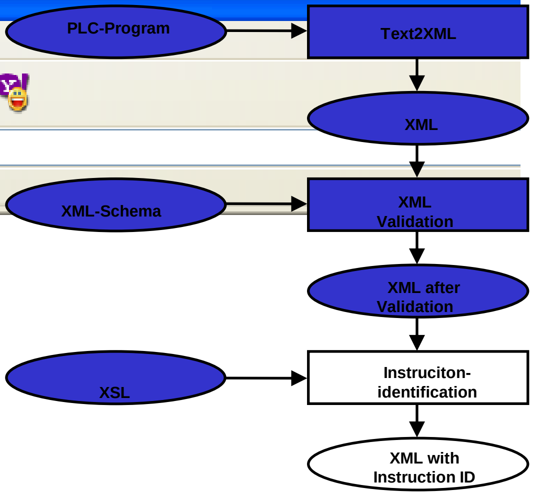


XSL for Instruction Identification of STP5

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


F:_peter_ text-17.00 hrs\XSLFiles\xml2ivxml.xsl - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück
Suchen
Favoriten
Medien
Adresse F:_peter_ text-17.00 hrs\XSLFiles\xml2ivxml.xsl
Google
Web-Suche
563 blockiert
Optionen

```
        <xsl:value-of select="text()" />
      </ILRowComment>
    </xsl:for-each>
  - <xsl:for-each select="Address">
    - <Address>
        <xsl:value-of select="text()" />
      </Address>
    </xsl:for-each>
  - <xsl:for-each select="Label">
    - <Label>
        <xsl:value-of select="text()" />
      </Label>
    </xsl:for-each>
  - <xsl:for-each select="Instruction">
    - <xsl:choose>
      - <xsl:when test="string(text())='U'">
        - <Instruction Denotation="beeinf_VKE" Condition="no condition" Type="typ2a" InstructionId="Logical Operator">
            <xsl:value-of select="text()" />
          </Instruction>
        </xsl:when>
      - <xsl:when test="string(text())='U('">
        - <Instruction Denotation="beeinf_VKE begr_VKE" Condition="no condition" Type="typ2a typ3b" InstructionId="Logical Operator">
            <xsl:value-of select="text()" />
          </Instruction>
        </xsl:when>
      - <xsl:when test="string(text())='U='">
        - <Instruction Denotation="beeinf_VKE" Condition="no condition" Type="typ2a" InstructionId="Logical Operator">
            <xsl:value-of select="text()" />
          </Instruction>
        </xsl:when>
      - <xsl:when test="string(text())='UN'">
        - <Instruction Denotation="beeinf_VKE" Condition="no condition" Type="typ2a" InstructionId="Logical Operator">
```



TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

PLC Example

OB 1

```
NETZWERK   1
0000      :SPA PB 1  Jump Absolute to P
0002      :BE
```

PB 1

```
NETZWERK   1
0000:U       E38.1        //AND Operation
0002:U       E38.2
0004:O                    //OR Operation
0006:U       E38.1
0008:U       E38.3
000A         :O
000C         :U           E38.2
000E         :U           E38.3
0010:=       M100.0       at least two Fans running
0012:UN      E38.1        // ANDN Operation
0014:UN      E38.2
0016:UN      E38.3
0018:=       M100.1       no running Fan
001A         :U(
001C         :O           M100.0  Continuous Light
001E         :O
0020:U       M100.1
0022:U       M99.1        Flashing with 2 Hz
0024:O
0026:UN      M100.0
0028:UN      M100.1
002A         :U           M99.2 Flashing with 0,5 Hz
002C         :)
002E         :U           A42.4 „Active“
0030:=       A51.7        LCD lamp
0032:BE
```

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

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

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

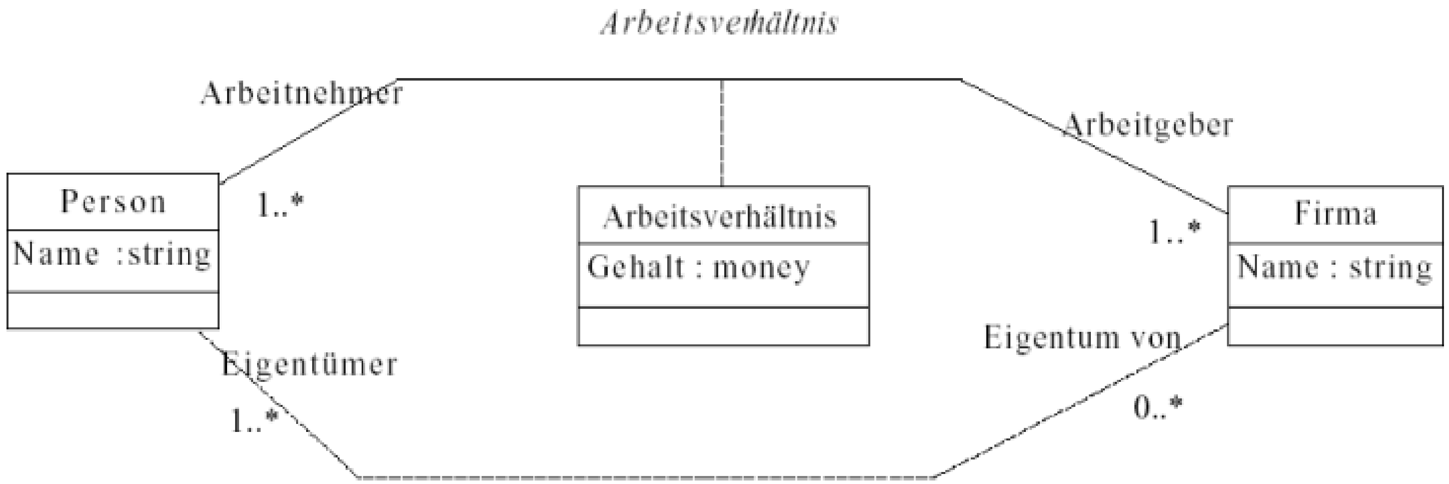
It's useful for transferring the model from one step to the next as your design and coding progress

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

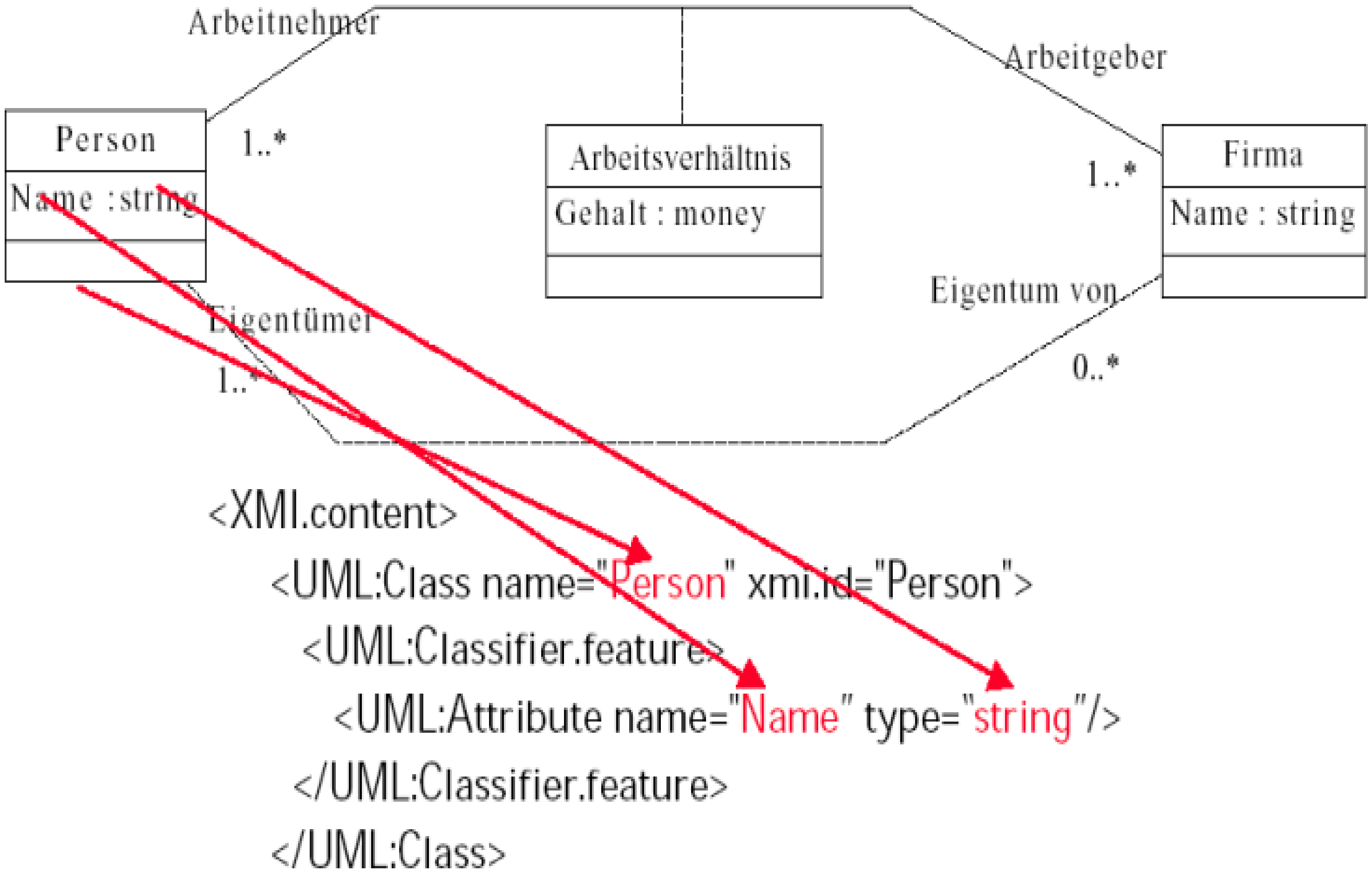
SW tools available made it possible to integrate XMI to UML (by import project form XMI or export project to XMI)

XMI_UML Example



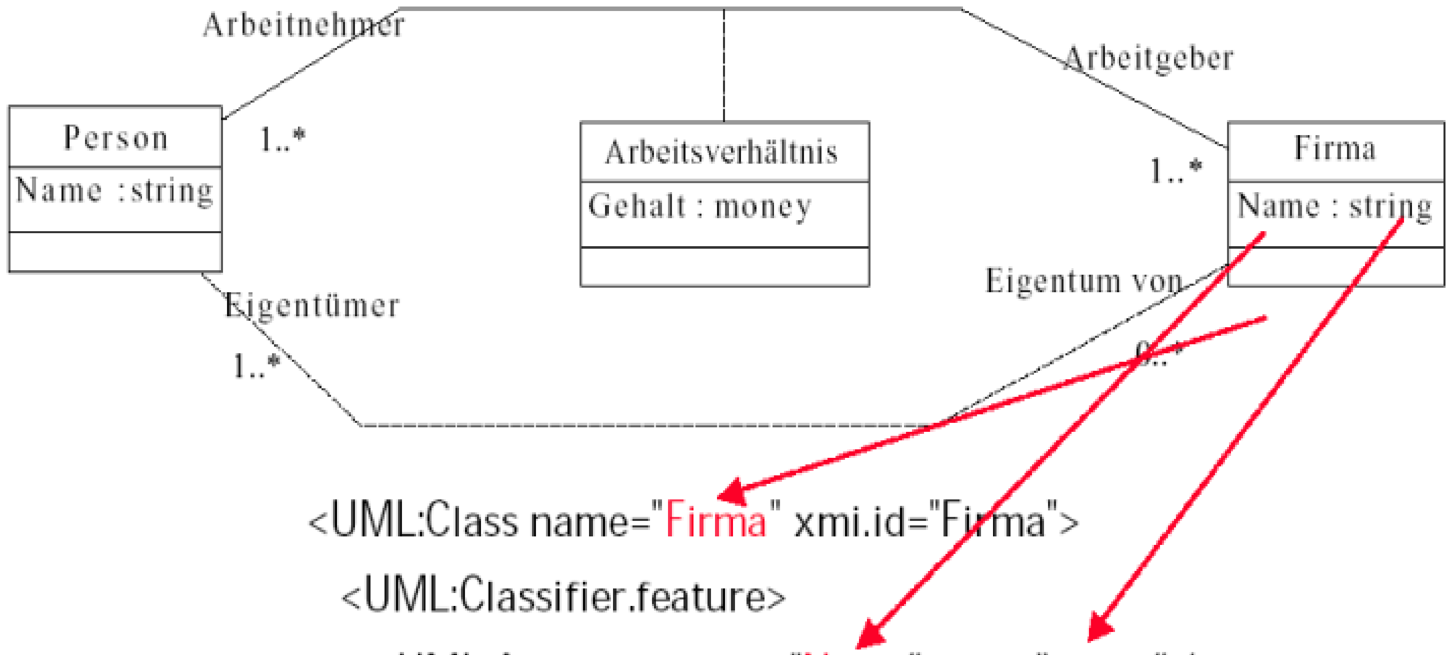
```
<XMI timestamp="2000-10-09T17:00:00" verified="true" xmi.version="1.1">
  <XMI.header>
    <XMI.model xmi.name="SimpleClassModel"/>
    <XMI.metamodel xmi.name="UML" xmi.version="1.3"/>
</XMI.header>
```

XMI_UML Example



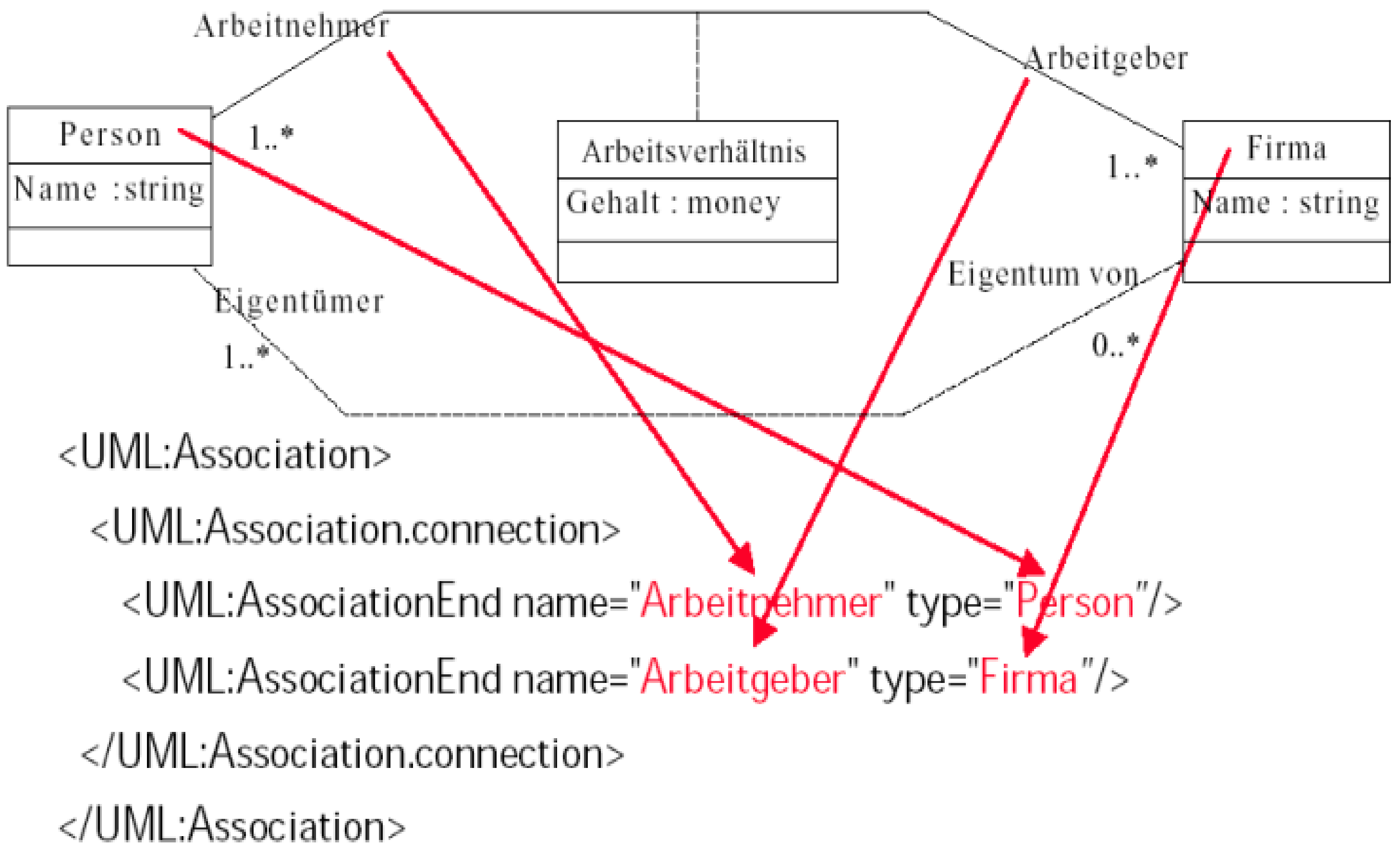
```
<XMI.content>
  <UML:Class name="Person" xmi.id="Person">
    <UML:Classifier.feature>
      <UML:Attribute name="Name" type="string"/>
    </UML:Classifier.feature>
  </UML:Class>
```

XMI_UML Example



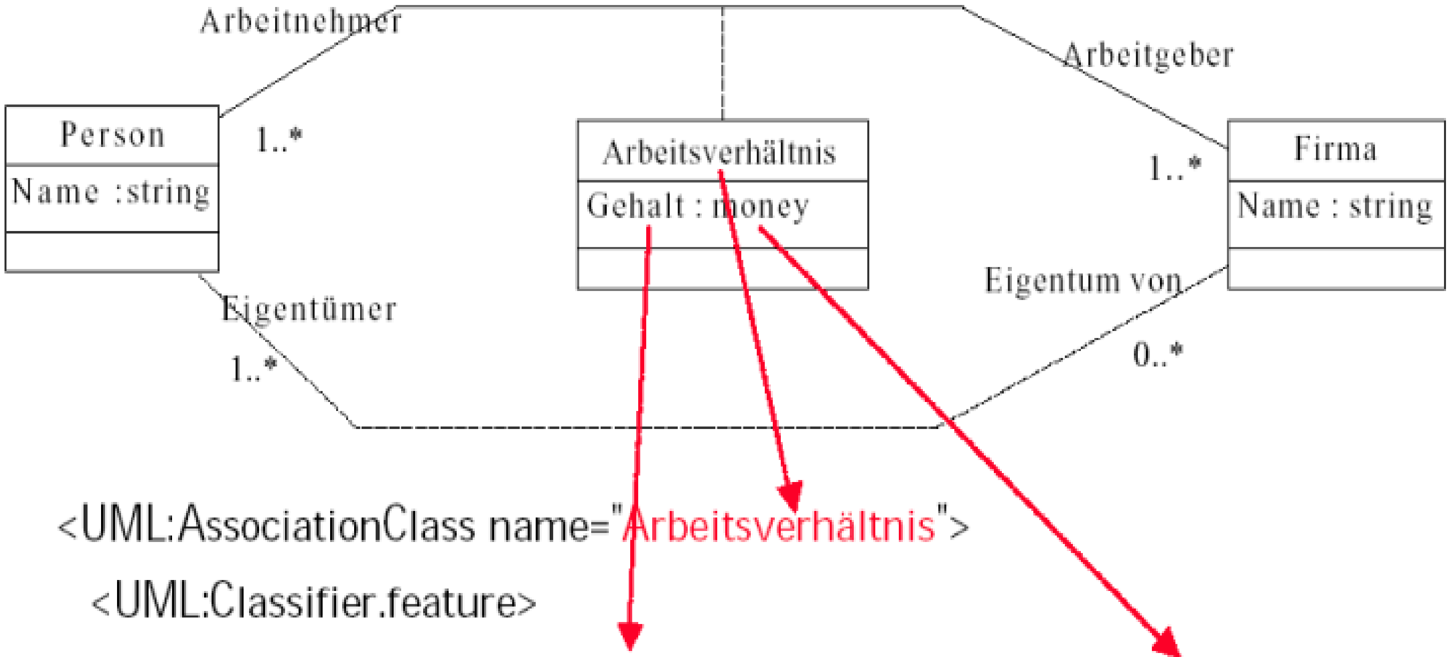
```
<UML:Class name="Firma" xmi.id="Firma">
  <UML:Classifier.feature>
    <UML:Attribute name="Name" type="string" />
  </UML:Classifier.feature>
 </UML:Class>
```

XMI_UML Example



```
<UML:Association>
  <UML:Association.connection>
    <UML:AssociationEnd name="Arbeitnehmer" type="Person"/>
    <UML:AssociationEnd name="Arbeitgeber" type="Firma"/>
  </UML:Association.connection>
</UML:Association>
```

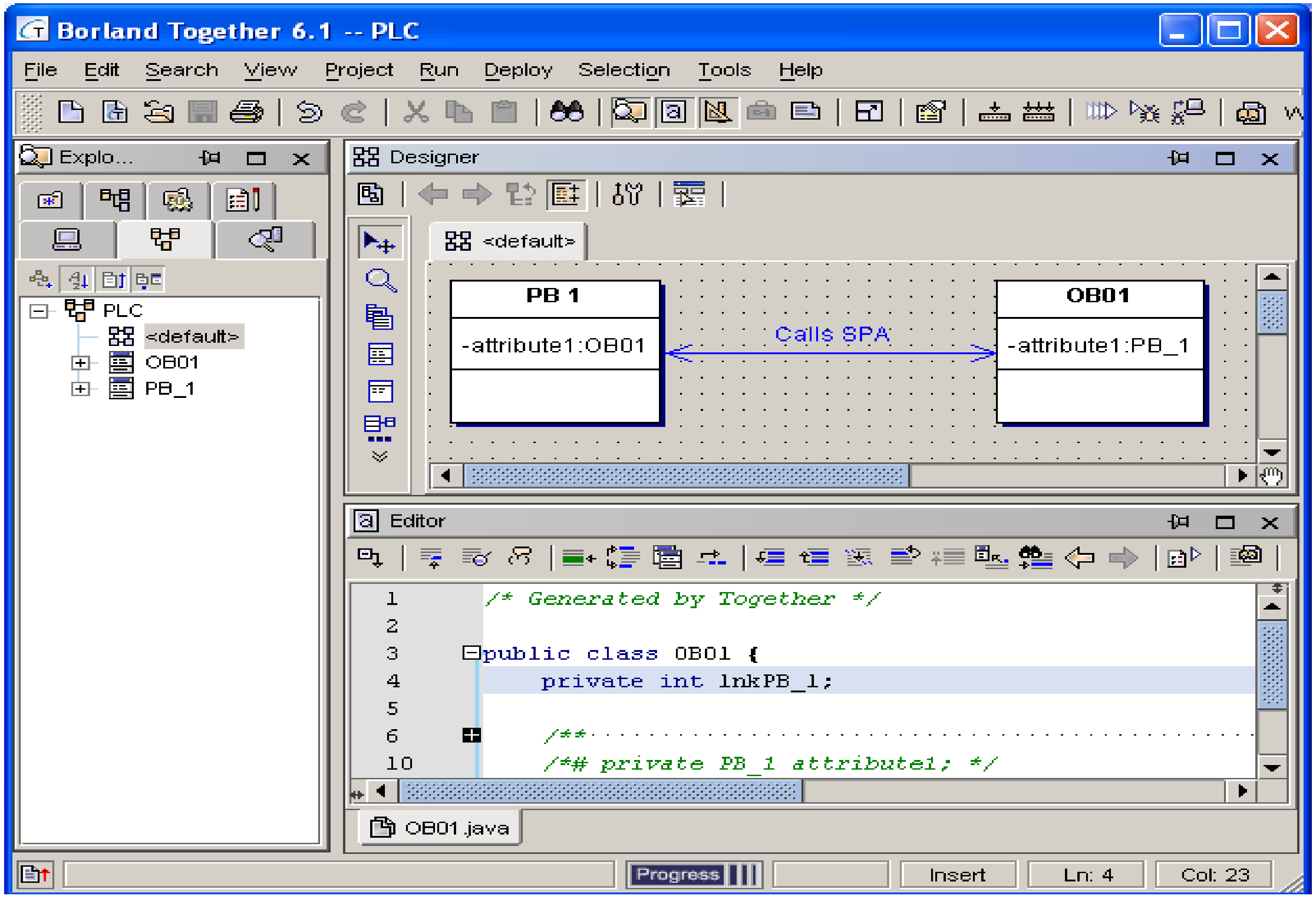
XMI_UML Example



```
<UML:AssociationClass name="Arbeitsverhältnis">
  <UML:Classifier.feature>
    <UML:Attribute name="Gehalt" multiplicity="1..1" type="money"/>
  </UML:Classifier.feature>
</UML:AssociationClass>
```

Modules Seperation using XMI

UML of OB 1 imported in Together



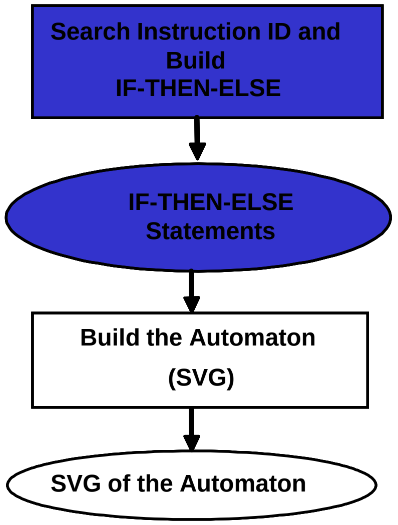
```
/* Generated by Together */

public class OB01 {
    private int lnkPB_1;

    /**
    /*# private PB_1 attribute1; */
```

Example: Abstraction into IF-THEN-ELSE

PB 1

```
NETZWERK   1
0000:U        E38.1        //AND Operation
0002:U        E38.2
0004:O                     //OR Operation
0006:U        E38.1
0008:U        E38.3
000A          :O
000C          :U           E38.2
000E          :U           E38.3
0010:=        M100.0       at least two Fans running
0012:UN       E38.1        // ANDN Operation
0014:UN       E38.2
0016:UN       E38.3
0018:=        M1
001A          :U(
001C          :O
001E          :O
0020:U        M1
0022:U        M9
0024:O
0026:UN       M1
0028:UN       M1
002A          :U
002C          :)
002E          :U
0030:=        A5
0032:BE
```

```
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
IF-THEN-ELSE Statements
Build the Automaton (SVG)
SVG of the Automaton

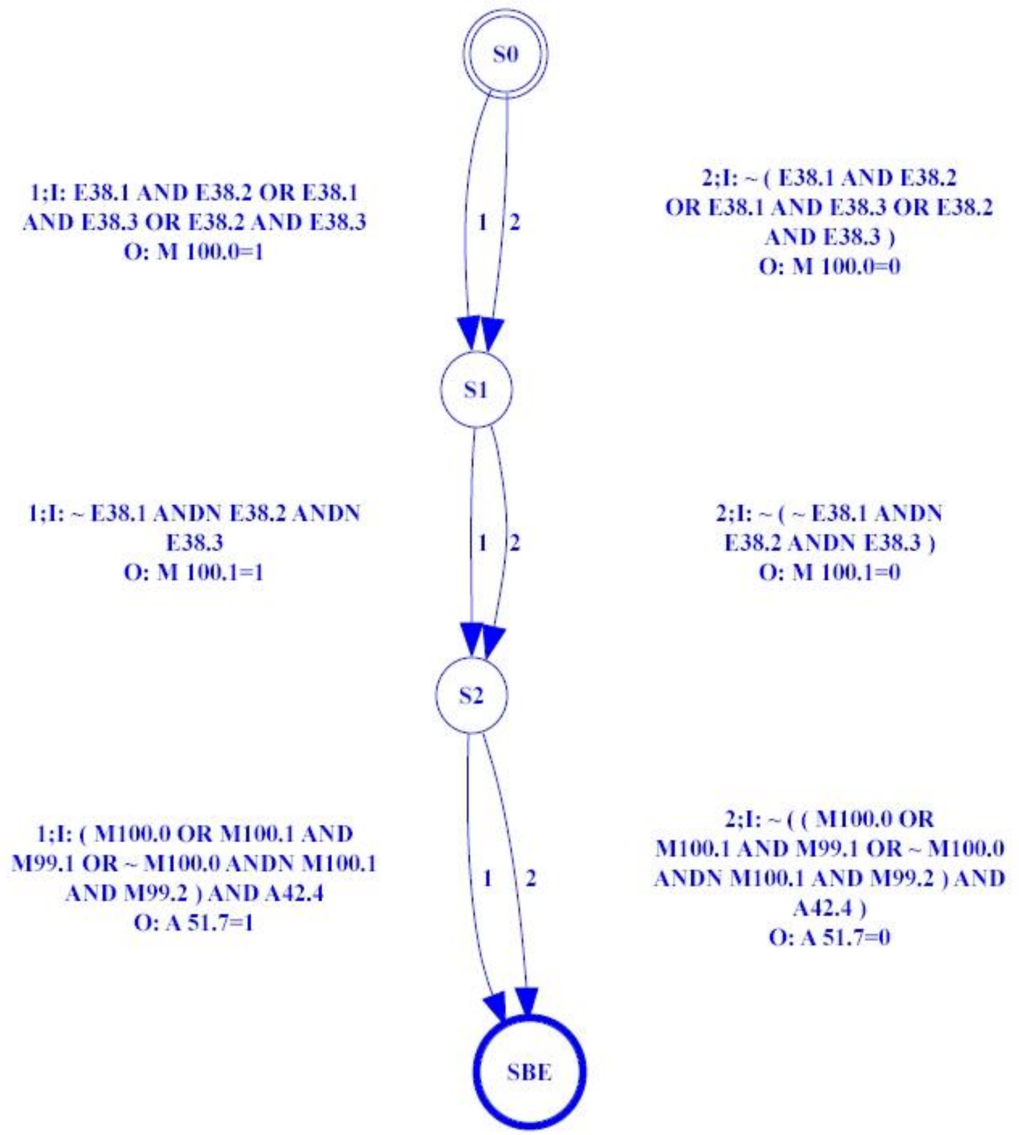
Example: Visualization through XML

```
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
```

```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>
```

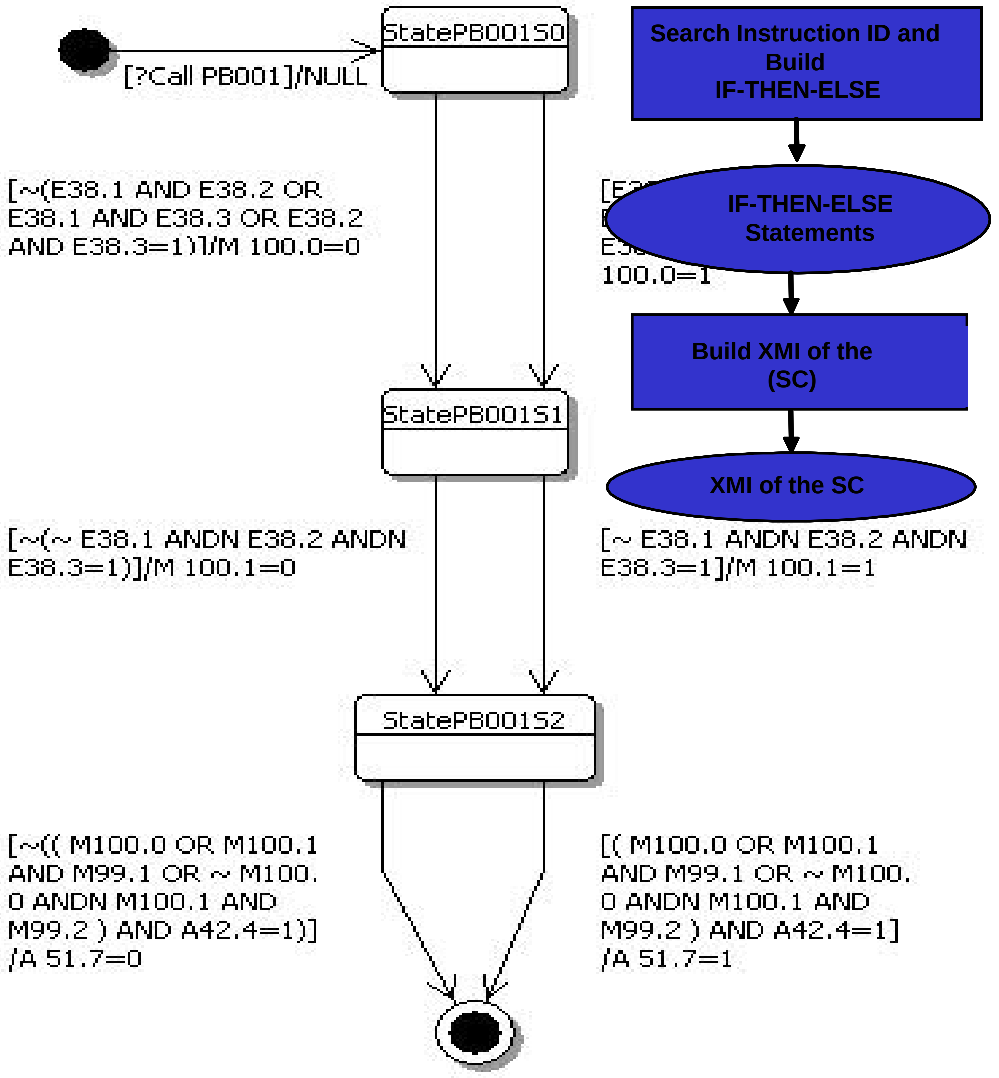
Example: Visualization through SVG

```
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
```


S0
1;I: E38.1 AND E38.2 OR E38.1 AND E38.3 OR E38.2 AND E38.3 O: M 100.0=1
1
2
2;I: ~ (E38.1 AND E38.2 OR E38.1 AND E38.3 OR E38.2 AND E38.3) O: M 100.0=0
S1
1;I: ~ E38.1 ANDN E38.2 ANDN E38.3 O: M 100.1=1
1
2
2;I: ~ (~ E38.1 ANDN E38.2 ANDN E38.3) O: M 100.1=0
S2
1;I: (M100.0 OR M100.1 AND M99.1 OR ~ M100.0 ANDN M100.1 AND M99.2) AND A42.4 O: A 51.7=1
1
2
2;I: ~ ((M100.0 OR M100.1 AND M99.1 OR ~ M100.0 ANDN M100.1 AND M99.2) AND A42.4) O: A 51.7=0
SBE

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	



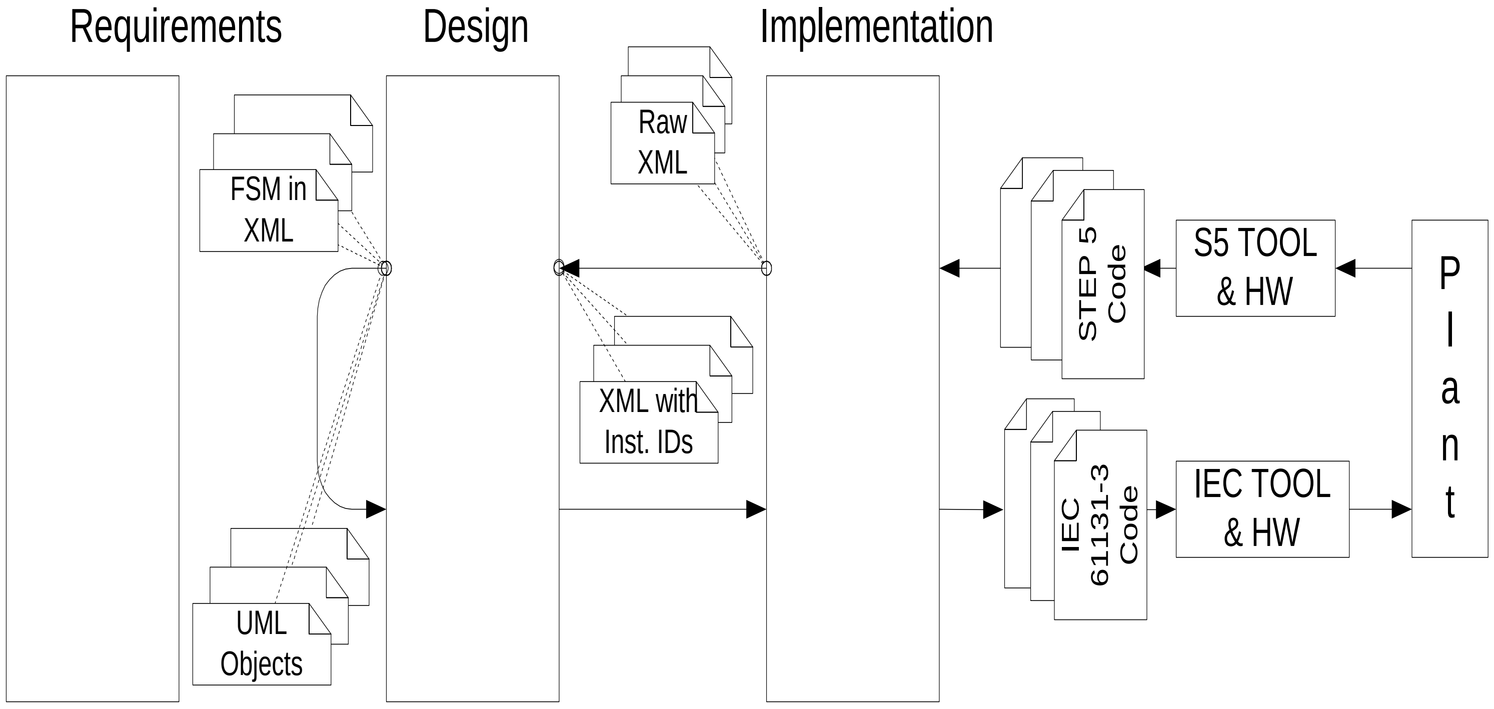
StatePB001S0
[?Call PB001]/NULL
Search Instruction ID and Build IF-THEN-ELSE
IF-THEN-ELSE Statements
[~(E38.1 AND E38.2 OR E38.1 AND E38.3 OR E38.2 AND E38.3=1)]/M 100.0=0
100.0=1
Build XMI of the (SC)
StatePB001S1
XMI of the SC
[~(~ E38.1 ANDN E38.2 ANDN E38.3=1)]/M 100.1=0
[~ E38.1 ANDN E38.2 ANDN E38.3=1]/M 100.1=1
StatePB001S2
[~((M100.0 OR M100.1 AND M99.1 OR ~ M100.0 ANDN M100.1 AND M99.2) AND A42.4=1)] /A 51.7=0
[(M100.0 OR M100.1 AND M99.1 OR ~ M100.0 ANDN M100.1 AND M99.2) AND A42.4=1] /A 51.7=1

Visualization's Concluding Remarks

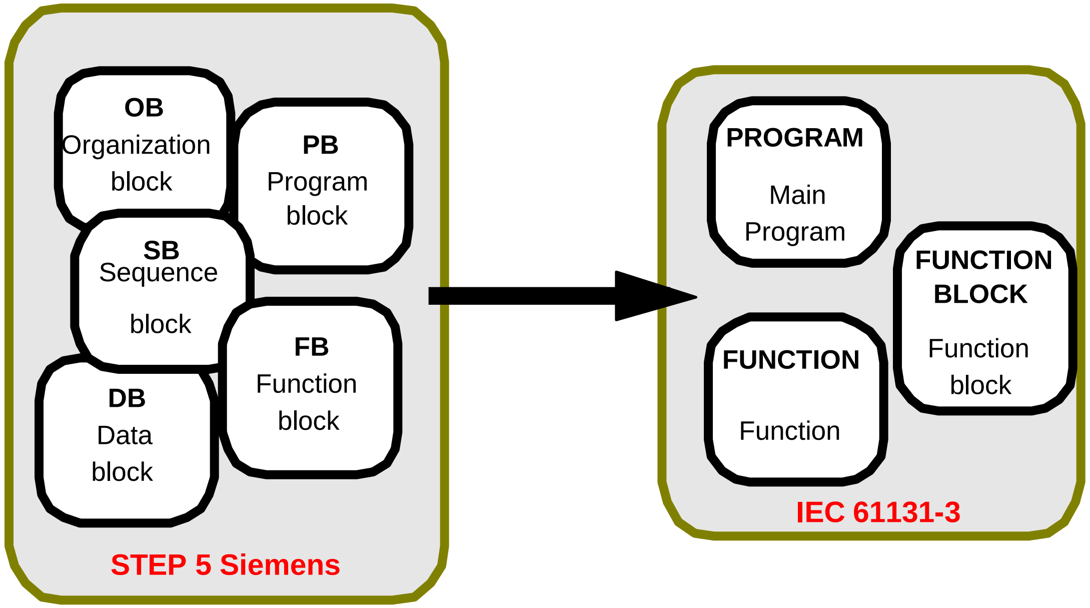
- XML allow the Visualization of the Formalization
- SVG to draw the FSMs
- Extraction of the PLC structure through XMI
- SC as an alternative for the Visualization
- CFSM in XML as a Basis for the Re-Implementation
- XML transformation for deriving the SW-Quality

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Overview

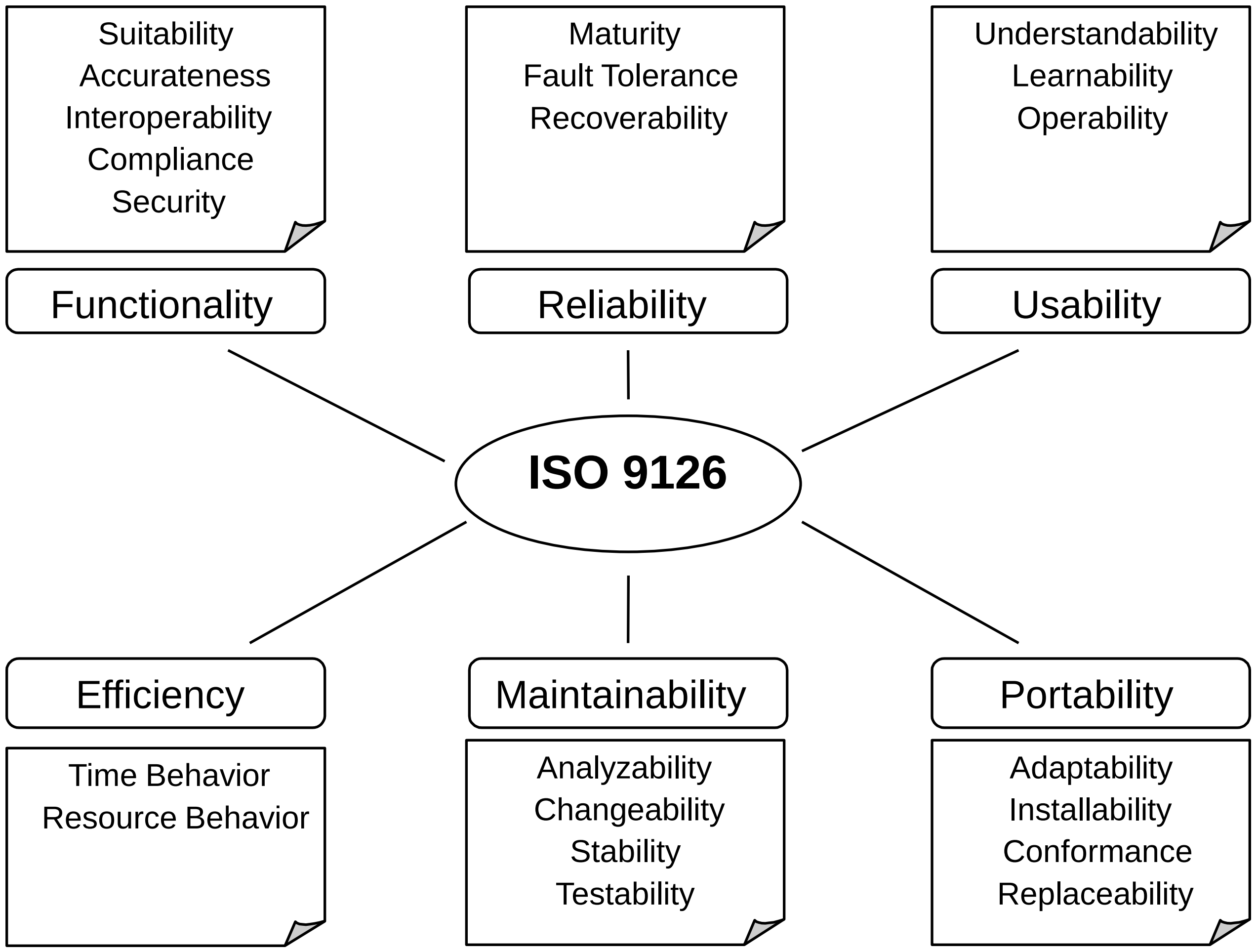


Concept of Re-Implementation



- 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



ISO 9126

Criterion	Definition
Functionality	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.
Reliability	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.
Usability	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.
Efficiency	Attributes that bear on the relationship between the level of the performance of the software and the amount of resources used, under stated conditions
Portability	Attributes that bear on the ability of software to be transformed from one environment to another.
Maintainability	Attributes that bear on the effort needed to make specified modifications

Evaluation of the Metrics

Name	practicability in Software	Usability to IL	Diagnosability Explanatory	Later use for the Diagnosis (online)
Size	++	++	- (Serves for the coarse appraisal)	--
Halstead	++	++	0 (Overview about operators and operands)	--
McCabe Cyclomatic Complexity	+	0 (Graph is necessary)	- (Information about conditioned jumps)	--
Information flow	0	- (flow definition?)	0 (Variables flow bet. Blocks)	0
Tree Impurity	0	+ (Graph is necessary)	+ (Blocks call)	--
Coupling	-	- (interconnection Definition)	0 (Variables flow bet. Blocks)	--

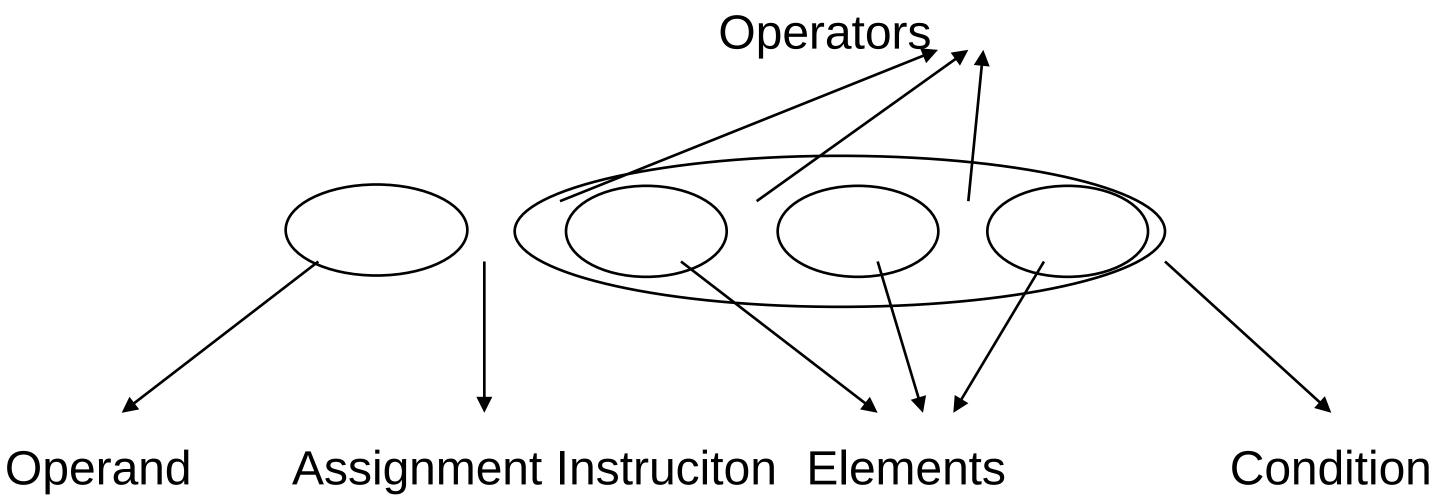
Condition Equation Construction

- 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

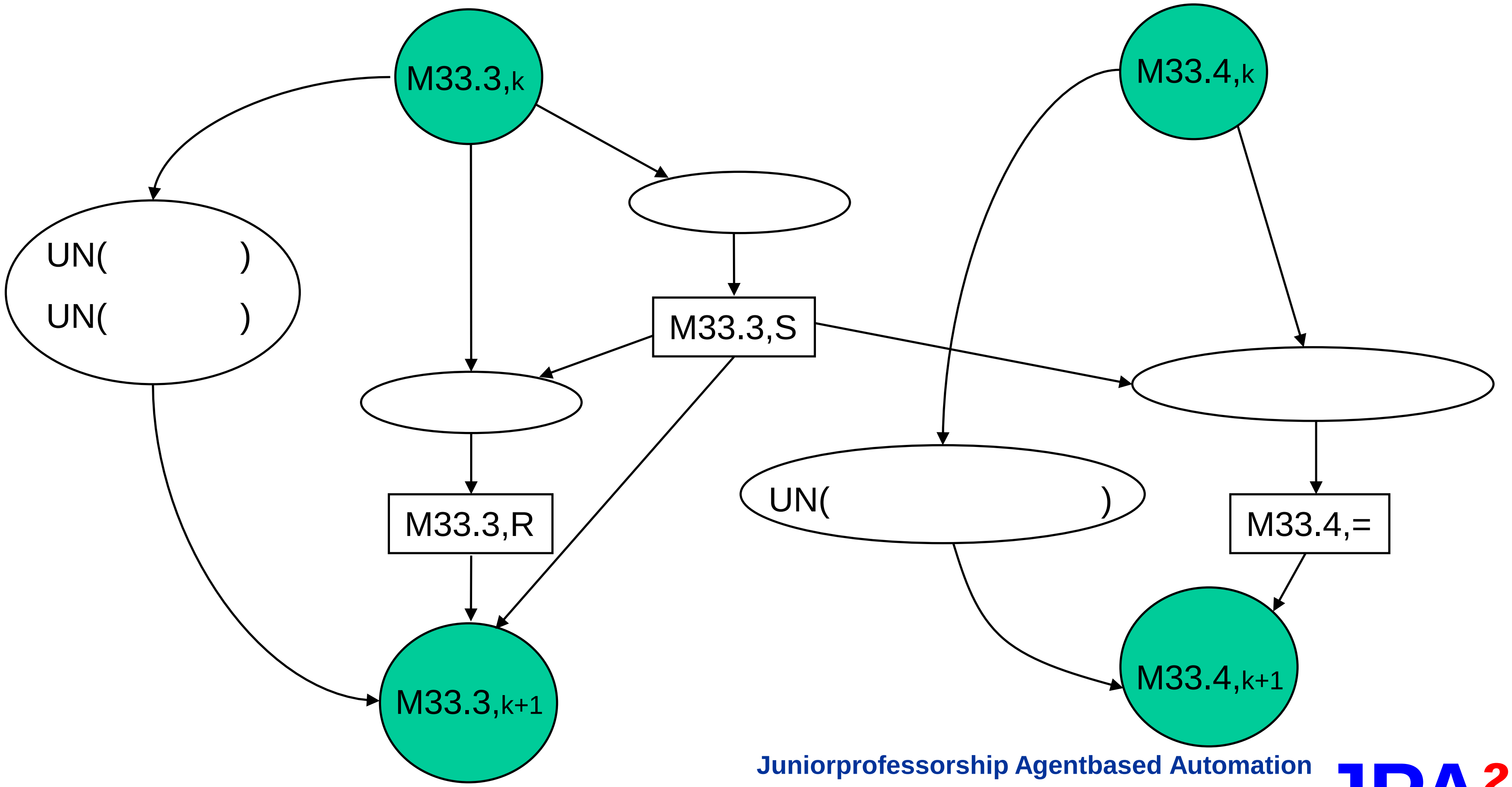
U E 12.1
U(
U M 33.1
S M 33.3
U M 33.3
)

U E 13.6
S M 33.4



Implementierung McCabe

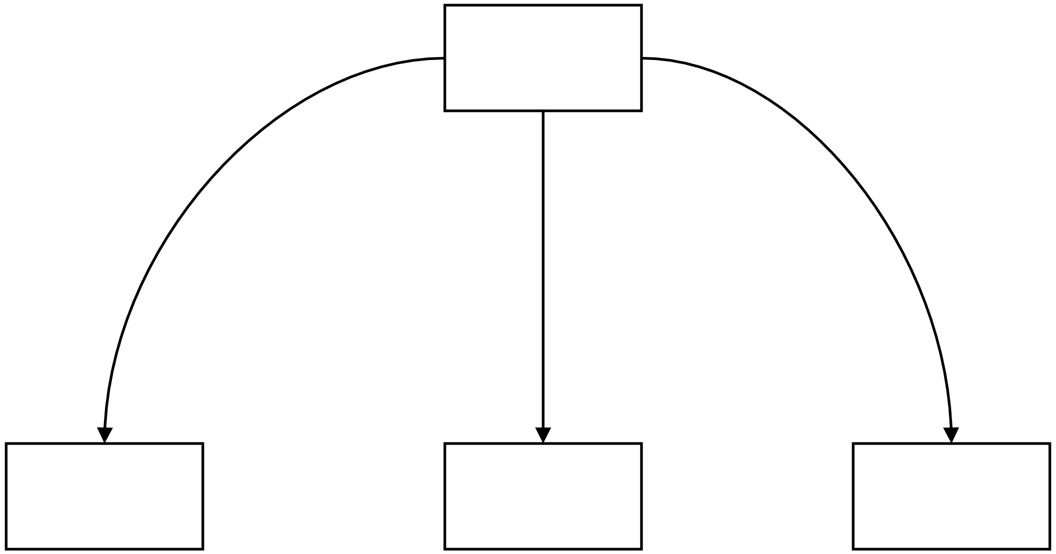
M33.3 S U M33.1	M33.3 R U M32.0	M33.4 = U E12.1 U M33.3



TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Tree Impurity Implementation

```
OB 001

U M10.5
U A12.9
SPB   PB 141
SPA   FB 140
SPA   FB 141
```



$\Rightarrow\ m(G) = 0$

→ Pure Tree Structure; easy Graph

Halstead-Measure

- μ_1: number of distinct Operators
- μ_2: number distinct Operands
- N_1: total number of Operator occurances
- N_2: total number of Operands occurances

$\mu = \mu_1 + \mu_2$: size of the vocabulary

$N = N_1 + N_2$: implementing length

Volume of the program: $V = N \log_2 \mu$

$\Rightarrow$ difficulty: $D = \frac{\mu_1}{2} \cdot \frac{N_2}{\mu_2}$ Effort: $E = V \cdot D$

U; U(; O; O(; S; SI; SV; SE; SS; SA; ZV; ZR; !F; ><F; >F; >=F; <F; <=F; +F; -F; L; LC; T; R; =;)

Operators

E; A; M; T; Z

Operands

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:

Value	**Risk**
1-10	An easy program, low risk
11-20	Complex program, endurable risk
21-50	Very Complex program, high risk
>50	Non testable program, extremely high risk

➢ 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:

$$m(G) = \frac{2(e-n+1)}{(n-1)(n-2)} \qquad 0 \le m(G) \le 1$$

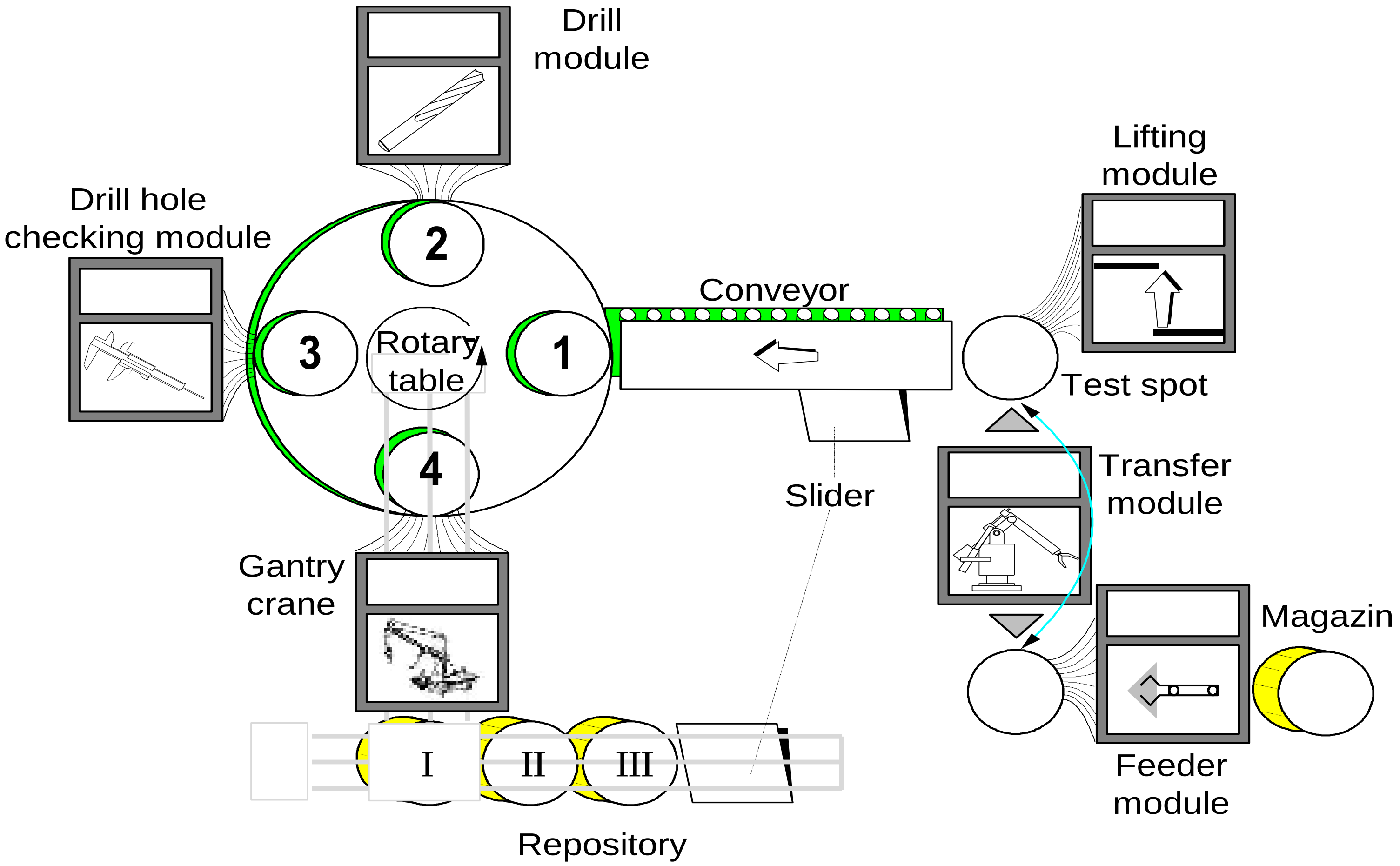
  4. If the value tends to Zero this implies it is an easy graph

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Didactic Case Study

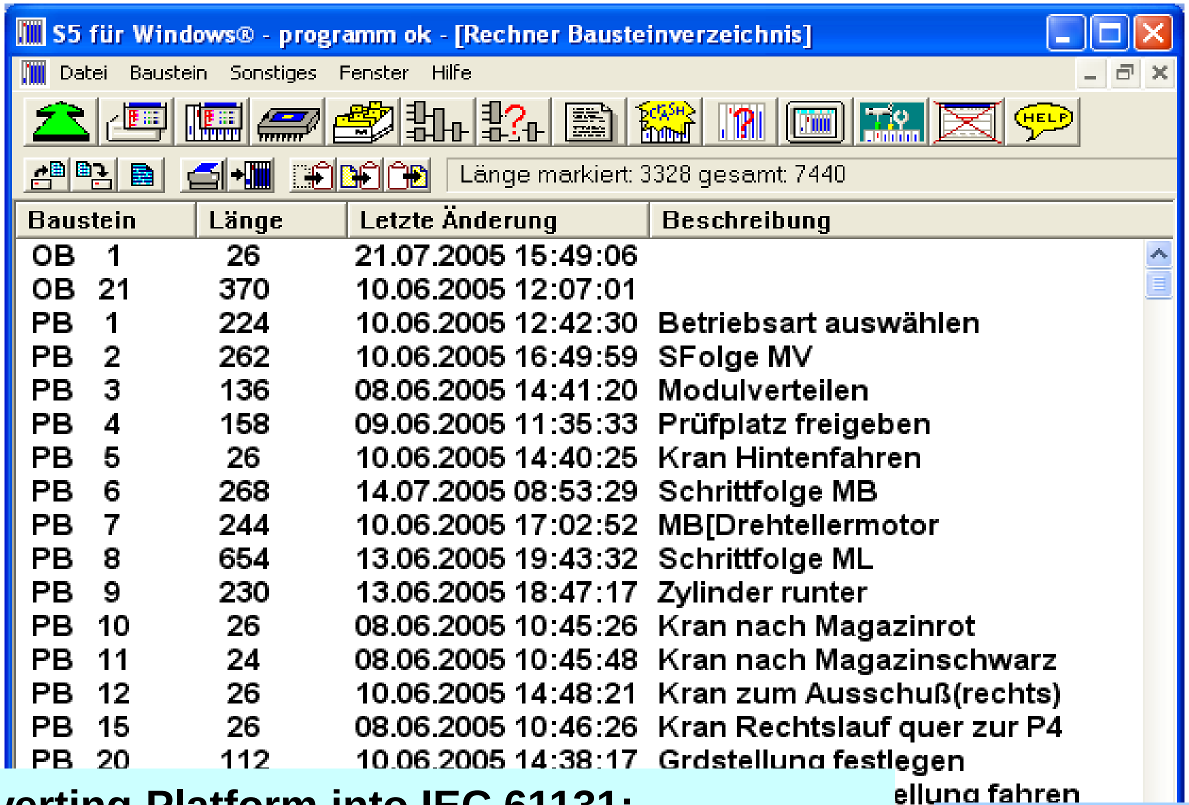
Description of the MPS (FESTO):

The task of this MPS is Sorting, processing, and Lifting of Cylindrical Pieces of different Materials



TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Conversion Process 1


S5 für Windows® - programm ok - [Rechner Bausteinverzeichnis]
Datei Baustein Sonstiges Fenster Hilfe
HELP
Länge markiert: 3328 gesamt: 7440

Baustein	Länge	Letzte Änderung	Beschreibung
OB 1	26	21.07.2005 15:49:06	
OB 21	370	10.06.2005 12:07:01	
PB 1	224	10.06.2005 12:42:30	Betriebsart auswählen
PB 2	262	10.06.2005 16:49:59	SFolge MV
PB 3	136	08.06.2005 14:41:20	Modulverteilen
PB 4	158	09.06.2005 11:35:33	Prüfplatz freigeben
PB 5	26	10.06.2005 14:40:25	Kran Hintenfahren
PB 6	268	14.07.2005 08:53:29	Schrittfolge MB
PB 7	244	10.06.2005 17:02:52	MB[Drehtellermotor
PB 8	654	13.06.2005 19:43:32	Schrittfolge ML
PB 9	230	13.06.2005 18:47:17	Zylinder runter
PB 10	26	08.06.2005 10:45:26	Kran nach Magazinrot
PB 11	24	08.06.2005 10:45:48	Kran nach Magazinschwarz
PB 12	26	10.06.2005 14:48:21	Kran zum Ausschuß(rechts)
PB 15	26	08.06.2005 10:46:26	Kran Rechtslauf quer zur P4
PB 20	112	10.06.2005 14:38:17	Grdstellung festlegen
			ellung fahren

Converting Platform into IEC 61131:

Symbol Table → Global variables

OB 1 → Program (Main)

PBs → Function Block FBs

FBs → Function Block FBs with instance index

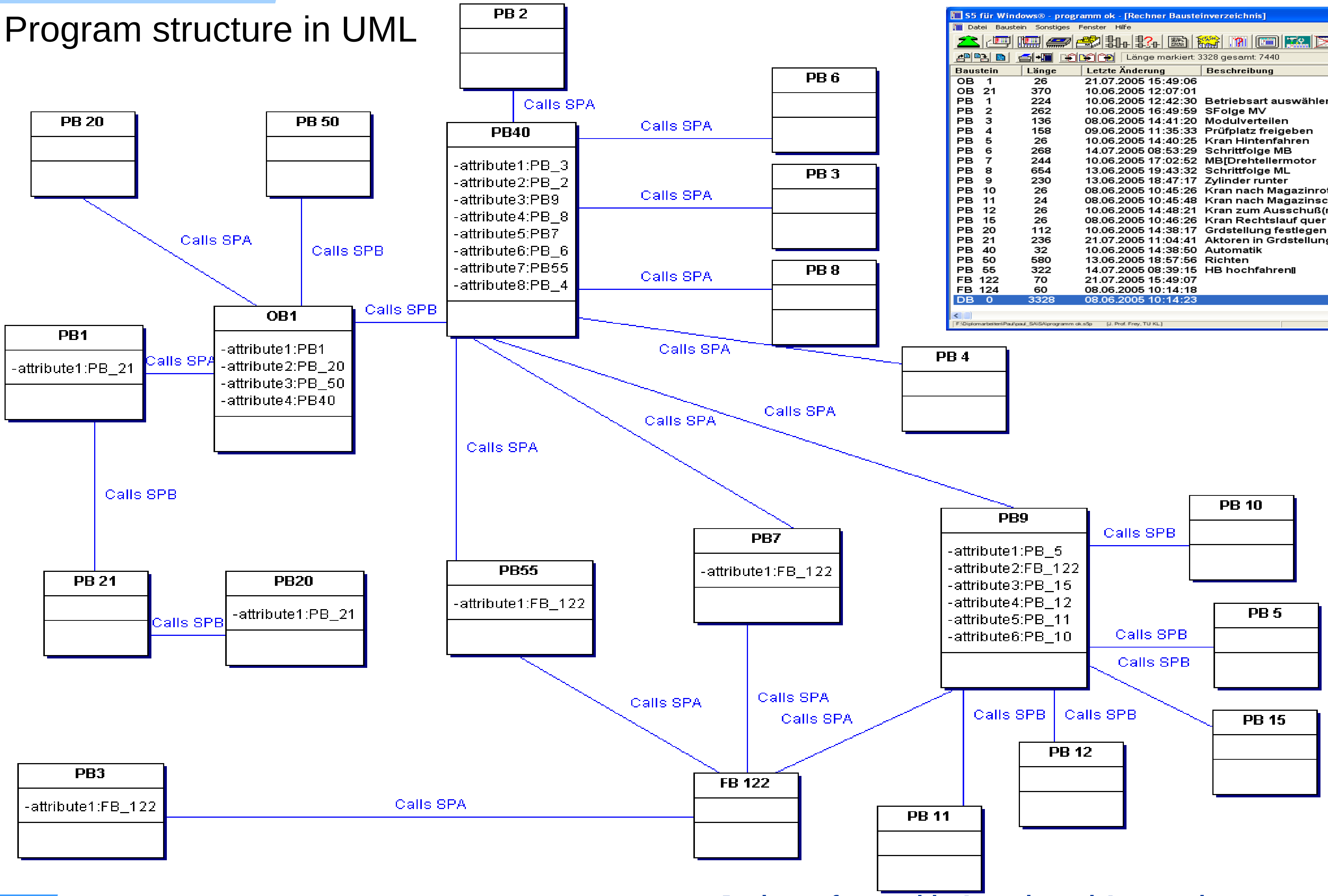
Blocks contains:

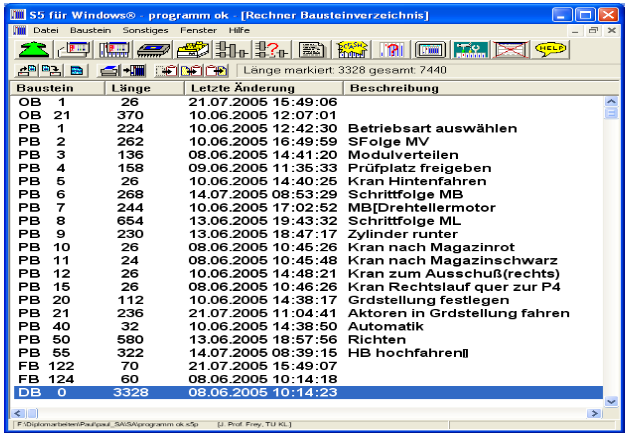
- Binary Operations
- Timer and Counters
- Non-Binary

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Conversion Process 2

Program structure in UML




S5 für Windows® - programm ok - [Rechner Bausteinverzeichnis]
Datei Baustein Sonstiges Fenster Hilfe
Länge markiert: 3328 gesamt: 7440

Baustein	Länge	Letzte Änderung	Beschreibung
OB 1	26	21.07.2005 15:49:06	
OB 21	370	10.06.2005 12:07:01	
PB 1	224	10.06.2005 12:42:30	Betriebsart auswählen
PB 2	262	10.06.2005 16:49:59	SFolge MV
PB 3	136	08.06.2005 14:41:20	Modulverteilen
PB 4	158	09.06.2005 11:35:33	Prüfplatz freigeben
PB 5	26	10.06.2005 14:40:25	Kran Hintenfahren
PB 6	268	14.07.2005 08:53:29	Schrittfolge MB
PB 7	244	10.06.2005 17:02:52	MB[Drehtellermotor
PB 8	654	13.06.2005 19:43:32	Schrittfolge ML
PB 9	230	13.06.2005 18:47:17	Zylinder runter
PB 10	26	08.06.2005 10:45:26	Kran nach Magazinrot
PB 11	24	08.06.2005 10:45:48	Kran nach Magazinschwarz
PB 12	26	10.06.2005 14:48:21	Kran zum Ausschuß(rechts)
PB 15	26	08.06.2005 10:46:26	Kran Rechtslauf quer zur P4
PB 20	112	10.06.2005 14:38:17	Grdstellung festlegen
PB 21	236	21.07.2005 11:04:41	Aktoren in Grdstellung fahren
PB 40	32	10.06.2005 14:38:50	Automatik
PB 50	580	13.06.2005 18:57:56	Richten
PB 55	322	14.07.2005 08:39:15	HB hochfahren
FB 122	70	21.07.2005 15:49:07	
FB 124	60	08.06.2005 10:14:18	
DB 0	3328	08.06.2005 10:14:23	

F:\Diplomarbeiten\Paul\paul_SA\SA\programm ok.s5p [J. Prof. Frey, TU KL]

Symbol Table

Operand	Symbol	Kommentar
E 0.0	1K1	STEUERUNG AUS/EIN
E 0.2	1S10	LAMPENTEST
…	…	…
A 0.0	1H5	STEUERUNG
A 0.2	1H7	EINRICHTEN
…	…	…
M 0.0	M0.0	VKE = 0 FUER BCD WANDLUNG + VORZEICHEN
M 0.2	M0.2	RESET STOERMELDUNGEN
…	…	…
MB 120	MB120	STOERUNGEN S.WALZE FUER TEXTANZEIGE

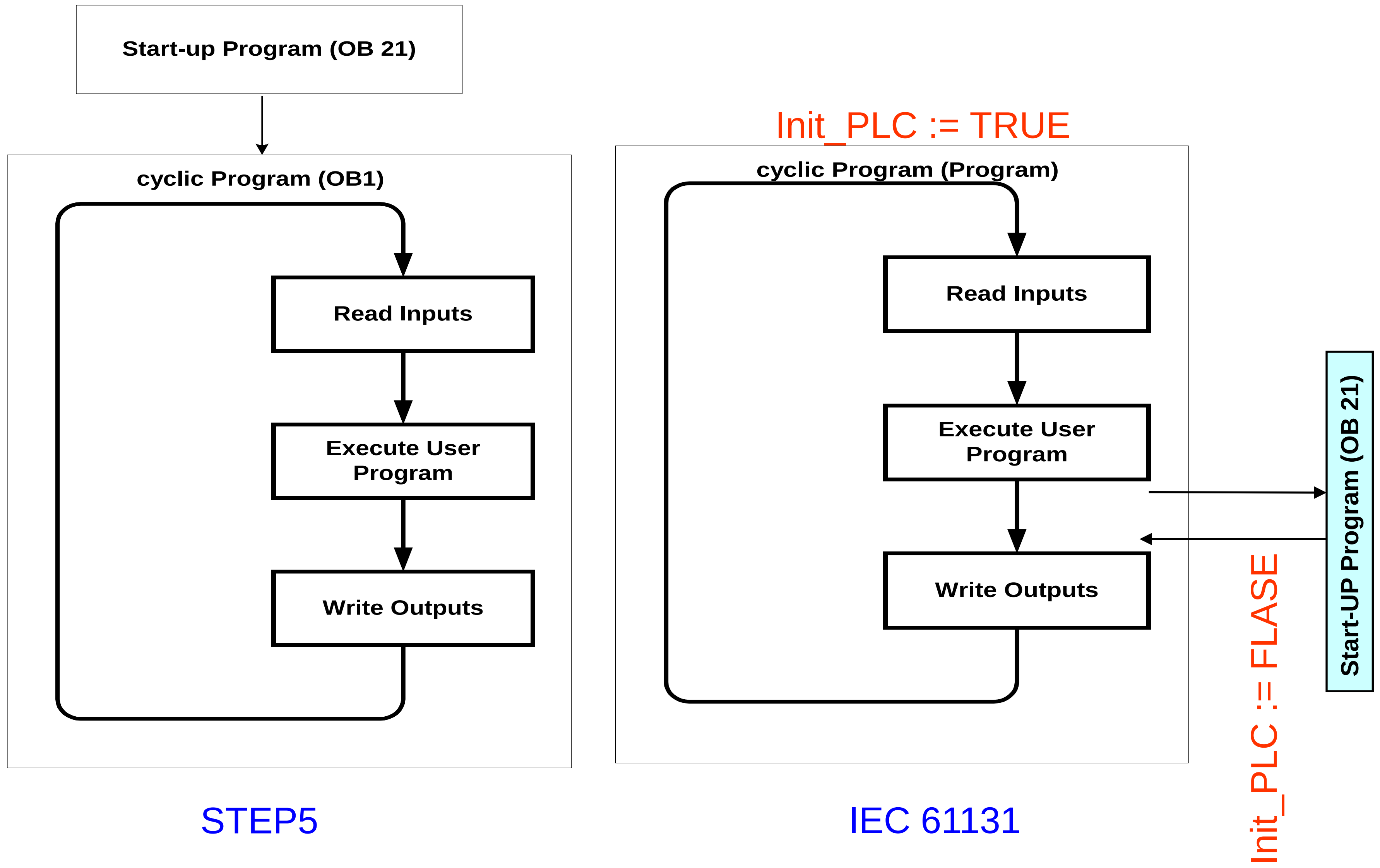
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

S5 to IEC Keywords

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

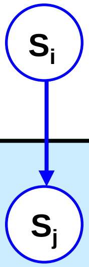
S5 in FSM	IEC 61131-3
AND	AND
ANDN	ANDN
OR	OR
ORN	ORN
<, <=	LT, LE
>, >=	GT, GE
><	NE
=	=
+	ADD
-	SUB
*	MUL
/	DIV

S5 to IEC 61131 Cycle



Re-Implementation of Binary PLC programs

- Binary Instrucations
  - 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 →IEC Instructions
- Example



```
O: 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
```

```
<?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>
    :
    :
 </fsm>
```

IEC 61131

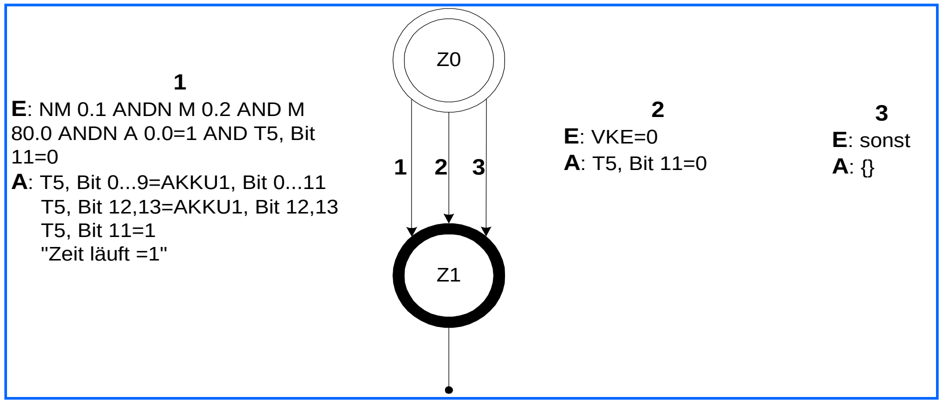
```
VAR
    M100_0: BOOL;
END_VAR
VAR_GLOBAL
    E38_2 AT IX38.2: BOOL;
    E38_1 AT IX38.1: BOOL;
    E38_3 AT IX38.3: BOOL;
END_VAR
```

```
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:
```

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


Z0
Z1
1
2
3
1
E: NM 0.1 ANDN M 0.2 AND M 80.0 ANDN A 0.0=1 AND T5, Bit 11=0
A: T5, Bit 0...9=AKKU1, Bit 0...11
T5, Bit 12,13=AKKU1, Bit 12,13
T5, Bit 11=1
"Zeit läuft =1"
2
E: VKE=0
A: T5, Bit 11=0
3
E: sonst
A: {}

Example: STEP 5

```
[3  Richten blinkt

    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
VAR_OUTPUT
    A0_1: BOOL;
END_VAR
```

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

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

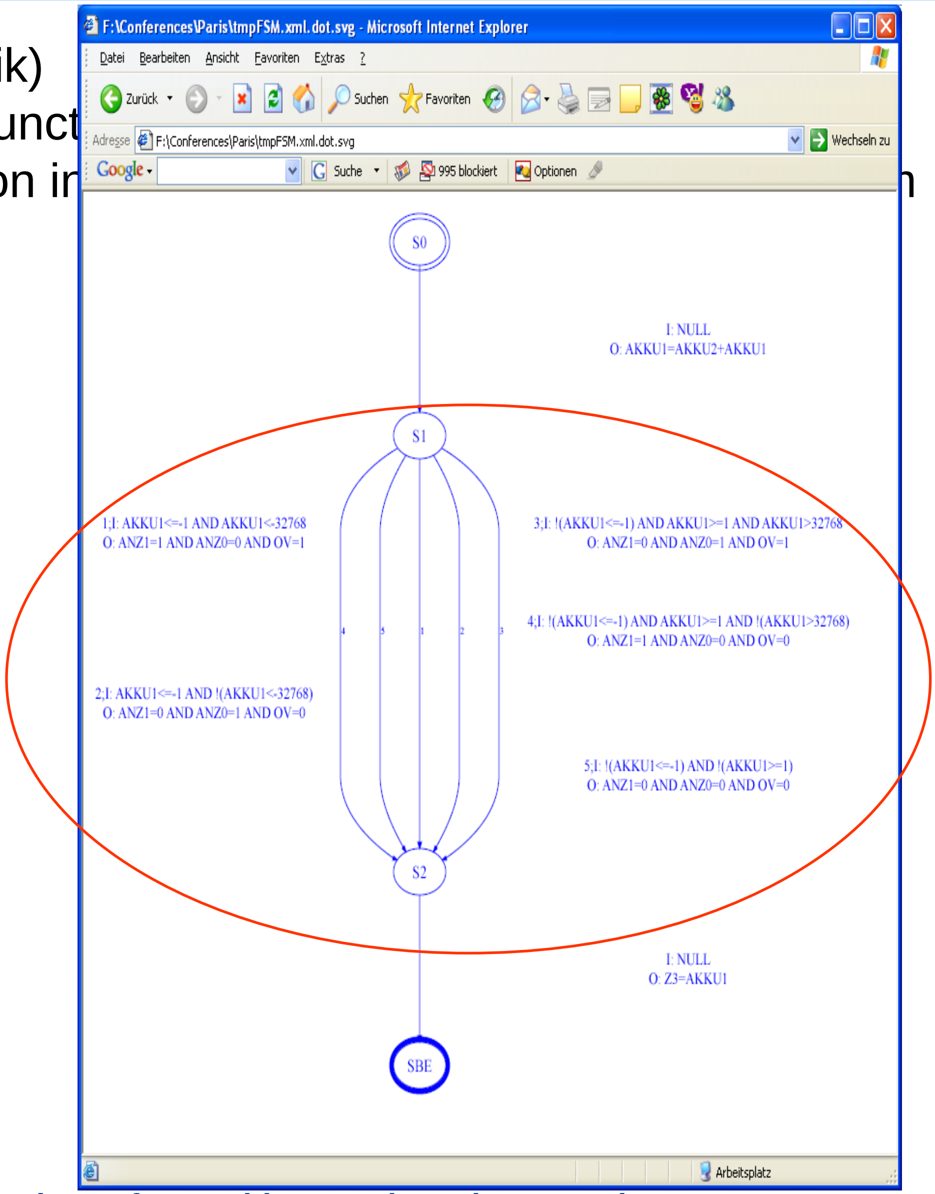
Non-Binary PLC programs

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

- From the FSM (Logik)
- A new Function or funct
- non-binary instruction ir   n

Example: STEP 5

```
[1
NAME:     ADD
BEZ :     Z1        EW
BEZ :     Z2        EW
BEZ :     Z3        AW

          L         =Z1
          L         KF +800
          +F
          T         =Z3
          BE
]
```


F:\Conferences\Paris\tmpFSM.xml.dot.svg - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten
Adresse F:\Conferences\Paris\tmpFSM.xml.dot.svg
Wechseln zu
Google Suche 995 blockiert Optionen
S0
I: NULL
O: AKKU1=AKKU2+AKKU1
S1
1;I: AKKU1<=-1 AND AKKU1<-32768
O: ANZ1=1 AND ANZ0=0 AND OV=1
3;I: !(AKKU1<=-1) AND AKKU1>=1 AND AKKU1>32768
O: ANZ1=0 AND ANZ0=1 AND OV=1
4
5
1
2
3
4;I: !(AKKU1<=-1) AND AKKU1>=1 AND !(AKKU1>32768)
O: ANZ1=1 AND ANZ0=0 AND OV=0
2;I: AKKU1<=-1 AND !(AKKU1<-32768)
O: ANZ1=0 AND ANZ0=1 AND OV=0
5;I: !(AKKU1<=-1) AND !(AKKU1>=1)
O: ANZ1=0 AND ANZ0=0 AND OV=0
S2
I: NULL
O: Z3=AKKU1
SBE
Arbeitsplatz

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

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

IEC 61131

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

Example: STEP 5 (PB 1)

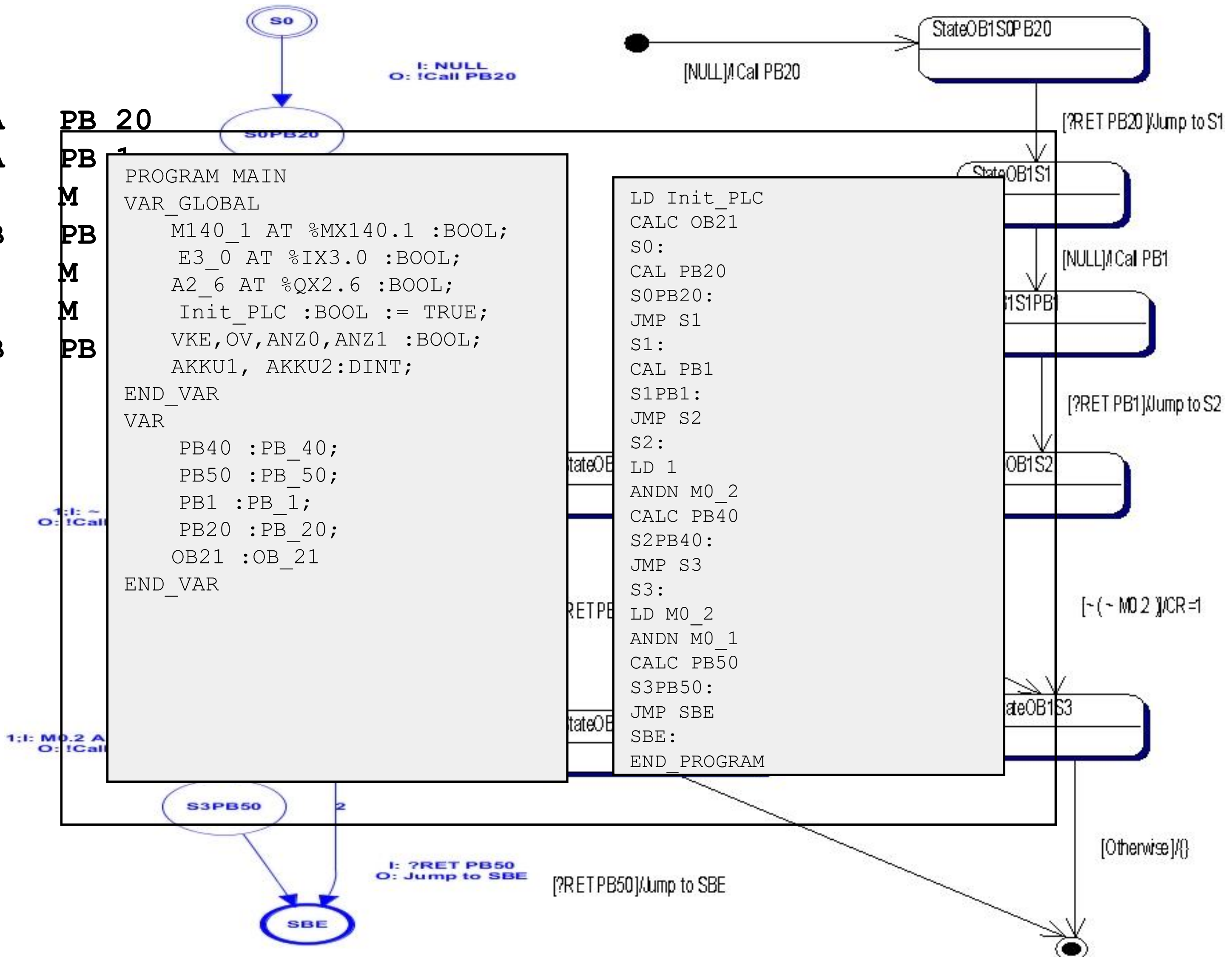
```
[1
          A         DB 2
          SPA       FB 2
NAME:     ADD
Z1   :    DW 3
Z2   :    DW 4
Z3   :    DW 5

          L         DW 5
          T         MW 10
          BE
]
```

```
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
FB2_1(Z1:=DB2[3],Z2:=DB2[4]|DB2[5
]:=Z3)
S1FB2:
JMP S2
S2:
LD DB2[5]
ST MW10
SBE:
END_FUNCTION_BLOCK
```

Conversion Process 3

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

```
[1
SPA    PB 20
SPA    PB 1
UN     M
SPB    PB
U      M
UN     M
SPB    PB
BE
]
```



```
PROGRAM MAIN
VAR_GLOBAL
    M140_1 AT %MX140.1 :BOOL;
     E3_0 AT %IX3.0 :BOOL;
    A2_6 AT %QX2.6 :BOOL;
     Init_PLC :BOOL := TRUE;
    VKE,OV,ANZ0,ANZ1 :BOOL;
    AKKU1, AKKU2:DINT;
END_VAR
VAR
     PB40 :PB_40;
     PB50 :PB_50;
     PB1  :PB_1;
     PB20 :PB_20;
    OB21 :OB_21
END_VAR
```

```
LD Init_PLC
CALC OB21
S0:
CAL PB20
S0PB20:
JMP S1
S1:
CAL PB1
S1PB1:
JMP S2
S2:
LD 1
ANDN M0_2
CALC PB40
S2PB40:
JMP S3
S3:
LD M0_2
ANDN M0_1
CALC PB50
S3PB50:
JMP SBE
SBE:
END_PROGRAM
```

Conversion of OB1

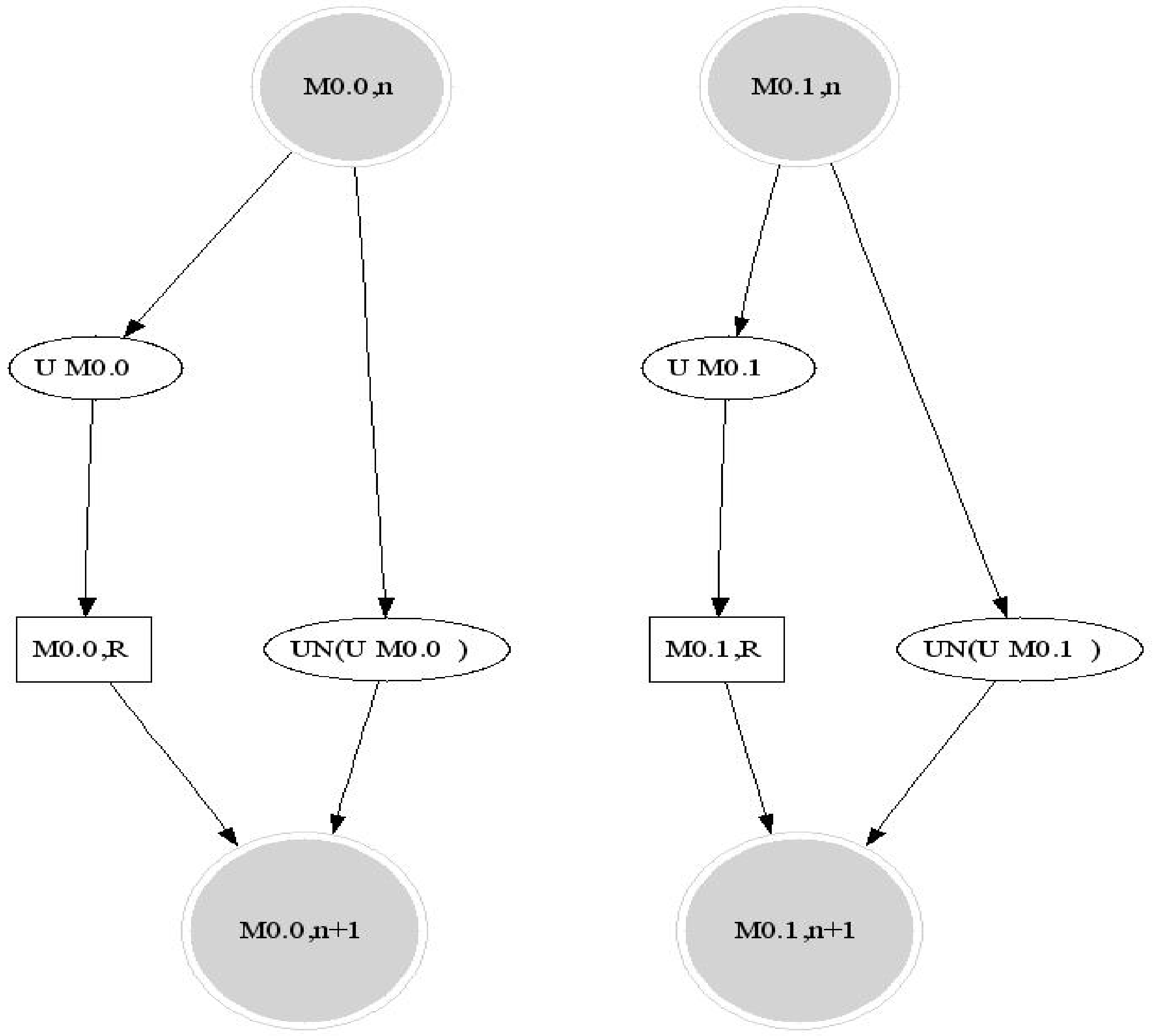
SW Quality (LOC)

Block	LOC (NCSS)	Block	LOC (NCSS)
FB122	8	PB3	20
OB1	8	PB4	56
OB21	177	PB40	9
PB1	72	PB5	6
PB10	6	PB50	209
PB11	5	PB55	99
PB12	6	PB6	97
PB15	6	PB7	70
PB2	104	PB8	261
PB20	40	PB9	56
PB21	81	**total NCSS:**	**1403**

SW Quality (Halstead)

Block	Volume	Difficulty	Effort
FB122	0	0	0
OB1	0	0	0
OB21	2864	1	2864
PB1	828	8	6624
PB10	32	2	64
PB11	22	2	44
PB12	32	2	64
PB15	32	2	64
PB2	998	15	14970
PB20	357	2	714
PB21	1237	8	9896
PB3	104	1	104
PB4	495	9	4455
PB40	0	0	0
PB5	32	2	64
PB50	3281	40	131240
PB55	862	15	12930
PB6	1261	12	15132
PB7	673	6	4038
PB8	3363	20	67260
PB9	390	4	1560
Average	937	8	15115

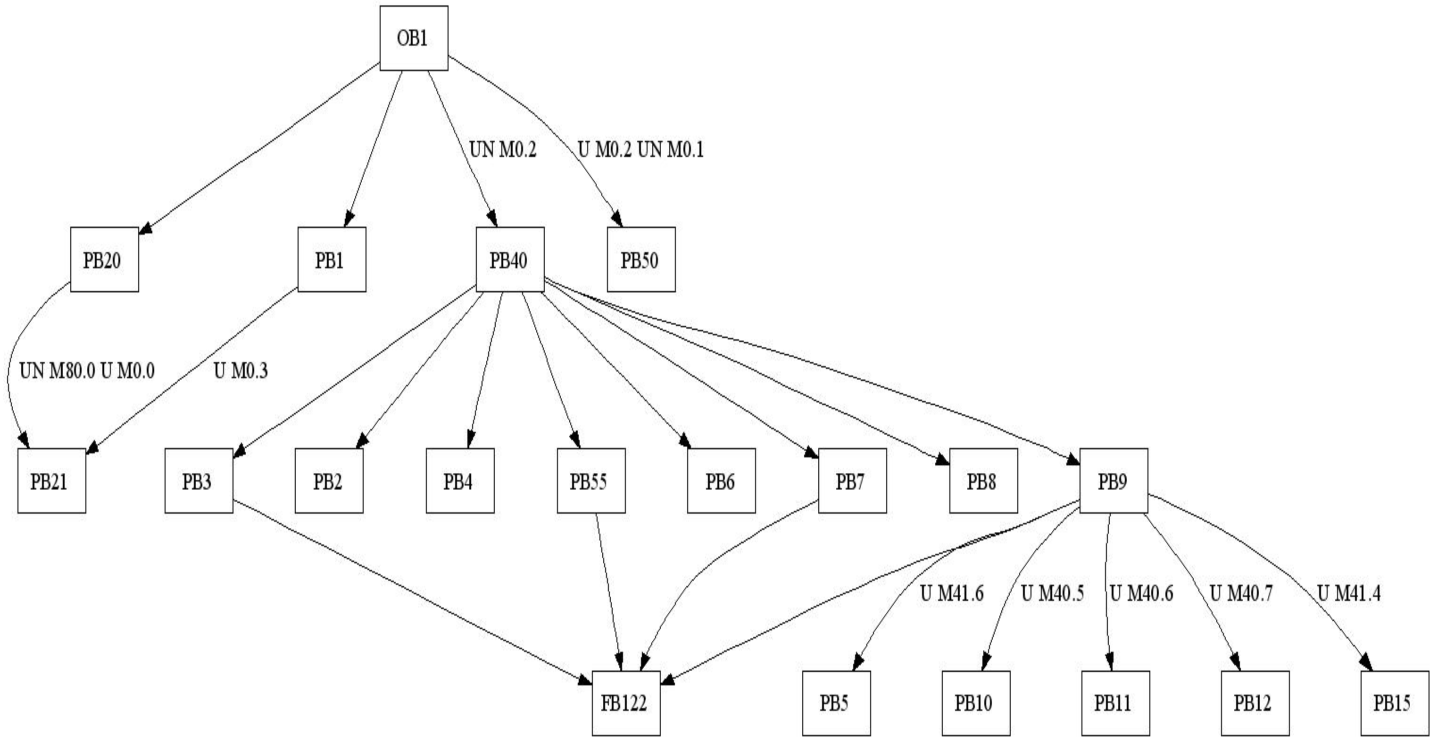
SW Quality (McCabe)



OB 21 Segment

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

SW Quality (Tree Impurity)



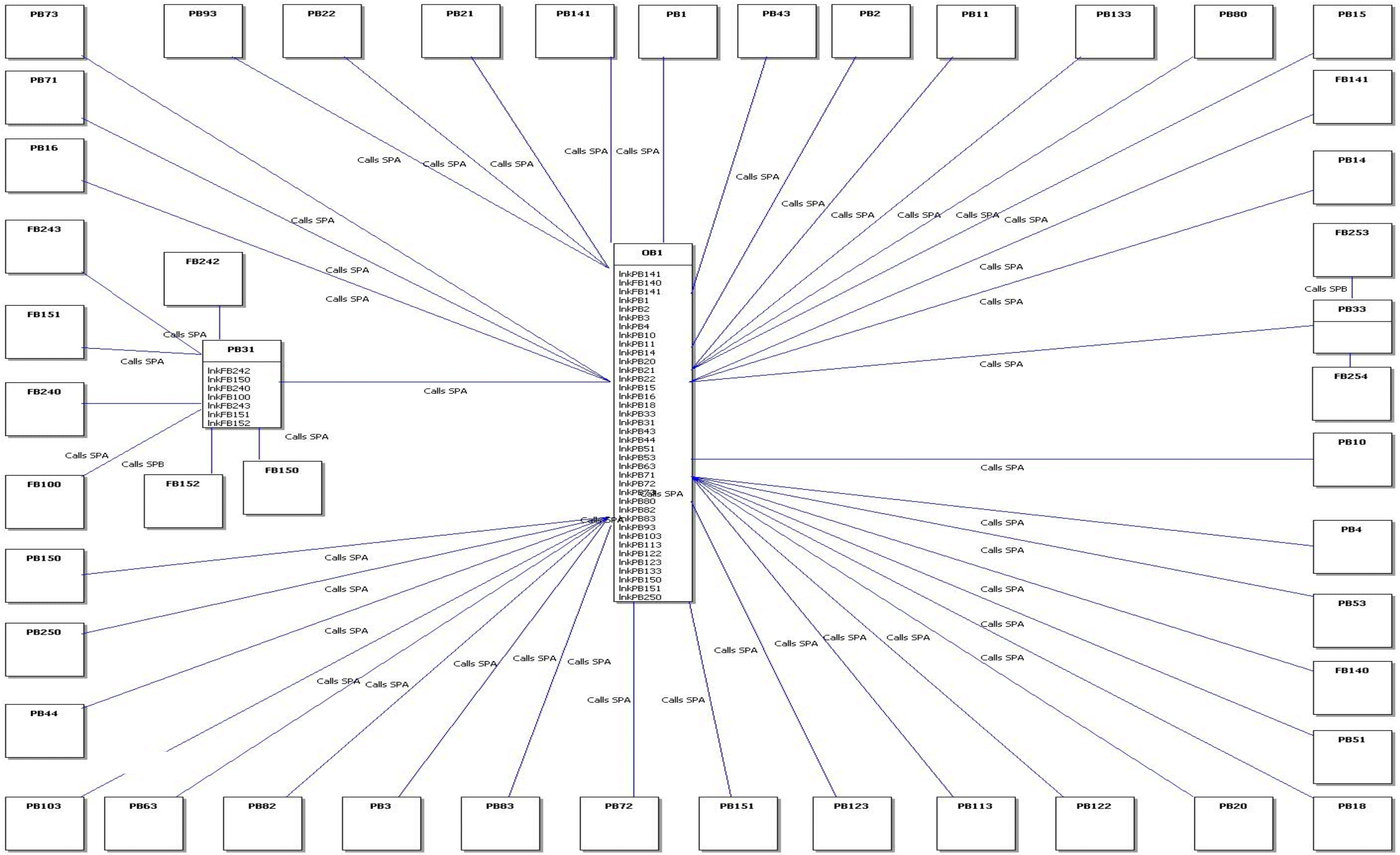
Tree Impurity = 0.023391813

Industrial Case Study

Freudenberg (PK14)

Main goal was the diagnosability

Conversion Process 1



Program structure in UML

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Conversion Process 2

Introduction

Re-Engineering

Formalization

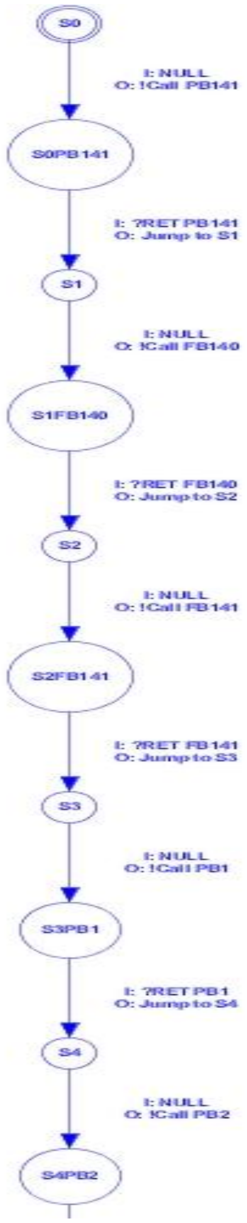
Visualization

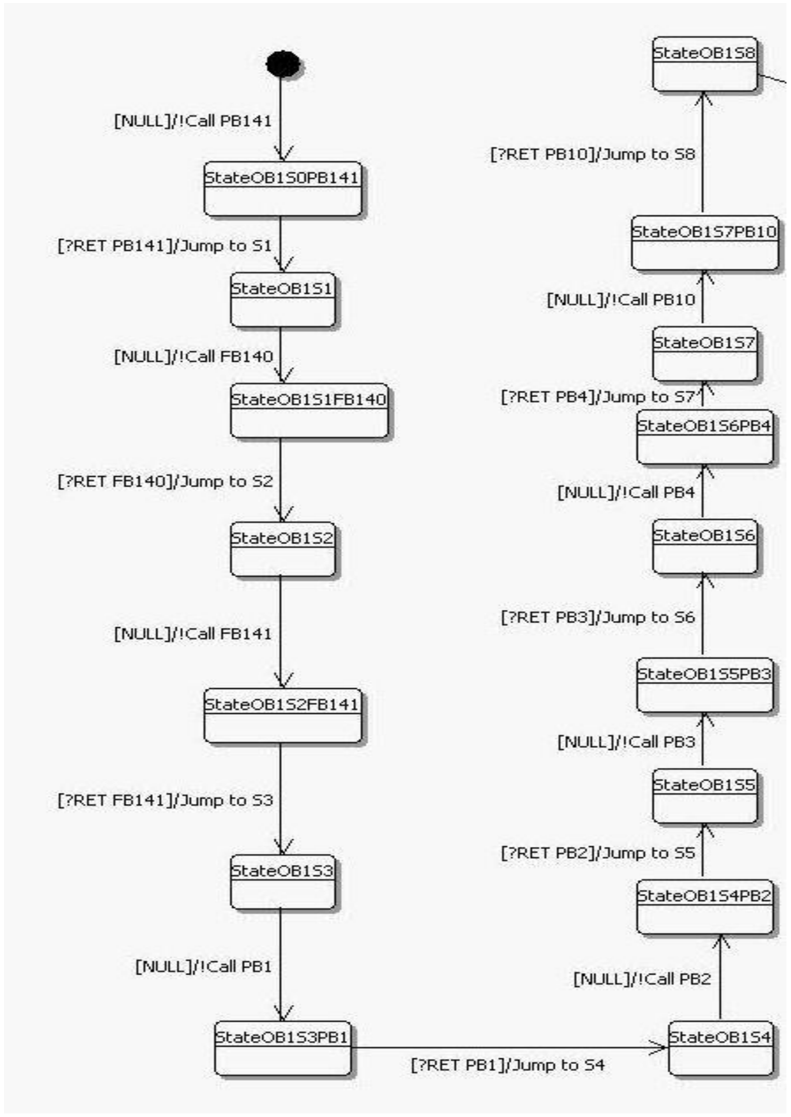
Re-Implem.

SW-Quality

Case Stud.

Summary





Conversion of OB1 segment

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

SW Quality (LOC)

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

Block	NCSS	Block	NCSS	Block	NCSS	Block	NCSS
FB100	3	FB101	3	PB113	15	PB122	131
FB140	10	FB141	10	PB123	46	PB133	55
FB150	10	FB151	6	PB14	386	PB141	76
FB152	6	FB153	63	PB15	57	PB150	62
FB154	21	FB240	2	PB151	30	PB16	314
FB241	2	FB242	2	PB18	173	PB19	1
FB243	2	FB244	2	PB2	50	PB20	93
FB245	2	FB246	2	PB21	179	PB22	79
FB247	2	FB248	2	PB250	479	PB251	13
FB249	2	FB250	2	PB252	5	PB3	43
FB251	2	FB252	27	PB31	113	PB33	34
FB253	14	FB254	29	PB4	17	PB43	39
FB255	30	OB1	39	PB44	39	PB51	18
OB13	2	OB21	3	PB53	8	PB63	80
OB22	3	OB31	2	PB71	20	PB72	24
PB1	36	PB10	209	PB73	63	PB80	62
PB103	43	PB11	23	PB82	20	PB83	34
PB93	19	**total NCSS**	**3493**				

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

SW Quality (Halstead)

Block	Volume	Difficulty	Effort	Block	Volume	Difficulty	Effort
FB140	36.0	6.0	216.0	PB16	3847.0	18.0	69246.0
FB141	36.0	6.0	216.0	PB18	1939.0	15.0	29085.0
PB1	337.0	5.0	1685.0	PB2	456.0	4.0	1824.0
PB10	2246.0	16.0	35936.0	PB20	801.0	10.0	8010.0
PB103	342.0	2.0	684.0	PB21	2077.0	9.0	18693.0
PB11	172.0	4.0	688.0	PB22	656.0	2.0	1312.0
PB113	91.0	2.0	182.0	PB250	6648.0	14.0	93072.0
PB122	1381.0	15.0	20715.0	PB3	321.0	3.0	963.0
PB123	407.0	3.0	1221.0	PB31	899.0	7.0	6293.0
PB133	677.0	3.0	2031.0	PB33	221.0	6.0	1326.0
PB14	4398.0	30.0	131940.0	PB4	143.0	3.0	429.0
PB141	657.0	1.0	657.0	PB43	296.0	5.0	1480.0
PB15	549.0	8.0	4392.0	PB44	296.0	5.0	1480.0
PB150	540.0	10.0	5400.0	PB51	135.0	8.0	1080.0
PB151	253.0	4.0	1012.0	PB53	42.0	3.0	126.0
PB80	564.0	12.0	6768.0	PB63	792.0	5.0	3960.0
PB82	154.0	3.0	462.0	PB71	167.0	3.0	501.0
PB83	265.0	4.0	1060.0	PB72	188.0	3.0	564.0
PB93	131.0	2.0	262.0	PB73	585.0	4.0	2340.0
Average	888.03	6.92	12034.5				

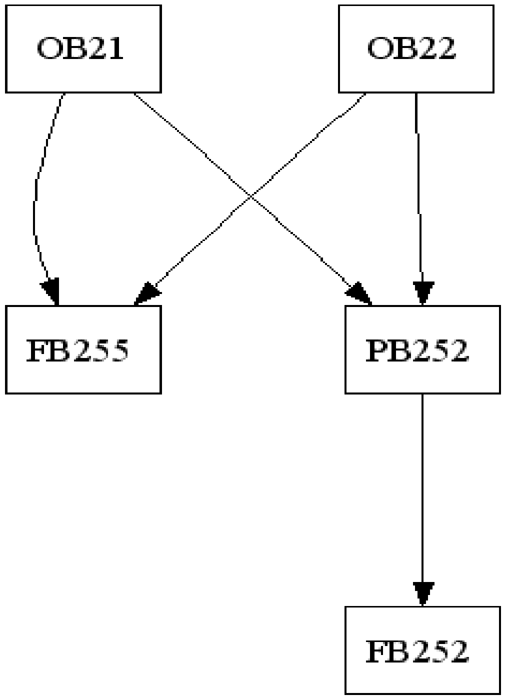
SW Quality (McCabe)

Block	Value	Risk	Block	Value	Risk
FB140	2	easy Program, low Risk	PB16	60	untestable Program, extremely high Risk
PB10	38	Very Complex Program, High Risk	PB18	44	Very Complex Program, High Risk
PB44	9	easy Program, low Risk	PB2	14	Complex Program, endurable Risk
PB4	8	easy Program, low Risk	PB20	11	Complex Program, endurable Risk
PB103	2	easy Program, low Risk	PB21	6	easy Program, low Risk
PB11	9	easy Program, low Risk	PB22	2	easy Program, low Risk
PB113	2	easy Program, low Risk	PB3	2	easy Program, low Risk
PB122	20	Very Complex Program, High Risk	PB250	132	untestable Program, extremely high Risk
PB123	2	easy Program, low Risk	PB31	18	Complex Program, endurable Risk
PB133	6	easy Program, low Risk	PB33	15	Complex Program, endurable Risk
PB14	55	untestable Program, extremely high Risk	PB141	64	untestable Program, extremely high Risk
FB141	2	easy Program, low Risk	PB43	9	easy Program, low Risk
PB15	12	Complex Program, endurable Risk	PB80	14	Complex Program, endurable Risk
PB150	3	easy Program, low Risk	PB51	8	easy Program, low Risk
PB151	5	easy Program, low Risk	PB53	2	easy Program, low Risk
PB1	10	Complex Program, endurable Risk	PB63	10	Complex Program, endurable Risk
PB82	5	easy Program, low Risk	PB71	5	easy Program, low Risk
PB83	5	easy Program, low Risk	PB72	7	easy Program, low Risk
PB93	2	easy Program, low Risk	PB73	5	easy Program, low Risk

SW Quality (Tree Impurity)

Tree Impurity = -6.4935064E-4

→ 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

Technische Universität Kaiserslautern

Summary

- ◎ 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 → IEC 61131
- ◎ SW Quality derivation of the PLC program

Discussion

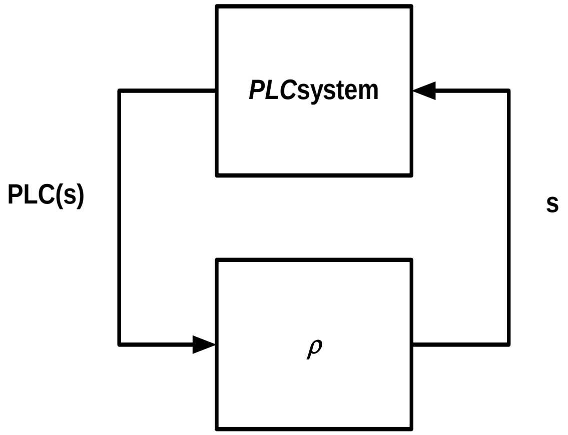
Oops clicked firmly ☺

Additional Slides

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

PLC as Discrete Event System

- *PLC*$_{\text{system}}$/ρ → *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 ρ
- Σ finite alphabet of ρ
- δ partial transition function mapping $X \times \Sigma$ *to X*

→ $\delta(x_i, e)$ *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 $\langle PLC_{\text{SW}}, PLC_{\text{HW}}, PLC_{\text{Cycle}} \rangle$
- *PLC*$_{\text{SW}}$ PLC program as tuple

$$\langle PAE, PAA, I, A_{PAE}, PLC_{\rho r}, x_0, x_f \rangle$$

- *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 $\{0,1\}^{|I|}$ which is the alphabet generated by the nonempty ordered set of variables I
- PLCρr is the PLC program described as a partial function

$$PLC_{\rho r}(x,e) : \alpha(I) \times \alpha(PAA) \times A_{\text{PAE}} \rightarrow \alpha(I) \times \alpha(PAA)$$

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

- A_{PAE} ⊆ ℘(*PAE*) of recognized inputs

PLC Blocks as FSM

- *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
- $\sum = \alpha(PAE)$ input alphabet

- *b* ∈ *Binary* range over binary variables
- *bexpr* is derived which ranges over Boolean expressions
- *bexpr* ∈ *Bexpr+* where *Bexpr+ is the language generated by Gexpr* grammar

Gexpr = 1|0|b|~b|~(Gexpr)

| (Gexpr ∧ Gexpr)|(Gexpr ∨ Gexpr)

| (Gexpr ≡ Gexpr)|(Gexpr ❄ Gexpr)

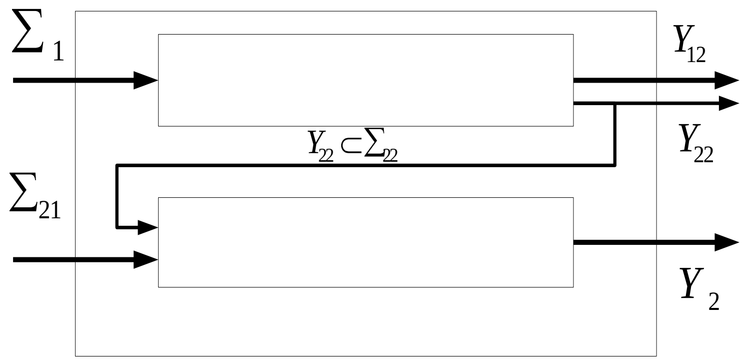
→*Gexpr* is also an alphabet since ∆🕯☎ are called closed binary on pow(υ) where υ is the universe

PLC Program as Communicating Automata

- *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*$_{M1}$…..*PLC*$_{Mn}$ of $\langle S_i, \Sigma_i, Y_i, \delta_i, \lambda_i, s_{0,i} \rangle$
- The model *PLC*$_{Mi}$ $\forall i \in \{1,\ldots,n\}$ PLC$_{M1}$⊗PLC$_{M2}$…⊗PLC$_{Mn}$ builds the automaton PLCu:= ↑*S*, ●, *Y*,*δ*,*λ*,*s*$_0$ ⇧ such that in case of no Sync.

General feed-forward composition

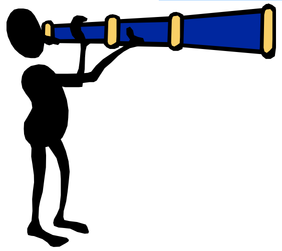


$$Y_i = Y_{i1} \times Y_{i2}$$
$$\Sigma_i = \Sigma_{i1} \times \Sigma_{i2}$$

Technische Universität Kaiserslautern

Industrial Case Study

Outlook



- ⇨ 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: $m(G)=\frac{2(e-n+1)}{(n-1)(n-2)}$ $\quad 0 \le m(G) \le 1$

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