

Modularity and architecture of PLC-based software for automated production Systems: An analysis in industrial companies

1. Introduction and Motivation
2. SWAT4aPS concept
3. SWAT4aPS hypothesis
4. Selected results
5. Overall maturities
6. SWAT4aPS+ and Outlook

Univ.-Prof. Dr.-Ing. Birgit Vogel-Heuser

Ordinaria, full professor, head of institute
Automation and Information Systems (AIS)
Department of Mechanical Engineering,
Technical University Munich
www.ais.mw.tum.de; vogel-heuser@tum.de

SE 2018, Universität Ulm
March 8th, 2018

B. Vogel-Heuser, J. Fischer, S. Feldmann, S. Ulewicz and S. Rösch. "Modularity and Architecture of PLC-based Software for Automated Production Systems: An analysis in industrial companies", *Journal of Systems and Software (JSS)*, vol. 131, pp. 35-62, May 2017.



Technical Constraints of aPS and Motivation

Technical constraints of automated Production Systems (aPS):

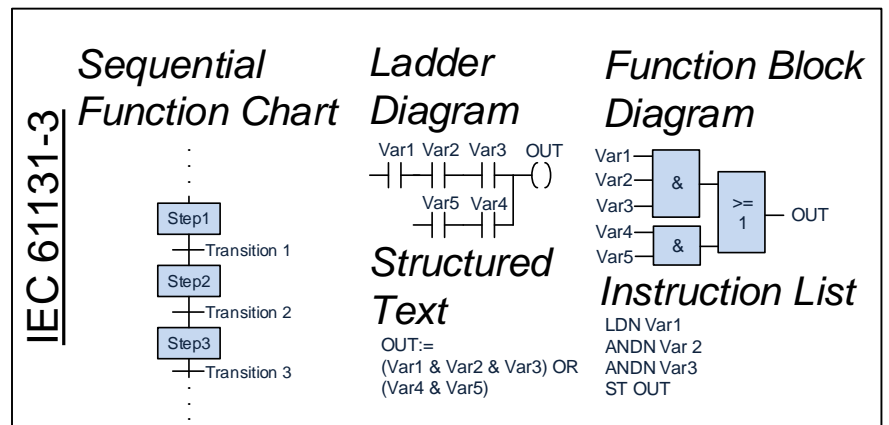
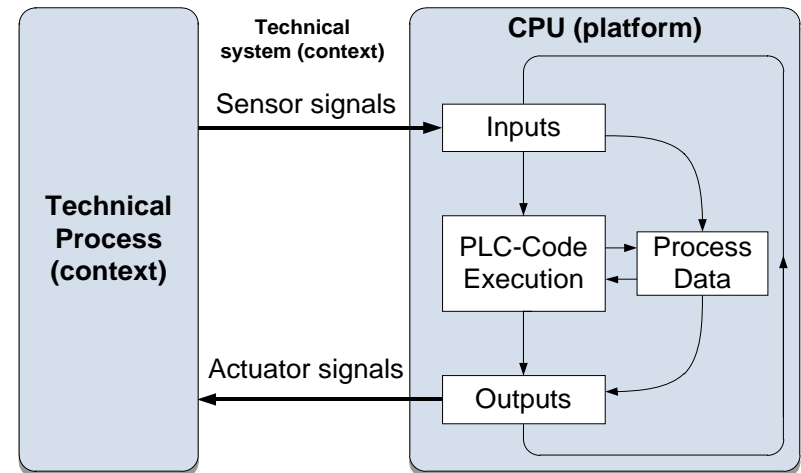
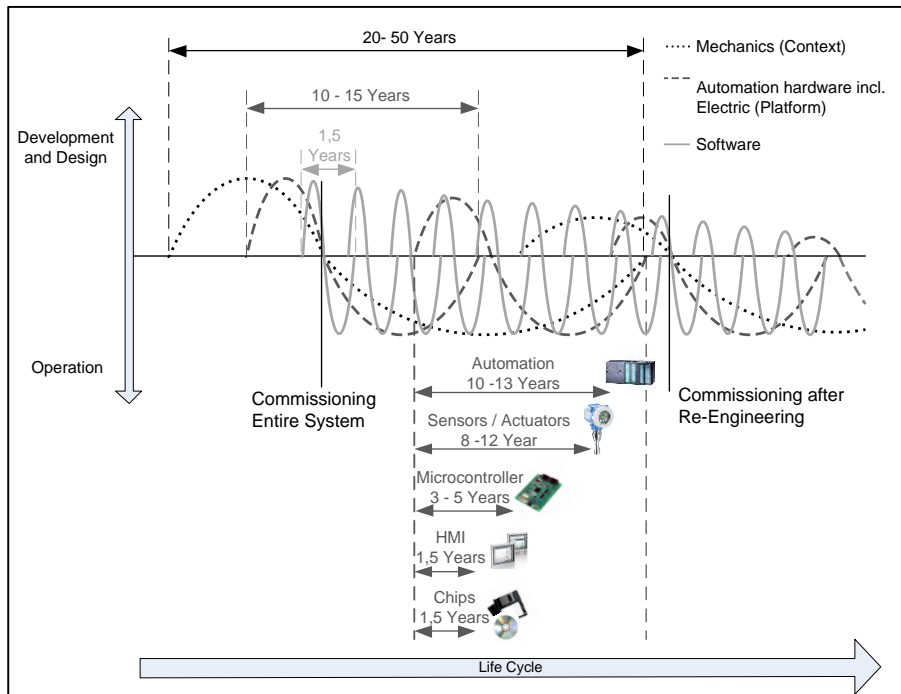
- Hard real-time requirements, cyclic behavior ($1\mu\text{s} - 1\text{s}$), and proprietary hardware (PLC).
- Online change is mandatory
- Domain specific programming language (IEC 61131-3)



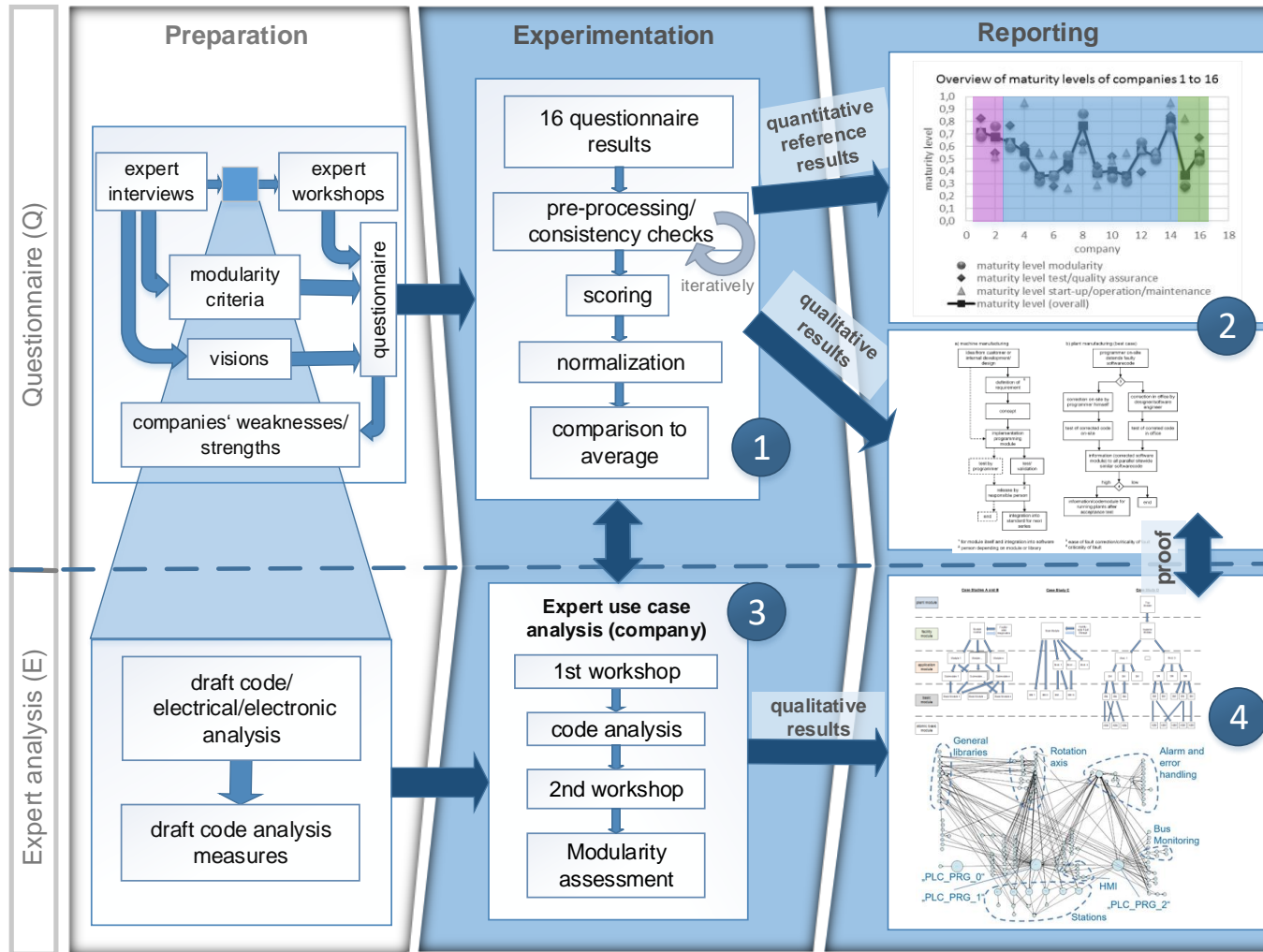
Source: Siemens AG



Source: Bayer AG, Leverkusen



Software Maturity for aPS (SWMAT4aPS)-Benchmark process to identify strengths and weaknesses in software modularity



Categorization of companies

- library and platform providers (1 and 2)
- machine suppliers (3–14)
- plant suppliers (15–16)

16 world-leading companies in machine and plant manufacturing including four case studies

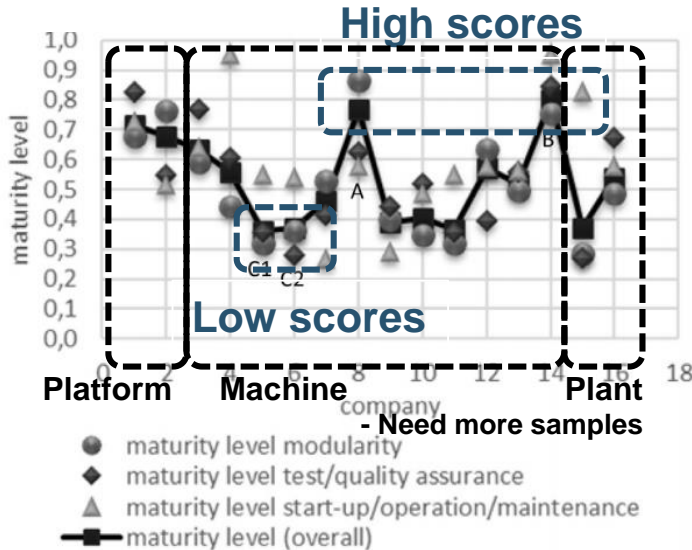
Research questions and hypotheses

Research Questions	Related Hypotheses	Proof
Does the questionnaire deliver valid results to identify weaknesses in gaining software modularity of aPS? (RQ1)	Questionnaire delivers valid results (H1.1)	Q&E
	Maturity level: Platform suppliers > Machine suppliers > Plant manufacturers (H1.2)	Q
Do the three different sub-maturity levels deliver further insights compared to one general maturity level? (RQ2)	Maturity level differ among M_{MOD} , M_{TEST} , M_{OP} (H2)	Q
What are the most significant weaknesses in software maturity in aPS and in which phase do they occur and what are possible causes / reasons / prerequisites? (RQ3)	Universally low maturity levels arise in the different phases, indicating possible causes or prerequisites for weaknesses in software maturity. (H3.1)	Q
	High M_{MOD} AND high M_{TEST} → high M_{OP} . A proper engineering process eases and shortens start-up, operation and maintenance. (H3.2)	Q
	Different release procedures for SW libraries due to on-site changes (H3.3)	Q
	Weaknesses in the tool chain support can be identified for selected aspects (H3.4).	Q
	Module libraries, release procedure, version management and change tracking are prerequisites for all ways of reuse (H3.5).	Q
	SW complexity → low M_{MOD} AND low M_{OP} . (H3.6)	Q
Does the detailed expert analysis deliver additional insights into the weaknesses of software maturity? (RQ4)	Expert analysis delivers additional insights (H4.1).	E
	Different approaches for code configuration can be assigned to different governance levels. (H4.2)	E
	(call graphs enable insight into control SW's structure. (H4.3)	E
	Decomposability, composability, understandability and protection enable high governance level → mature SW architecture & code graph → higher M_{MOD} . (H4.4)	Q&E

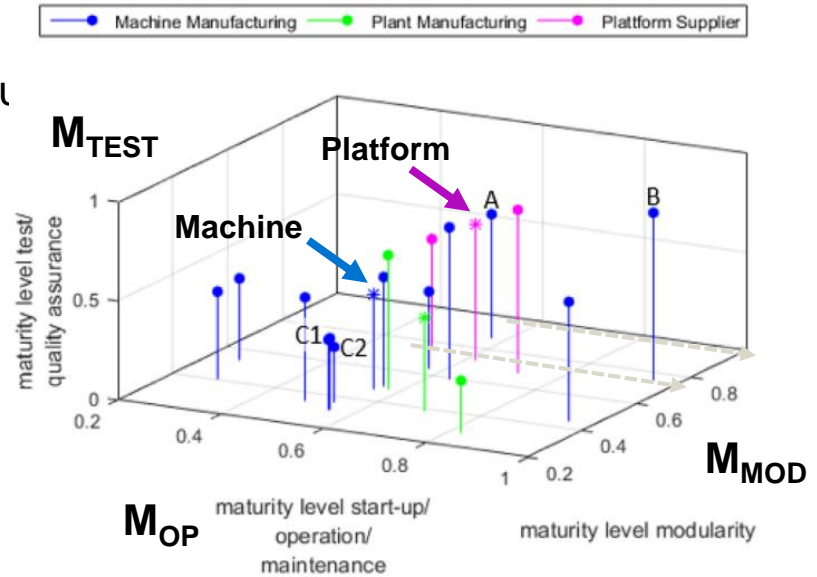
Validation of SWMAT4aPS (RQ1)

H1.1: The questionnaire delivers valid results in accordance with the detailed expert analysis of four selected companies. **True**

H1.2: Platform suppliers reach higher maturity values than machine suppliers than plant manufacturers. **Partially true**



<Overview of maturity levels of companies>



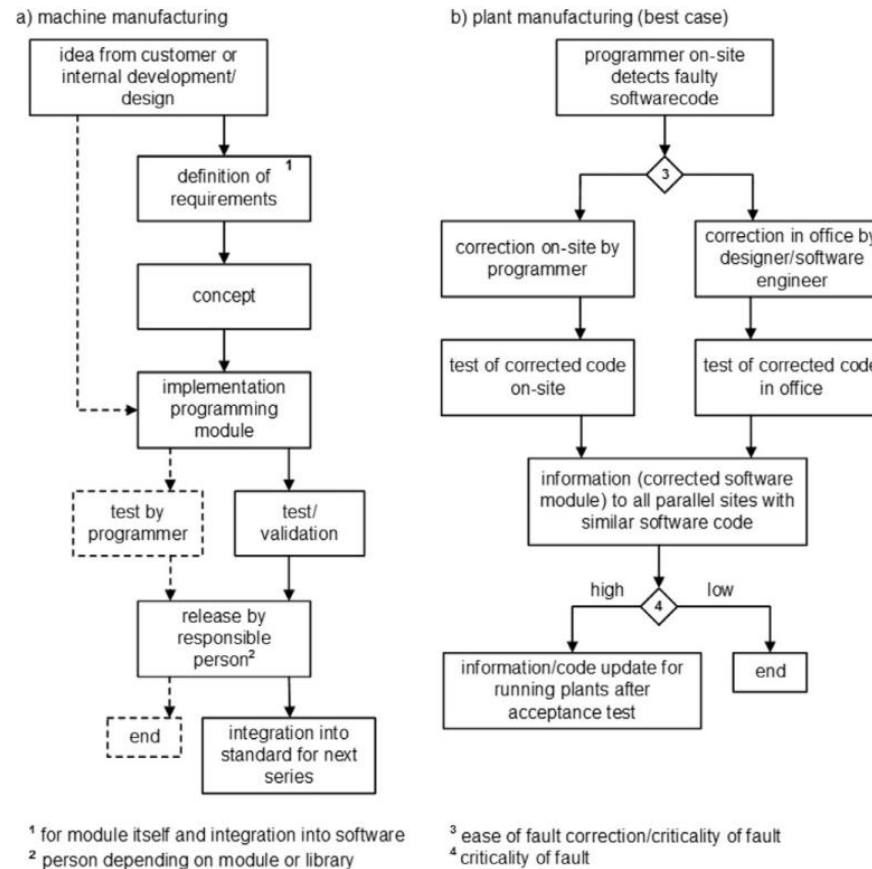
Interdependencies of maturity levels (* mean value)

	Case study A (8)	Case study B (14)	Case study C1 (5)	Case study C2 (6)	Machine manufacturing companies, mean
Maturity level					
Modularity	0.86	0.75	0.32	0.36	0.50
Test/Quality assurance	0.63	0.85	0.36	0.28	0.51
Start-up/Operation/Maintenance	0.58	0.95	0.55	0.54	0.58
Overall	0.77	0.80	0.36	0.37	0.52

Maturity levels of case studies compared to the machine manufacturing companies mean

Most significant weaknesses in software maturity phase do they occur, possible causes / reasons / prerequisites?

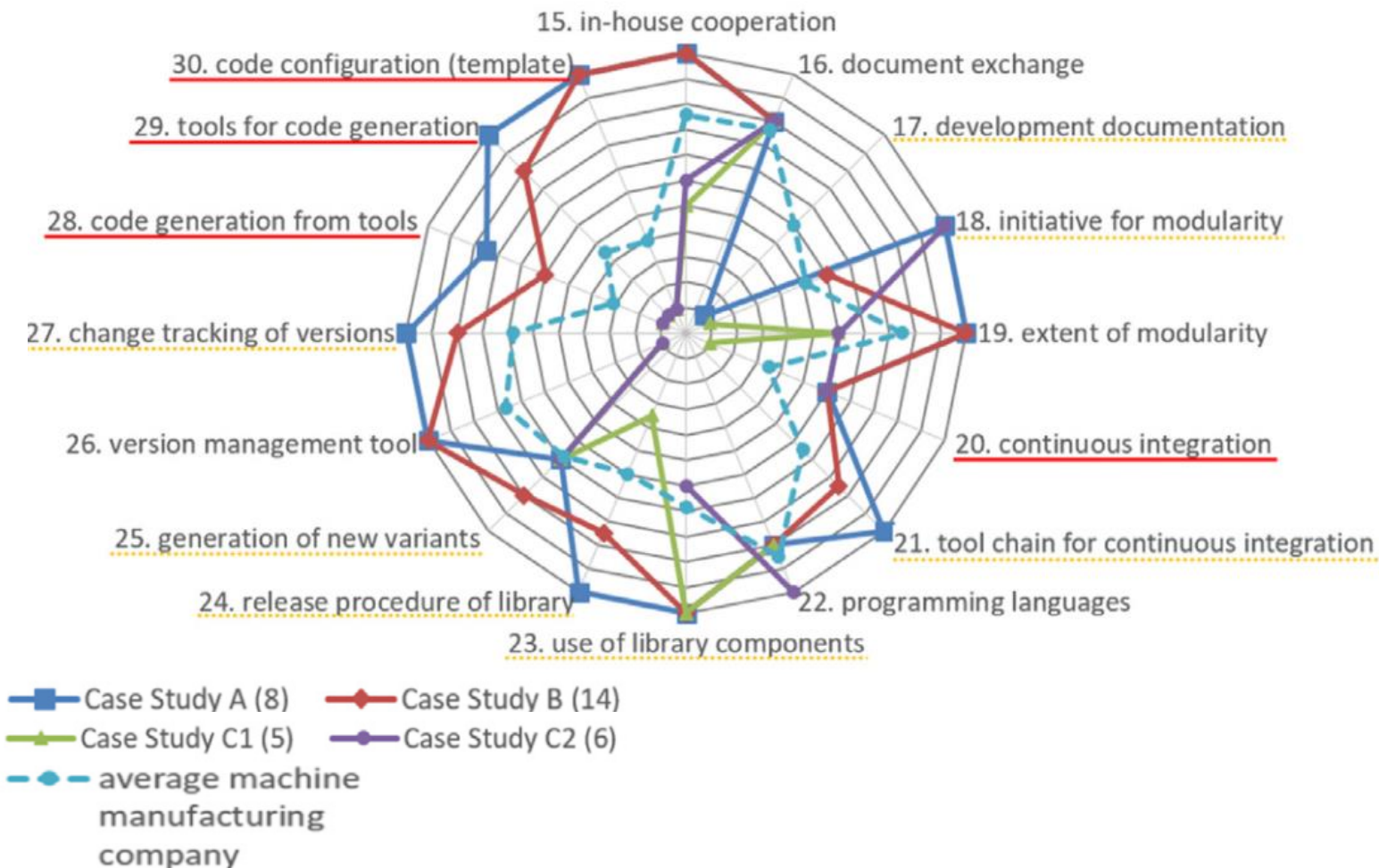
H3.3: Due to necessity of on-site changes in plant manufacturing, machine and plant manufacturers follow different release procedures for software libraries. **True**



Release procedure (workflow) of library element in machine (a) vs. plant manufacturing industry (b)

Most significant weaknesses in software maturity phase do they occur, possible causes / reasons / prerequisites?

H3.4: Weaknesses in the **tool chain support** (mean value machine manufacturing companies) can be identified for selected aspects, e.g. continuous integration, code generation or version management. True



H3.5: Appropriate module libraries, release procedure of library components, version management and change tracking are prerequisites for all ways of reuse. **True**

- Correlation analysis of an interaction variable's impact on two reuse indicators
- Additive interaction variable includes four questions from the questionnaire
 - use of library components
 - release procedure of these library components
 - used version management tool
 - change tracking of versions
- Considered ways of reuse: code generation and configuration

Table I. Correlations with Interaction Variable for Questionnaire Items # 23, # 24, # 26, # 27
Influencing Items # 28 and # 30

	interaction variable	(# 28)	(# 30)
interaction variable	1.000	.739**	.520*
(question # 28) code generation from tools	.739**	1.000	.846**
(question # 30) code configuration (templates)	.520*	.846**	1.000

** . Correlation is significant at the 0.01 level (2-tailed).

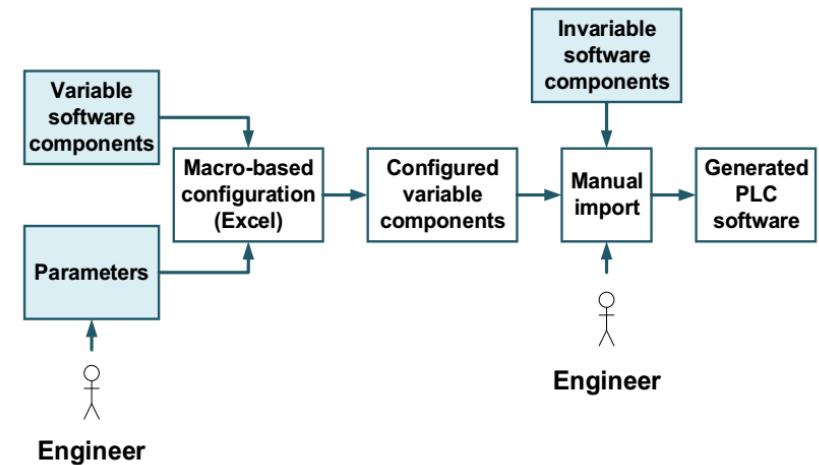
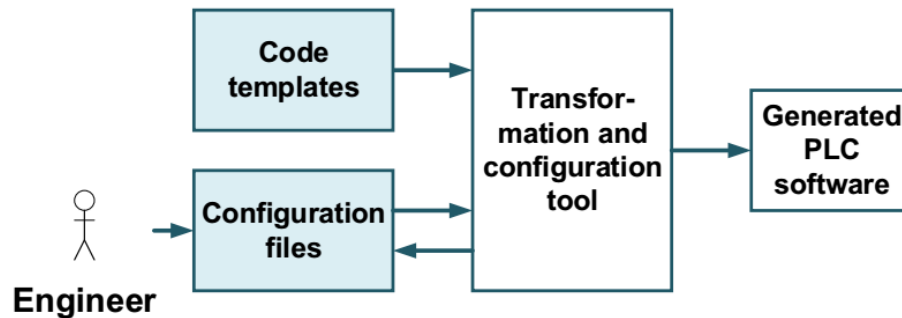
* . Correlation is significant at the 0.05 level (2-tailed).

Results of Expert Analysis

H4.2: Different approaches for code configuration exist in industry, that can be assigned to different governance levels.

True

Template-based configuration procedure in case study



Parameter-based configuration procedure in case study D



Prerequisites of Modularity Maturity M_{MOD}

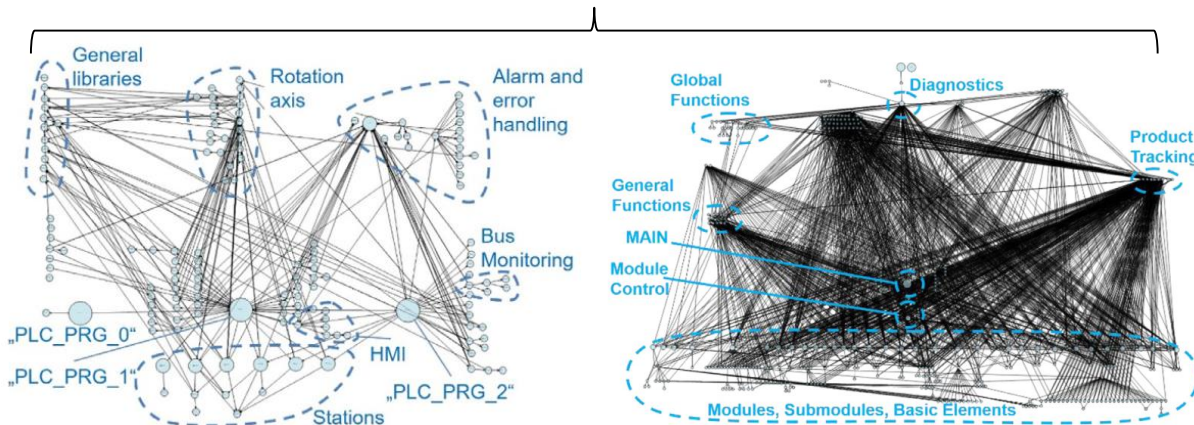
H4.4: The better the criteria decomposability, composability, understandability and protection are fulfilled, the higher the governance level the more mature the software architecture level as well as the code graph, and the higher the modularity maturity (M_{MOD}).

Partially true

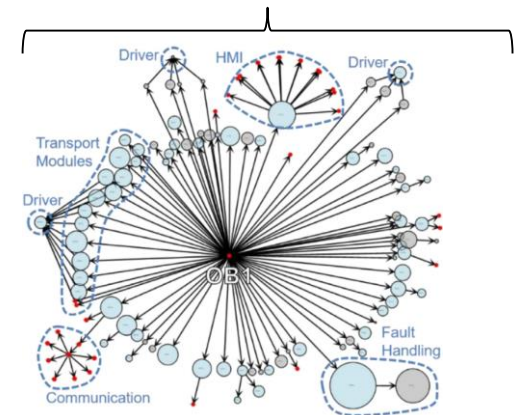
	Maturity Level	Case Study A (8)	Case Study B (14)	Case Study C1 (5)	Case Study C2 (6)	Case Study D
Q	M_{MOD}	0.86	0.75	0.32	0.36	-
E	Governance level	+ (L1 *)	+ (L3)	- (L0)	- (L0)	+ (L2)
	Decomposability	++	+	-	-	+
	Composability	+	++	+	+	++
	Understandability	++	+	+	+	+
	Protection	++	++	-	-	+
	Overall Scores from expert analysis (sum)	8	7	2	2	6

Call graphs generated for the analysis of case study A, B and C














Frequent cross connecting calls



Strict tree structure



Research questions and hypotheses results

Research Questions	Related Hypotheses	Proof	Results
Does the questionnaire deliver valid results to identify weaknesses in gaining software modularity of aPS? (RQ1)	Questionnaire delivers valid results (H1.1)	Q&E	
	Maturity level: Platform suppliers > Machine suppliers > Plant manufacturers (H1.2)	Q	
Do the three different sub-maturity levels deliver further insights compared to one general maturity level? (RQ2)	Maturity level differ among M_{MOD} , M_{TEST} , M_{OP} . (H2)	Q	
What are the most significant weaknesses in software maturity in aPS and in which phase do they occur and what are possible causes / reasons / prerequisites? (RQ3)	Universally low maturity levels arise in the different phases, indicating possible causes or prerequisites for weaknesses in software maturity. (H3.1)	Q	
	High M_{MOD} AND high M_{TEST} → high M_{OP} . A proper engineering process eases and shortens start-up, operation and maintenance. (H3.2)	Q	
	Different release procedures for SW libraries due to on-site changes (H3.3)	Q	
	Weaknesses in the tool chain support can be identified for selected aspects (H3.4).	Q	
	Module libraries, release procedure, version management and change tracking are prerequisites for all ways of reuse (H3.5).	Q	
	SW complexity → low M_{MOD} AND low M_{OP} . (H3.6)	Q	
Does the detailed expert analysis deliver additional insights into the weaknesses of software maturity? (RQ4)	Expert analysis delivers additional insights (H4.1).	E	
	Different approaches for code configuration can be assigned to different governance levels. (H4.2)	E	
	(call graphs enable insight into control SW's structure. (H4.3)	E	
	Decomposability, composability, understandability and protection enable high governance level → mature SW architecture & code graph → higher M_{MOD} . (H4.4)	Q&E	

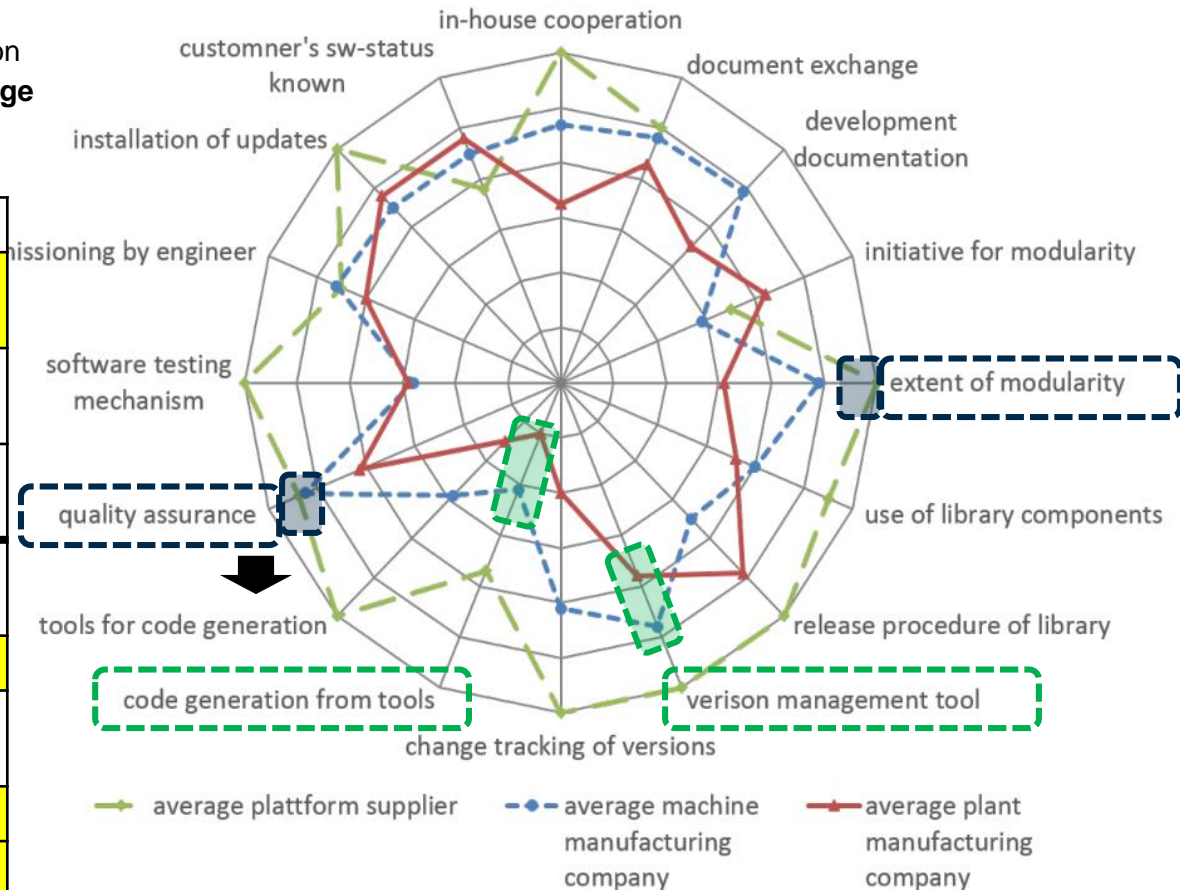
Current status of software development in industrial practice

SWMAT4aPS+

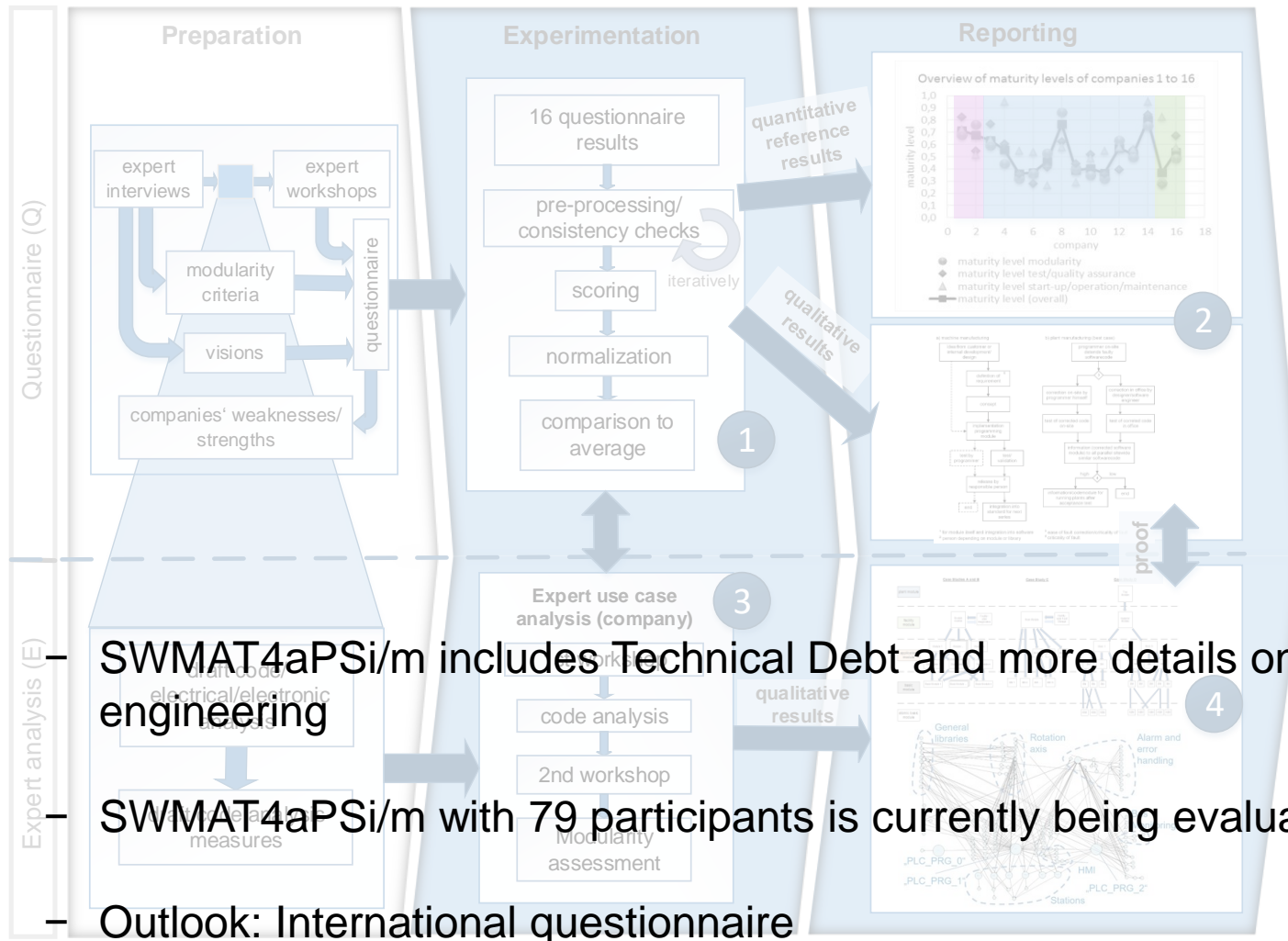
Evaluation of participants who answered the question
„How are your control software projects on average
made up?“

with > 50% machine-specific code:

↑	usage of IEC 61131-3 IL
↑	interfaces implemented as data exchange across global variables
↑	Team Foundation Server as Version Management Tool
↑	source code hand-over to the customer
↓	n-axis-positioning rated as critical application
↓	degree of modularization
↓	standards for the implementation of software projects
↓	amount of library blocks
↓	release process of library blocks
↓	disciplines using version management tool
↓	usage of a variant mng. tool
↓	usage of automated configuration based on templates
↓	usage of templates



SWMAT4aPS-Benchmark process to identify strengths and weaknesses in software modularity



Thank you for your attention!

Univ.-Prof. Dr.-Ing. Birgit Vogel-Heuser

Institute of Automation and Information Systems
Technical University of Munich

<http://www.ais.mw.tum.de>

vogel-heuser@tum.de

B. Vogel-Heuser, J. Fischer, S. Feldmann, S. Ulewicz and S. Rösch. "Modularity and Architecture of PLC-based Software for Automated Production Systems: An analysis in industrial companies", *Journal of Systems and Software (JSS)*, vol. 131, pp. 35-62, May 2017.



Questions of the first questionnaire

General descriptive information (not included in maturity calculation) besides #14 for complexity

- How many engineers and technicians are involved in the development projects?
- How many engineers and technicians work on-site?
- How many programmers are employed in the IT department?
- What number of start-up personnel is employed in the department?
- How many programmers are on-site (at customer's premises)?
- How many employees are involved in on-site start-up (at customer's premises)?
- How many programmers are there per application/machine?
- How many start-up employees are there per application/machine?
- Number of CPUs per machine/plant?
- Are these CPUs PC-based?
- What is the scale of the main applications created in your company?
- What is the scope of an application: lines of code?
- What is the scope of an application: number of components?
- Measure for complexity calculated as $0.5 \text{ (CPUs + programmer)}$

Sub items included in modularity maturity calculation (M_{MOD})

- How is the in-house cooperation arranged?
- Which documents are exchanged during a development project?
- How is the development project documented?
- Who started the initiative to use modularization?
- What is modularized?
- Is continuous integration used?
- If yes, what is the tool chain you use?
- What programming languages are used in your company?
- How often are library components used?
- Please briefly describe the release procedure of library components.
- How is the decision to form new variants made?
- Is your company using a tool for version management?
- How are changes for versions in your company tracked?
- How often is code generation from EPLAN or other engineering tools applied?
- Which tools/models are used for code generation in your company?
- Are projects configured automatically from libraries based on templates?

Questions of the first questionnaire

Sub items included in quality and testing maturity calculation (M_{TEST})

- Are there any quality gates before adding a new library component?
- What quality assurance measures are used in your company?
- What scenarios are tested or what requirements have to be met by the created tests?
- How is the software tested?
- Are simulations used for testing?

Sub items included in start-up, operation and maintenance maturity calculation (M_{OP})

- Is the start-up of the machine/plant done on-site by the designer/programmer?
- How is the delivery to the customer conducted?
- How are updates installed?

Does the service department know the current customer's software status on-site?

Manually evaluated questions from the questionnaire (not included in company profile lines because of insufficient answers)

- How long does a typical start-up process take?
- How are new elements added to libraries? – related additional text to #24
- Please describe the release procedure of a library element (from implementation/programming of the element to its library integration) – related additional text to #24
- By whom is the start-up of the machine/plant done on-site otherwise?
- On which level of the software do you use which programming language?
- Which are the most critical technical tasks to be automatically controlled in your applications?

- Software engineering for automated production systems (aPS) seems to be lagging behind classical software engineering
- The changes towards Industrie 4.0 require the software to be more maintainable over decades for thousands of machine and plant variants
- Reusability and variants & version manageability are key factors for efficient development for multi and frequent customization
- Manage and identify the view on software modularity
 - Industrial companies from automated production systems (machine and plant manufacturing)
- A **diagnosis tool** or **process** is needed for detecting **weaknesses** in **software engineering** or workflow characteristics