**Journal of Software Engineering and Applications, 2014, 7, 943-973**
Published Online October 2014 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2014.711084

Scientific Research

# Usability Experiments to Evaluate UML/SysML-Based Model Driven Software Engineering Notations for Logic Control in Manufacturing Automation

**Birgit Vogel-Heuser**

Institute of Automation and Information Systems, Technische Universität München, Munich, Germany
Email: vogel-heuser@ais.mw.tum.de

Received 26 August 2014; revised 20 September 2014; accepted 15 October 2014



## Abstract

**Many industrial companies and researchers are looking for more efficient model driven engineering approaches (MDE) in software engineering of manufacturing automation systems (MS) especially for logic control programming, but are uncertain about the applicability and effort needed to implement those approaches in comparison to classical Programmable Logic Controller (PLC) programming with IEC 61131-3. The paper summarizes results of usability experiments evaluating UML and SysML as software engineering notations for a MDE applied in the domain of manufacturing systems. Modeling MS needs to cover the domain specific characteristics, *i.e.* hybrid process, real time requirements and communication requirements. In addition the paper presents factors, constraint and practical experience for the development of further usability experiments. The paper gives examples of notational expressiveness and weaknesses of UML and SysML. The appendix delivers detailed master models, representing the correct best suited model, and evaluation schemes of the experiment, which is helpful if setting up own empirical experiments.**

## Keywords

**Manufacturing Automation Software, Programming Environment, Software Engineering, User Interface Human Factors**

## 1. Introduction

Today, manufacturing automation systems (MS) mostly consist of PLC-based control systems [1] programmed

**How to cite this paper:** Vogel-Heuser, B. (2014) Usability Experiments to Evaluate UML/SysML-Based Model Driven Software Engineering Notations for Logic Control in Manufacturing Automation. *Journal of Software Engineering and Applications*, **7**, 943-973. http://dx.doi.org/10.4236/jsea.2014.711084

in the languages of the IEC 61131-3 standard [2]. Since the proportion of system functionality that is realized by software is increasing [3], concepts for supporting automation engineers in handling software complexity are strongly required. Furthermore, spatially distributed MSs tend to result in a spatial distribution of software [4]-[6], *i.e.* networked automation systems (NAS). One key challenge of manufacturing automation companies in high wage countries is to increase efficiency, effectiveness and quality in design of software engineering for MS to shorten engineering and start-up time and ease maintenance. To reach this goal, model driven software engineering (MDE) including code generation is a promising systematic approach [1]. A variety of different modeling notations (general and domain specific ones) were developed and defined by academia and/or tool suppliers during the last decades to improve efficiency and effectiveness and to increase software quality in manufacturing automation, e.g. Vyatkin [7]-[9], Fantuzzi, *et al*. [10]-[12], Thramboulidis, *et al*. [3] [13] [14], Vogel-Heuser, *et al*. [15] [16] or Estévez, *et al*. [17] [18].

Many industrial companies and researchers are looking for more efficient MDE in software engineering of MS, but are uncertain about the applicability and effort including training and reengineering of component libraries and workflow needed to implement those approaches in their companies. In the last decade many new approaches, e.g. notations as UML and SysML and tools to support these approaches were developed and presented to industry, to improve software quality and efficiency as well as maintainability and evolution including more or less the typical requirements as real time, communication and partially hybrid process characteristics or networked automation systems. The challenge for manufacturing automation companies is to find the most appropriate MDE approach fitting to the companies' particular requirements, e.g. market requirements, customer relation (what to get from the customer and what to deliver), workflow requirements, qualification of personnel, budget and so forth. Consultants try to bridge this gap and support automation companies specifying their requirements and evaluating available approaches and/or tools. In a next step a beta-test may be conducted which is always short in time and delayed by higher prioritized "real" projects. Often poor usability of tools, missing features or missing modular and flexible training units lead to a rejection of a MDE approach [19], which would be applicable and beneficial if systematically selected and introduced. But the risk to spend too much time of experienced application engineers for evaluating a probably inappropriate notation and/or tool is often estimated as too high to allow a systematic and exhaustive evaluation.

To overcome this drawback and enable selecting the best notation which fits well to a company's needs, this paper provides a discussion of different ways for usability evaluation in Section 2 including usability evaluation methods and procedures that are well-established and have successfully been applied for decades, e.g. in the automotive domain [20] and in nuclear safety research [21]. In the following Section 3, aspects and rules to be considered when designing experiments are investigated in detail. These proposed aspects are exemplified by a variety of experiments conducted in the last decade on usability of software engineering in MS presented in (Section 4). The paper closes with a summary and discussion in Section 5 and 6, respectively. The appendix gives three examples of the evaluation of subjects' models. Appendix A shows subjects' models from the first experiments highlighting the difficulties in understanding class as a structural mechanism to foster modularity and reuse using UML 1.4. Appendix B shows an example of subject's UML model (Evaluation scheme E1) compared to a master model also highlighting the calculation of the later used complexity measure WMC. Appendix C explains the different abstraction mechanism between subjects' solution and master model.

## 2. Methods in Usability Evaluation

Up to now the standard procedure in industry when testing the applicability of different modeling notations in automation technology is a consultation of experts (application engineers) working in the industrial sector of interest [22]. Alternatively—or as a supplement to that practice—end users are questioned, e.g. the programmers of PLCs [23] having in fact the same drawbacks. From a methodological point of view, the consultation of experts can either be individually (*i.e.* interviews) or in an interactive group setting (*i.e.* focus group or case studies in workshops). Both approaches measure a subjective assessment of the respondents, *i.e.* their attitudes, opinions and knowledge can be gained—not their objective behavior. Focus groups as well as case studies operate with such small numbers of experts to start exploring usability issues. Focus groups require a moderator asking questions and moderating the discussion. In case studies an observer is required monitoring and observing the experts actions. Gained insights can serve as an inspiration for further, more detailed, in-depth studies with more explicit hypotheses. Consequently, the consultation of experts or end users is not the only appropriate way to

investigate the applicability of a notation but to identify which notations might be relevant and/or to gain first qualitative results on weaknesses and drawbacks. A major advantage of focus groups compared to individual interviews is a more natural atmosphere that ideally leads to increased talkativeness and openness of the participants [24]. An example of a focus group evaluation is given in [25]. In contrast, individual interviews are easier to perform (especially with busy industrial experts), as the persons can be questioned at different times and different places, *i.e.* no common appointment has to be found. Mostly, a very limited number of experts is available for evaluation, and consequently, a representative quantitative opinion cannot be gathered. However, an evaluation by experts, as for example described in [26], can act as a beta-test of a prototypical tool indicating the opinion trend. The same challenge occurs with case studies testing industrial experts and/or researchers as e.g. described in [27] and [28]. In both studies, a significant small group of participants were observed during courses on programming in accordance to the IEC 61499 compared to IEC 61131-3 standard and an Object Oriented approach. Conclusions based on observed aspects can only serve as indicators. Consequently, experiments with a higher number of subjects are required to gain objective, detailed and quantitative results for specific research questions. The in depth investigation of research questions like the applicability of novel modeling notations or programming paradigms presuppose that test persons get familiar with the novel notation or paradigm. The introduction of the required knowledge is very time-consuming and consequently not applicable for a large number of experts, *i.e.* minimum 15 - 20 per compared notation, required for significant, objective results. Furthermore, the comparability of results of different experts with different previous knowledge is very difficult to handle. Thus, such research questions (e.g. the usefulness of a certain new notation) can only be quantified with laboratory experiments, where comparable testing conditions can be created and where, due to the sheer number of tested subjects (plus respective grouping), their individual characteristics and preferences can be considered. Such relevant testing conditions and constraints which have to be considered during experimental design are discussed in detail in the following section. Furthermore, their relationship to usability and how to quantify experimental results in order to deduce quantifiable usability results is presented.

## 3. Factors for Experimental Design in MS

Patig [29] [30] and Gemino and Wand [31] present results of various experiments on the usability of modeling notations in requirement engineering and classical software engineering introducing relevant factors for successful experimental design. Following an experimental approach the experimental design needs to be fixed ad first, choosing the constraints of the experiment, the so called affecting variables (**Figure 1**), *i.e.* the process to be controlled, the engineering task as well as the duration of the whole experiment depending on the temporary availability of the subjects and the subjects themselves. These affecting variables are described in Section 3.1. Hereafter we discuss the affected variables or usability requirements for experimental design in Section 3.2, showing the measures to evaluate the different notational approaches. Thereby, we draw both on [29] [30] and [31] as well as our earlier reflections [32]. On that basis, we adapted and enlarged the extent of variables with focus on the specific requirements in design and maintenance of MS.

### 3.1. Overview of Affecting Variables

According to the best of the authors' knowledge, the main three influencing factors seem to be (1) the task, (2) the preceding training and (3) the tested subjects (**Figure 1**). Because subjects' attendance for training and experiment is a prerequisite, duration is not only a training constraint but also a constraint for the whole experiment including the performances of the engineering task. For experiments with students or apprentices two days is most promising from an organizational point of view.

#### 3.1.1. Task

To describe the requirements from the technical process point of view the automation task is introduced and separated from the engineering task performed by an application engineer using notations and tools.

**1) Automation Task**

When designing an experiment, e.g. in order to evaluate a given notation for an enterprise, the task to be automated, *i.e.* the automation task, and its characteristics has to be clarified. The characteristics should be selected similar to characteristics of the real automation tasks in the company's daily business. Due to the complexity of real industrial automation tasks, the main challenge in the experimental set-up is to develop an appropriate

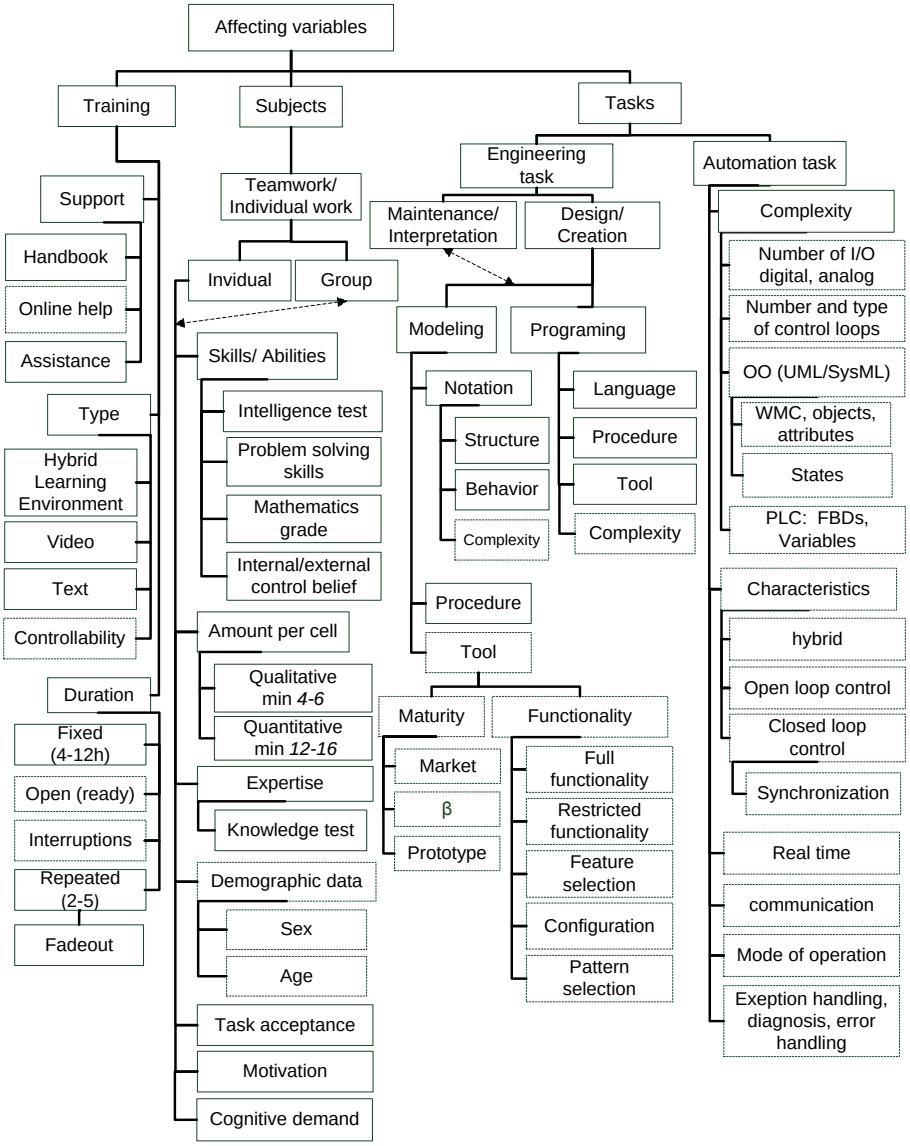


**Figure 1.** Affecting variables (dashed lines show dependencies between group and substructure and maintenance/interpretation and substructure).

automation task with adjusted complexity the subjects can cope within the limited time frame of the experiment and the required training.

As measure for complexity of an automation task, the number of inputs and outputs (I/O) is a familiar metric in cost estimation during preparation of an offer distinguishing between analog (closed loop) and digital (logic design) inputs and estimating the complexity of the closed loop control with a factor (see column "automation task"). However, it is far too simple to reduce the complexity of an automation tasks to the number of I/Os because the automation task itself involves even more complexity. To derive the complexity of the automation task, it is necessary to describe the automation task itself with a notation. Therefore, then notational influence may lead to different complexity measure for the same control task. In the following measures for control task complexity correlated to the chosen notation are introduced.

Lukas *et al.* [33] introduce different complexity measure, *i.e.* the size (number of operation and number of state variables), modularity and interconnectedness and applied those in logic control. Frey and Litz [34] introduced complexity metrics for Petrinet using besides others an adapted McCabe metric. Venkatesh *et al.* [35] proposed to count the number of elements required to represent a certain program in order to measure its com-

plexity as well as Lee and Hsu [36], who converted the programs in question into Boolean expressions by using if-then transformations and, afterwards, rated the programs' complexity by comparing the calculated values.

For applying an object-oriented notion for describing the automation task, Chidamber and Kemerer developed a set of metrics of OO design [37], e.g. weighted methods per class (WMC) which is a measure for class complexity used in this paper (see column control complexity in **Table 1**): in order to calculate the WMC of a program, the cyclomatic complexity measure of each method is summed up for all classes, cf. [38]. When applying classical IEC 61131-3 code to describe an automation task, the number of FBDs and their instances in case of IEC 61131-3 are a familiar measure [39].

Besides those metrics which rely on the description of the automation task for deriving its complexity, also its characteristics like the type of control loop, *i.e.* logic control, closed loop control (with synchronization) or a technical process requiring both, called hybrid in the following can be taken into account. Furthermore requirements on automation systems, which have to be fulfilled are introduced: real time requirements, communication requirements between different controllers, and networked automation systems (NAS) as a class of systems with real time and communications requirements because the code and functionality is distributed onto different automation devices.

Besides the regular machine control function, diagnosis, exception handling, visualization and other functionality need to be developed. "In fact, industry folklore suggests that approximately 90% of the overall control logic is used for exception handling" [40]. During the last years, the authors' team analyzed real PLC code from several MS companies and realized that 6% - 10% of the lines of code are dealing with diagnosis and safety [41]. As a consequence, the mode of operation (EN 13128 [42]: auto, hand, manual etc.) as well as diagnosis including error handling (according to [38] and [41]) are typical automation tasks which provide another possible classification.

**2) Engineering Task**

The category engineering task describes the task to be solved by the human in the experiment, e.g. model (UML, SysML) or program (IEC Code) to control the automation task. Gemino und Wand [31] distinguish two types of human tasks in automation and control:

- design and creation versus
- understanding and analysis (being typical for maintenance tasks in MS).

For both types of tasks, it must be between modeling and programming (dashed lines in **Figure 1**). Moreover, maturity and functionality of tool support have to be considered as influencing factors. As tool classification three maturity levels and four different functionality types are proposed. Because the comparison of different notations in MDE is focused in this paper the modeling notation as such and its measures should be discussed in more detail.

The notation needs to fulfill the requirements given by the automation task, the life cycle model and the engineering task allowing modeling structure and behavior. Different modeling notations also possess different complexities. Calculation schemes are proposed by e.g. Recker *et al*. 2009 [42] and Rossi and Brinkkemper [43]. The expressiveness of this measure is limited to the complexity of the pure notation calculated on the number of its elements.

$$C(M) = \sqrt{O^2 + R^2 + P^2} \tag{1}$$

With O being the number of object types, R describing the relationship types and P the property types of a method, C(M) is the resulting complexity of a heterogeneous modeling language.

Schalles [44] compared UML activity diagrams for behavioral modeling and UML class diagrams for structural modeling, taking into account the high complexity differences (**Table 2**).

To allow the comparison of notations' complexity, the complexity of the later discussed notations is calculated, too. The complexity of the UML class diagram (**Table 2**) is nearly double compared to IEC 61131-3, SysML-AT (see also [45]) and CFC showing that between the typical MS notations the difference is very small.

### 3.1.2. Subjects

Subjects' qualification and experience is essential for the outcome of the experiment. Sierla *et al*. [28] and Hajarnarvis *et al*. [46] included industrial experts. Sierla introduced teamwork with clearly separated tasks similar to a real project team in industry.

Many papers on programmers' competencies in modeling and informatics systems application are related to

**Table 1.** Overview of the experiments.

| | Criteria | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Compared MDE Approaches/ Notations | Engineering Support | Tool Support | Control Task | Training Type (Duration h) | Subjects' Qualification; Min. Number per Cell | Design Overall Evaluation | Results | Results Subjective Questionnaire | Results for Further Notation/Tool Development |
| O.1 | IEC 61131 LL, Petrinet, SIPN, mFSM Lucas and Tilbury [67], Lucas [68] | N | LL (from manufacturer of test bed) mFSM (paper and pencil) | Flexible manufacturing test bed Logic Control 30DI/O (n.A.) | n.A. | LL professional, PN, SIPN, mFSM by students | | Estimation of development times based on low level user operations, and observation of engineers performing their task LL-405 min PN-1110 min Mfsm-1500 min | n.A. | Schema for estimation of time based on low level user operations |
| O.2 | IEC 61131 LL, Petrinet, SIPN, mFSM Lucas and Tilbury [33] | N | LL (from manufacturer of test bed) mFSM (paper and pencil) | Flexible manufacturing test bed Logic Control, DI/O 30 4 scenarios (n.A.) | n.A. | LL professional, PN, SIPN, mFSM by students | | Number of operations, state variable, modularity, interconnectness | n.A. | LL small but very interconnected, mFSM most modular, although largest |
| O.3 | IEC 61499 vs IEC 61131-3 plus design pattern repo-sitory Strömman *et al.* [27] | N, P | Market tool, repository | Lifter unit Logic Control (-) | Presentation of design pattern, repository and FBDs | 20 Professional automation designers and researchers | | Models analyzed, and written feedback 3 | Tape recorded interviews in companies as preparation each 60 min. | Event driven model experience relevant, guidelines for reusable software, environment that fosters collaboration and exchange of information |
| O.4 | IEC 61499, IEC 61131, OO Sierla *et al.* [28] | N,P | Market tool, adapted 61499 framework, repository | (hyb) Per notation 1 week course for training incl. experiment All approaches different sequence | Set of domain-specific design principles | 7 experts different background team work | | Observations, field notes | Interviews after course | Further guidelines, design patterns and tool support. |
| O.5 | LL, sequence Hajanarvis [46] | N | EC, RS Logix 5000, RS View, market tool | Turn motors and valves Logic, sequence change including diagnosis and HMI LL (1); sequence (1) All approaches different sequence | Briefing sheets, questions, tool | 45 skilled and 18 unskilled | | Performance, Right first time rates, completion time, observation, sid effects of prior experience; Step logic and EC best results, EC superior for untrained Observations helped to identify problems the participants prevented from solving the task correctly | Self assed skill test | Observations revealed deficiencies in SFC programming interface |
| E1 | UML 1.4 CD vs. ICL vs. PLC S7 UML 1.4 not restricted ICL truth table, SFC, idioms Friedrich *et al.* [69]-[71] | N | ~ S7 IL, LL, FBD | p&p structure/behavior (hyb) DI 22, AI 1 DO 13, AO 3, WMC = 45 Modeling (1.5); PLC programming (1.5) | L (1.5), E (0.75) ppt-slides | BSc 5 (2 in each group) Students in Practice 3 Technicians 3 | - | Behavior (19.31 steps from 32 realized $p < 0.01$) BSc 20.26, StiP 15.44, Techn. 12.78 ($p = 0.02$) Error rate: 43.84% ($p < 0.01$) BSc 53.36%, StiP 40.26% Techn. 13.64% ($p < 0.01$) | UML applicability structure 1.33, behavior 3.07 (1 not applicable-5 applicable) ICL (s 1.71; b 3.60) error: novices 48.7% experts 21.6% $p = 0.02$ | Reduced number of UML diagrams, tool and modeling procedure needed |
| E2 | UML PA (Instance structure D) vs. UML 2.0 (CD, Component D. Composite Struct. D, Deployment D) Focus: configuration, deployment Katzke *et al.* [72] [73] | N, M, P | Prot. | PE p&p structure (hyb, comm) ME: Continuous hydr. Press; switch between pressure and distance control, selection of pattern AI 60, AO 30 WMC = 3, UML (3.81); UML-PA (2.27) | PE L(3)/E(3) ME: L(3)/ E(-) 1 rep. online-help | BSc Mech. & CS 4th sem. 6 | Q+ Restricted tool | PE: 20 min breaks, Too complex (difficult); ME: time to interpret UML 2.0 $\bar{\tau}_{20}$ = 22:52, SD = 3:14 UML-PA $\bar{\tau}_{PA}$ = 13:40, SD = 4:42 ($p \leq 0.05$) Self-assessment/steps UML 2.0 1.78; UML-PA 3.93 ($p \leq 0.10$) Errors [42] | ~ | • Tool flexibility needed<br>• Change between diagram requires additional time<br>• Communication and deployment support successful |
| E3 | UML Embedded SC vs. SFC, focus: error handling and consistency check of sensor states Witsch *et al.* [75] [77] | N | Prot. CG | Cylinder (p&p) structure/behavior error handling: interception of time and erroneous sensor values; movement of 1 cylinder with two sensors DI/O 2(2) | L (1), E (0.75) per notation | BSc Mech. & CS 4th sem. 15 | + Small subtask | SC $\bar{x}$ 1.92 p/min ($\bar{x}$ 1.98 p/min with comp. states; $\bar{x}$ 1.81 without comp. states); SFC $\bar{x}$ 1.41 p/min | Subjects using comp. states higher prog. Experience: $r = 0.479$, $p = 0.033$ | Error handling beneficial using UML SC with composite states Tool: side effects: demand for auto placing |

**Continued**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| E4a | | | | | | BSc 2.Sem 41CD/21SC | - | no significant differences in terms of modeling order | | |
| E4b | UML CD/SC vs. SC/CD order Vogel-Heuser, Seidel [78] | N | ~ | (p&p) structure/ behavior DI/O 13 (DI 9; DO 4), WMC = 11 small group (0.42) | L (3.2) E (0.42) | BSc 2.Sem 70CD/32SC | + | "behavior first" group: $\bar{x}$ 23.4 p (SD =10.326; SE = 0.982), Structure first group $\bar{x}$ 18.4 p (SD = 8.220; SE = 1.825) out of 46 p max. s~: quality structural model bad: "structure first" $\bar{x}$ 12.26 p "behavior first" group $\bar{x}$ 11.14 p, structural quality of model is independent from modeling order | Post experiment interview and questionnaire to reveal challenges and reasons for mistakes ~ | Forcing subjects to follow a specific modeling order is not helpful to improve structural models; subjects had problems to create suitable classes from similar objects |
| E5 | UML Embedded CD/SC vs. IEC FBD Vogel-Heuser *et al.* [63] | N | ß CG | (p&p) *with reuse of buffer*, structure/ behavior DI/O 38 (DI 24; DO 14), WMC = 44 (UML) FBD 64 variables (2.2 + 0.92 quest.) | HLE and HB Fade out (Rep) (6.42) | 18 Apprentices 1./2. year | ++Strong correlation with abilities | Behavior ($p = 0.22$) and structure ($p = 0.15$) not significant; UML: thinking breaks Relatives to abilities (**Table 1**, 51): grade mathematics: ($r = 0.234$, $p = 0.055$); grade automation: ($r = 0.327$, $p < 0.005$) grade mechatronics: ($r = 0.327$, $p < 0.005$); cognitive demand: ($r = -0.255$, $p < 0.05$); previous knowledge: ($r = 0.485$, $p < 0.001$) | Frustration levels of UML group signif. higher than FBS group ($p = 0.02$), clearness of FBD signif. higher than of UML ($p = 0.017$) ease of use for behavior programming UML signif. lower than FBD ($p = 0.012$), Subjective quality estimation of UML group matches the factual quality signif. better than the FBD group ($p = 0.025$) | Difficulties with abstraction and building correct classes, e.g. cylinder, relationship CD and SC |
| E6 | SysML-AT Parameter Diagram (PD) vs. CFC vs. IEC 61131-3 ST; Frank, Schütz/Obermeier, Schütz [25] [26] | N | ~ | Physical laws, structure, Maintenance task, analysis, understanding, interpret 4 - 5 sub-blocks, 7 - 8 variables, WMC = 5 - 6 (0.167 per notation + 0.083 quest.) | L (0.167) E (0.084) per notation | 6 BSc. (mech. eng.) All notations different sequence | Q | Correct solutions PD 68.2% CFC 63.4% ST 64.3% | Answers correlated with objective results, learning effects across notations | Adequate method for requirement analysis and architecture design was missing |
| E7 | SysML-AT vs. IEC 61131-3Notation + method Frank *et al.* [79] [80] | N, M | Prot. | NAS Coking plant structure, hyb, communication, RT: Material sorting, belt synchronization, level control; DO 3, AI 8; AO 4 WMC = 18, (a) 2,3; b) 2,6; c) 2,6); all mean times | L (1.5) E (1.5) | 15 BSc (mech.eng) | ++ | CFC $\bar{x}$ 93.5 (182) points NM $\bar{x}$ 123.1 (182) points ($p < 0.001$) | From NM to NMCP: mental demand and workload: ↓, fear of failure: ↑, fatalistic externality: ↑, NMC worst values: suitability for task and individualization (see **Table 3**) | Compromise between support and complexity. Difficulties with module abstraction |
| | a) plus characteristics Frank *et al.* [81] | N, M, C | | | L (2) E (2) | 5 | Q+ | NM $\bar{x}$ 97.20 (182) NMC $\bar{x}$ 116.6 (182) | | |
| | b) plus pattern Eckert *et al.* [82] | N, M, C, P | | | L (2.5) E (2.5) | 5 | Q- | NMC $\bar{x}$ 116.6 (182) NMCP $\bar{x}$ 119.75 (182) | | |

Legend: N—notation; Prot—prototype; p&p—pick and place; L—Lecture; Q—only qualitative results; M—method; HB—handbook; (p&p)—part of p&p; E—exercise; er—error rate; ß—Version; A—analogue; I/O—input/output; Rep—repetition; CG—code-gen IEC 61131-3; P—pattern; D—digital; ~—only pencil & paper; C—characteristics; ME/PE—main/pre experiment; HLE—hybrid learning environment.

**Table 2.** Complexity values for heterogeneous modeling languages.

| Table Head | O | R | P | C(M) |
|---|---|---|---|---|
| UML Class Diagrams [44] | 7 | 18 | 18 | 26.40 |
| UML State Diagrams [44] | 10 | 4 | 11 | 15.39 |
| IEC 61131-3 FBD[1] | 5 | 9 | 8 | 13.04 |
| SysML-AT PAR[2] | 8 | 2 | 9 | 12.21 |
| CFC[3] | 7 | 8 | 7 | 12.73 |
| (SysML-AT a)[4] [45] | 8 | 7 | 7 | 12.73 |

[1]The values listed in the table are gained by counting the following elements: O (function, function block, variable, (read, write), constant) = 5; R (direct connection 1 to 1, direct connection 1 to n, set, reset, rising edge, falling edge, negated connection, branch instrution, return) = 9; P (FB name, FB type, FC type, input name, output name, variable name, variable type, network number) = 8; [2]O (Block, ConstraintBlock, attribute (In, Out, Local), PortIn, PortOut, Property) = 8; R (BindingConnection, FunctionalDependency; direct connection 1 to 1) = 2; P (BlockName, BlockEntityName, ConstraintName, AttributeName, AttributDataType, PortpertyName, PropertyDataType, PortName, PortDataType) = 9; [3]O (Function, Function Block, Input, Output, Comment, Composition, Selector) = 7; R (direct connection 1 to 1, direct connection 1 to n, Set, Reset, negated connection, branch instruction, connection marker based, return) = 8; P (FB name, FB type, FC type, input name, output name, jump mark name, execution sequence number) = 7; [4]O (Input Ports, Output Ports, Protocol Ports, Knots, Block, Functions, FBs, requirements) = 8; R (1 to 1 concrete, 1 to n concrete, n to 1 concrete, 1 to 1 discrete, 1 to n discrete, n to 1 discrete, enclosing) = 7; P (FB name, Function name, Requirement type, Requirement name, Knot name, Knot type, Blockname) = 7.

competence models, e.g. [47]-[49]. Usually, competencies in this context are understood as abilities, skills, and knowledge—a perspective which is still prominent in most Anglo-American research on competencies. An example is provided by Curtis [50], who proposed that programming results depend on individual personal factors and mental abilities. His model covers intellectual aptitudes, the knowledge base, cognitive styles, the motivational structure, personality characteristics, and behavioral characteristics. Although Curtis did not empirically test his model, the factors show at least facial validity.

Other approaches for gaining insights into competencies required for different programming approaches or skills analyze interviews from experts in the questioned domain [51] or evaluate programmers' behavior when performing certain tasks, e.g. programming tasks [52] or debugging tasks [49].

In prior experiments regarding process operators we tried to measure workload using a secondary task (e.g. communication and documentation) as, among others, proposed by Wickens [53], but could not detect significant effects [54].

Because for statistical significance a minimum of 15 subjects per different notation or approach, are necessary [55], it is obviously impossible to conduct experiments with such a high number of experienced application engineers under same conditions.

In ergonomics it is usual to conduct usability experiments in engineering design with students of mechatronics or mechanical engineering because they are future application engineers. Maintenance tasks are performed in Germany mostly by skilled workers, therefore apprentices and technicians are appropriate subjects.

#### 3.1.3. Training

Kim [56] suggests that the complexity of the design task in the training period and the test period should be increased stepwise.

As Kim and Lerch [56], Ruocco [52] and others (*i.e.* [57]-[62]) point out, repetition is essential for learning object orientation. Ruocco decided for a stepped approach when teaching UML throughout a computer science program. He found that the application of UML during a database course and the incorporation of use case diagrams, sequence diagrams and activity diagrams led to a richer and deeper exposure to UML.

In longitudinal studies, too, teaching beginners or freshmen in computer science or object orientation mostly goes along with repeated training [52] [57]-[62]. For training purposes, the pedagogic methods of repetition and fade-out, *i.e.* decreasing support by trainer from training step to training step, seem to be suitable (see for more details 63). In case of IEC 611131-3 or UML prior knowledge (see expertise in section subjects) acts as a disturbing factor if not equally distributed over subjects. Therefore training is necessary to adapt prior knowledge before conducting the experimental task. Time between training and experiment should be nearly the same for all subjects to avoid time depending differences in results.

### 3.2. Overview of Affected Variables (Usability Requirements)

In order to perform usability experiments, it is necessary to clarify how usability can be measured (affected variables) and which metrics can be used to make quantifiable statements about the advantageousness of the object of research (given by affecting variables).

The standard ISO 9241-11 [64] includes studies regarding use efficiency and the satisfaction of the users suggesting the measurement of product's usability in its context of use. According to this standard concerning usability requirements, the main affected variables, are (1) effectiveness, (2) efficiency and (3) user acceptance (cp. **Figure 2**).

Effectiveness, *i.e.* the quality of the result depends on the completeness and correctness of an engineered solution. Efficiency is the effectiveness in relation to the effort to engineer a solution. Both measures analyze the models developed by the subjects during the experiment compared with a so called master model [64].

User satisfaction is the scale to which users are free of interference and their attitude to use a product [64]. Furthermore, the standard ISO 9241-110 contains dialogue principles for human-computer interaction as attributes to usability requirements. Those principles are suitability for task, for learning, for individualization, conformity with user expectations, self-descriptiveness as well as controllability and error tolerance.

#### 3.2.1. Effectiveness-Quality of the Resulting Model, Program

In (ISO 9241-11:1998) [64] it is proposed to determine the effectiveness by linking the grade of completeness

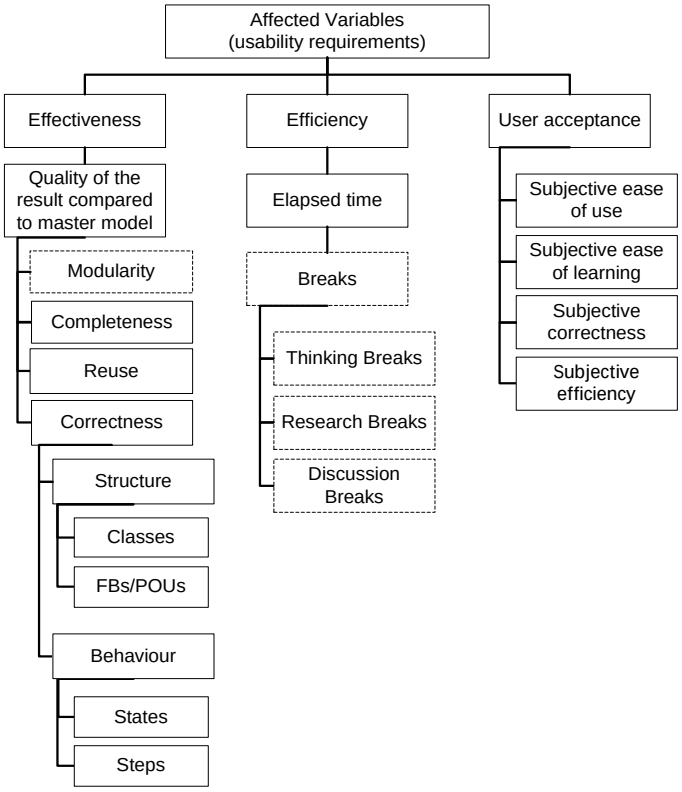


**Figure 2.** Usability requirements, *i.e.* variables in experiments.

with the grade of correctness. Bevan (1995) [65] defined effectiveness in a different way as a product of quantity and quality:

$$\text{Grade of completeness} = \frac{\sum\left(N_{task} + E_{task}\right)}{\sum\left(N_{goals} + E_{goals}\right)},\ D = \{0;1\},\ W = [0,1] \tag{2}$$

$$\text{Grade of correctness} = \frac{\sum\left(N_{goals} + E_{goals} - R\right)}{\sum\left(N_{goals} + E_{goals}\right)},\ D = \{0;1\},\ W = [0,1] \tag{3}$$

(With N being the number of nodes, E the number of edges and R the number of errors with index task being the model of the subject and goal being the model of the experts taken as the correct solution in the master model).

Schalles compared only UML structure and behavior diagrams for business process modeling on an abstract level [44]. Applying his approach on MS would fall short. Using different notations modeled solutions are different two regarding number of nodes and edges of the correct master model (see Appendix B for an example). As Strömmann [27] already realized often different correct solutions are created by different subjects, which should be evaluated equally good. Moreover, equality of nodes in a class diagram (abstract representation) and nodes and edges in an activity diagram (low level, object related) are rated equally by Schalles. In MS, the correctness of structural model elements (e.g. classes or function blocks) should be measured differently than the correctness of low level behavioral model elements (e.g. correct steps or transitions from one state to another one [63]) due to the different degree of difficulty and ease of change in case of an error. Because IEC 61131-3 FBD is a language without nodes and edges Schalles approach is not feasible.

In accordance with Annett's proposal of an Hierarchical Task Analysis (HTA) [66] the proposed evaluation scheme counts referring to a top down approach all detailed elements modeled by the subject, allowing similar scores e.g. for a combination of class diagrams and created objects in comparison to structure elements in FBD.

In order to assess the effectiveness of notations the grade of task completion was used instead of measuring the grade of completeness and the grade of correctness separately before multiplying them. A task is completed,

if its solution is logically and syntactically correct. As only correct task solutions, *i.e.* model elements are counted, it is not necessary to take additional errors into account. This results in the following term:

$$\text{Effectiveness (F)} = \text{Grade of task completion} = \frac{\sum(T_{solved})}{\sum(T_{goal})},\ D = [0,1],\ W = [0,1] \tag{4}$$

(With N being the number of tasks).

This approach has three main advantages:

First, it can be applied equally for all kinds of notations as long as the given task can be completed with it and there is no restriction as in [65], that only models with fewer nodes and edges than the master model can be evaluated.

Second, the task analysis can be used to select relevant tasks for evaluation and reduce the number of tasks to review, which then can be checked for logical and syntactical correctness, resulting in highly accurate data on task completion.

Third, no negative points for errors have to be used to calculate correctness, as this could manipulate results in an undesired way, e.g. errors lead to negative overall efficiency or errors in one part of the model nullify correct solution of others.

A fully automated analysis of the student's model compared to the master model is nearly impossible. The difficulty in rating the results of an experiment is comparable to a fair grading of exams by distributing points for correct solutions, but more sophisticated. As a consequence, the development of the correction guidelines for the manual evaluation is required. Points are given by two evaluators independently with a necessary interrater reliability of at least 65%.

### 3.2.2. Efficiency

Regarding industrial application the time needed to engineer an automation task correctly is one of the most important measures, defined as efficiency in usability evaluation. The efficiency of a notation can be calculated through a combination of effectiveness and time required for execution of a task (ISO 9241-11:1998).

Accoring to Schalles [44], efficiency is defined as:

$$\text{Efficiency (G)} = \frac{F}{T} \tag{5}$$

(With effectiveness F and time T).

If time is a freely selectable variable, this calculation basically provides a good comparison between notations in terms of effect per time. For the experimental design time may be fixed and restricted to a calculated amount of time with GOMS [67] or pre-experiments similar to an exam or left open for subject's decision, delivering the modeling results when they feel ready. Fixed timing implies that the effectiveness measure already includes a statement on efficiency.

Nevertheless time should be recorded to allow analysis of modeling performance over time. Automatic storage of the results in short time intervals allow, e.g. all 5 min or 7 min. the analysis of effectiveness per time in a more specific way.

### 3.2.3. User Acceptance-Subjective Aspects

Another possibly affected variable is the subjects' acceptance of the used notation (or tool) and the automation task when executing the task. Here, usability questionnaires based on the standard DIN EN ISO 9241 are best practice.

Moreover, aspects as the subjects' mental workload, control belief or motivation can be elicited (see Section 5, Section 4.7 for details).

For later analysis the affecting and affected variables and their relations will be evaluated to provide results for the comparison of the notation.

## 4. Selected Usability Studies

In the following Section usability studies (4.1) and usability experiments (4.2 - 4.8) with focus on different automation tasks are introduced and classified according to Section 3 (**Table 1**). The engineering task is classified

as structural and/or behavior modeling task. The automation task's complexity is given by number and type of I/O, as well as weighted methods of class and number of variables in case of a classical PLC programming approach using IEC 61131-3 FBD. The five related experiments are stronger related to case studies and to industrial application.

The seven experiments by the author's team (4.2 - 4.8) highlight different complex automation tasks and different automation systems characteristics as given in **Table 1**.

## 4.1. Related Experiments

### 4.1.1. Experiment O.1 and O.2: Measuring Size and Complexity, Estimation of Development Time

Lucas and Tilbury and Lucas [67] [68] demonstrate how task analysis could be usefully applied for the preliminary assessment of the effectiveness and perhaps even the efficiency of logic control design methodologies. Lucas [67] calculated the time to create a simple logic design program on the basis of low level user operations, e.g. keystrokes, mouse clicks and mental operations, for IEC 61131-3 Ladder Logic Diagrams (LL 405 min), Petri Nets (PN 1100 min) and modular Finite State machine logic (mFSM 1500 min) showing the significant difference given by the notation itself. To derive the necessary steps and the used strategies, *i.e.* copy & paste, manual copy, they observed engineers during the design process and surveyed the time needed. Moreover, Lucas and Tilbury [33] [67] provide a way of comparing the complexity of control logic models respectively code of a simple lab scale MS created with the above mentioned notations plus SIPN by analyzing existing programs. They introduce quantitative measurements of complexity of a piece of code: size (*i.e.* number of operations and state variables), modularity (number of modules) and connectedness. Additionally, they introduce four typical scenarios for accessibility of data from a programmer's point of view, *i.e.* 1) single output debugging (specific questions regarding specific unexpected behavior in the machine), 2) system manipulation (how the user can manipulate the machine to achieve a desired state), 3) desired system behavior (desired behavior of the machine when examining only the schematics and the logic) and 4) unexpected system behavior (system's response to unexpected events). Because all these questions refer to already existing code they can be categorized to maintenance tasks. The four notations evaluated are compared regarding the four scenarios showing that Ladder Logic is still most appropriate for the first two but hard for Scenario 3 and 4, whereas Petri net, SIPN and mFSM are rated moderate or easy in Scenario 3, but minor in Scenario 1. LL is the small but very interconnected and mFSM the most modular, although largest program.

### 4.1.2. Experiment O.3 and O.4: Reusability Strategies

Strömman *et al.* [27] compared IEC 61499 with IEC 61131-3 in logic control design to foster reuse. Professionals and researchers act as subjects programming a lifter application during a workshop. The resulting solutions differ totally showing different type of approaches, e.g. reuse of existing ST Code copied into an IEC 61499 frame, reuse of design patter, *i.e.* a state diagram, a mechatronic approach and classical IEC 61131 function block approach, concluding that guidelines to use IEC 61499 are required as well as an environment that fosters collaboration and exchange of information. The results were gained by model comparison and written feedback. Beforehand interviews were conducted to reveal the relevance of the study. Design approaches are context-dependent, *i.e.* the background of the designers, the existence of legacy software as well as business goals etc.

Based on this experience in experiment E0.4 Sierla *et al.* [28] organized one courses on IEC 61499 in 2005 to enable twenty practitioners and researchers to propose and negotiate about design alternatives in a team context with recorded interviews. In a second course in 2006 for professionals (3 subjects), researchers (3 subjects) and a standardization worker worked in a team representing the different social groups in a project evaluating the impact of team organization, knowledge integration, and software development method by an interview after the course. The benefit of a modular structure was realized as well as the risk of combining continuous control loops combined with sequential batch control logic. The necessity of shared guidelines, design patterns and tool support was highlighted in more detail especially for batch control systems section.

### 4.1.3. Experiment O.5: Change of Sequence

Hajarnarvis *et al.* [46] compared 63 subjects applying for different methodologies changing the sequence of a given simple program, *i.e.* contact logic, step logic, SFC and EC. The participants had to change the sequence of a simple task with three motors and one valve. The authors identified different main problems, e.g. insufficient

modifications for all but EC and incorrect algorithm for SFC and EC. The results are separated according to the participants' background, *i.e.* maintenance, planner, programmers and Rockwell personnel compared to the untrained.

### 4.2. Experiment E1-Pure UML 1.4 and PLC Programming-Exploratory Study

The series of experiment E1 explored the influence of group work compared to individuals, the influence of prior experience in PLC programming and modeling, different qualified subjects, *i.e.* bachelor students of electrical and information engineering with students integrated into companies (StiP) and technicians modeling and programming a pick & place unit [69]-[71] (see **Figure 3**).

As affected variables the number of steps realized and their correctness was evaluated compared to a master model. The notations compared are UML, ICL and a control group only using S7 PLC programming languages IL, LL and FBD.

The results regarding quality of the model, *i.e.* error rates with 43.84% are disappointing. The high impact of qualification level on number of realized steps and errors is significant (see **Table 1**, E1 results). The influence of prior knowledge which is in this experiment only based on subjective rating in a questionnaire is evident, too. In this experiment prior knowledge leads to halve the errors. Subjects rate the applicability of both UML and ICL for modeling structural aspects as very poor and for behavior as fair. Comparing groups (2 subjects) with individuals, groups reach a higher number of modeled steps in (23.44 compared to individuals 15.04; $p = 0.01$), but unfortunately the error rate is not significantly reduced. The experimental results, e.g. the identified errors in the developed models (see 9, **Figure B1**) are used as input for the further development of UML for MS (E2, E3, E4, E5). The pure models and high error rate reveal an insufficient training and experimental design, but also the weakness of pure UML 1.4 as modeling notation. Subjects claimed a reduced number of diagrams with a clear procedure for UML modeling, a tool to support modeling with integrated code generation, because paper and pencil is not accepted.

### 4.3. Experiment E2-Deployment Using Pattern and UML-PA

Based on the results of E1 a domain specific language UML-PA was developed with a reduced number of diagrams and domain specific stereotypes [72]. The research question was to prove the benefit of such a domain specific language under architectural aspects, *i.e.* regarding deployment of control loops and the related sensors and actuators connected via a field bus. The subjects should identify correct pattern and connect them to model the system from sensors to actuators including its deployment and communication relations. For this reason UML-PA provides ports to model communication interfaces in so called instance structure diagram.

The modeling approach using UML-PA and its instance structure diagram is compared with UML 2.0 diagrams, *i.e.* class diagram, component diagram, composite structure diagram and deployment diagram. As automation task a simplified real continuous hydraulic press was chosen with 30 control loops to be switched between distance control and pressure control in case of overpressure. Each valve is equipped with a distance sensor to measure the valve opening and each control loop with a pressure transmitter. As additional input the press operator sets the set values of the pressure in the cylinder connected to the valve. The controllers output is the set value of the valve position and to the HMI the valve opening.

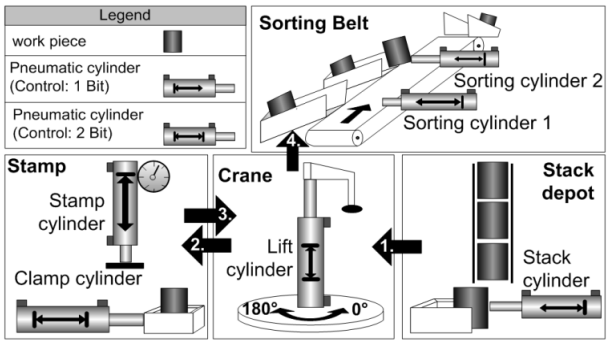


**Figure 3.** Pick & place unit (E1, E3, E4, E5).

UML participants checked their results after 1.78 changes and took the results as guidance to find an appropriate solution, UML-PA subjects checked their solution after 3.9 changes [72] (see also [73] for further information). The subjects properly analyzed the task and selected the given pattern establishing the required communication more efficient with UML-PA compared to UML 2.0 (see **Table 1**, E2), which is easy to understand due to the additional effort, *i.e.* diagram changes, needed in UML 2.0. This idea is included in the SysML-AT approach discussed in E7. The identified breaks and time needed to understand the relation between different diagrams needs to be optimized regarding improvement of MDE (see E5).

Subjects criticized the restricted tool. The restricted tool support encouraged students to follow a trial and error strategy which is unacceptable in a real industrial application.

### 4.4. Experiment E3-Error Handling Using plcUML SC vs. IEC 61131-3-SFC

Fulfilling the requirements from E1, a reduced number of diagrams with tool support and code generation, Witsch and Vogel-Heuser developed a prototypical plcUML editor implementing UML class diagram and state chart in a real IEC 61131-3 run time development with integrated code generation in CoDeSys 3.x [74] [75] (see also [1]). The plcUML diagrams are integrated similar to SFC as additional language transformed internally into a ST language derivate. Yang *et al.* [76] applied orthogonal regions in UML state charts to model primary system functions and corresponding traversal features and concurrent behavior. Witsch *et al.* [74] introduce composite states as groups of states allowing to model error behavior for those grouped states. Evaluation with experts showed the strength of the composite states for error handling as well as mode of operation, the focus of experiment E3.

The experiment validates that using state charts is more efficient than using classical SFC in IEC 61131 to proof cyclically sensor states regarding inconsistency as well as timing errors in a single moving cylinder, *i.e.* a cylinder component of the pick &place unit (cylinder in **Figure 3**). The mean steps programmed per minute using state charts with composite states was 1.98 points/min compared to classical SFC in IEC 61131-3 with 1.41 points/min given the same points for both solutions to be reached [77]. The modeling speed of the SC group was significantly higher than the SFC group even if the SC subjects didn't use composite states.

The benefit of composite states is evident for error handling (see **Figure 4**, left), *i.e.* in SC the error handling for all states can be handled by one exception transition out of a composite state instead of multiple transitions, *i.e.* after each activity, error handling activities follow. If an error in the exception handling algorithm is identified or an additional condition needs to be included modifications to the process can be covered in one path in SC, compared to multiple paths in SC (cf. **Figure 4**, right).

Subjects using composites states estimate their programming experience higher than those who didn't use composites states. Many subjects criticized the absence of an automatic placement of elements in the tool, a site effect in the plcUML condition. In this experiment only exception handling was evaluated with a prototypical

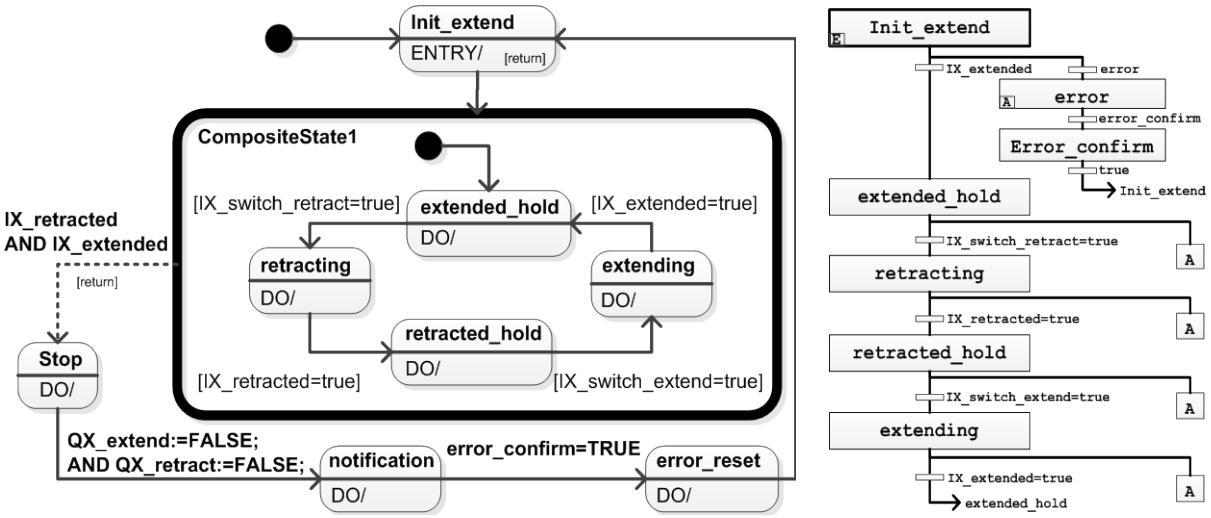


**Figure 4.** Comparison of subjects' best solution in SC group (left) and SFC group (right).

tool. A more general design is discussed in Experiment E5 using plcUML with a more mature tool version.

### 4.5. Experiment E4-Sequence of Structure and Behavior Modeling in Workflow Using UML with Elaborated Training Concepts

The research question to be answered is, whether subjects can be successfully forced to model structure, when asking them to model structure before behavior or whether behavior first is a good strategy for engineers to achieve proper model quality. Therefore a training concept as well as a subsequent experiment has been developed together with researchers from instruction theory [78].

In a pre-experiment for E4 the main focus was to reveal whether the order of modeling is important for the quality of the model. The assumption was that students start with behavior modeling because it is easier for them, and then run short in time before finishing the structural model.

The pre-experiment was conducted without tool support only with paper and pencil after a training realized by a lecture and exercise in a very large classes (bachelor students 2nd semester mechanical engineering). The subjects were split in two groups: one group was told to start with structure modeling, the other with behavior modeling.

It showed that 35% of the subjects had problems to create suitable classes from similar objects of a plant including their attributes, and methods.

Examples of typical errors in the class diagram were (error rate in %):

- Objects were listed in addition to the classes, which inherit from the classes (23%).
- Classes were used in which objects of the class occur as attributes (7%).
- Single objects were modeled without classes (5%) [78].

Overall, no significant differences concerning the modeling order could be found, but significant differences with respect to the trainer, as the two groups were trained by different teachers.

In order to eliminate that confounding effect in the main experiment one trainer trained for both groups. In that study, which has not yet been published, a larger sample (102 subjects) has been tested using the same procedure and task as described for the pre-experiment above.

Here, the average participant reached 19.97 out of 46 points, *i.e.* lacked 26.03 points (SD = 9.1819). Regarding the performance measures, the "behavior first" group scored remarkably higher than the "structure first" group: While the participants in the "behavior first" group achieved 23.4 points on average (SD = 10.326; SE = 0.982), the mean value of the "structure first" group was only 18.4 out of 46 points (SD = 8.220; SE = 1.825).

In contrast, the structural modeling performance of the two groups was comparable ($T = -0.972$, $df = 100$, $p = 0.33$): participants in the "structure first" group achieved 12.26 points on average (SD = 5.029; SE = 0.601), while in the "behavior first" group they reached 11.14 points (SD = 6.067; SE = 1.072).

In the "structure first" group, the subjects reached only a mean of 6.14 out of 24 points in behavior modeling (SD = 5.083; SE = 0.607); in the "behavior first" group, however, the average behavior modeling performance was 12.25 points (SD = 6.112; SE = 1.080). This difference is highly significant ($T = -5.278$, $df = 100$, $p = 0.00$).

As a result for the next experiments we learned that the class room training was not suitable enough and that forcing students to follow a specific modeling order is not helpful to improve structural models.

### 4.6. Experiment E5-plcUMLvs IEC 61131-3 FBD with Apprentices Optimizing Training, Design and Analysis of Results-Exploratory Study

In this experiment the superiority of UML compared to FBD in a design task with a sophisticated training and with repetitive application of the notation, the ß-version of an UML tool (called plcUML), for a complex open loop control task and apprentices as subjects should be demonstrated. To allow further analysis between modeling results and subjects' abilities and the development of an individual training fitting to individual abilities in a next step, selected abilities are collected as well as user acceptance.

As control task a sub-part of the pick & place unit with multiple reuse (only open loop control, weak real time requirements without communication requirements) should be modeled, *i.e.* three storage elements with one storage cylinder pushing the work pieces out of the storage and five different terminals with a terminal cylinder each, pushing the work pieces into the terminal. Because in industry very often skilled workers are conducting maintenance tasks and even easy design modifications 1st and 2nd year apprentices from a vocational school in Munich (89 subjects) act as subjects. Selected results of this experiment are reported already in [63].

A hybrid learning environment (HLE), allowing to switching between computer-based and conventional instructional designs] was developed and implemented. During training the groups repeatedly exercised programming and modeling tasks with increasing complexity (named fade out).

Several affecting variables related to abilities were obtained, *i.e.* grades in mathematics, German, automation, and mechatronics as well as cognitive capabilities, motivation levels, challenge, and workload (single instruments are described in [63]). As performance variable the programming/modeling achievement was evaluated. To obtain this value, the developed models/programs were stored (every 5 min) and analyzed manually by two evaluators, who compared them to a master model. The subjects performance was measured as number of correctly modeled or programmed elements and compared with respect to structure, e.g. classes or FBDs on the one hand and behavior, *i.e.* state charts and FBDs on the other (for details see Appendix A). Unfortunately, the results were disappointing, because an overall significant benefit of plcUML compared to FBD could not be detected, but nevertheless interesting results could be found, e.g.

- OO modeling and FBD programming show different relations to variables like cognitive abilities, experience, workload, and knowledge the students' performance in the plcUML/CD + SC groups seems to be less related to previous knowledge and cognitive abilities than students' performance in the 61131/FBD groups [63].
- Subjects needed different times for structural modeling using UML/CD vs. FBD (see master model **Figure 5**). Subjects needed in average 6.22 minutes more time for UML class creation (in comparison to the time needed to create the FB structure. This difference is slightly not significant (ANOVA, $F(1, 81) = 3.60$, $p = 0.06$), cf. [63].

On the basis of the unexpected results further analysis of models, modeling process and the relation between model and results as well as subjective results have been conducted. Analyzing the main errors especially the errors in structural model, *i.e.* classes built:

- 42 subjects out of 44 built classes (including superfluous ones) as part of the structural model;
- 23 out of 42 used these classes in their behavioral model;
- 31 out of 42 modeled a second cylinder class separating storage cylinder and terminal cylinder, but 14 out of those 31 subjects built the second class identically besides the name, this indicates that they understood the class concept but use another type of abstraction, which is more related to the mechanical structure, *i.e.* a terminal and a storage cylinder are different instead of the software view in which both cylinders are identical.

Analyzing tool and training effects gathered from the subjective rating from questionnaire (**Figure 6**), 27 subjects of the plcUML grouped asked for additional training. From subjects' observation and analysis of the time needed, the authors expected the abstraction needed to build classes and the relationship between CD and SC to be the main challenge, because in the UML groups long thinking breaks occur before modeling classes. In the questionnaire only 5 subjects mentioned that development of classes is difficult, which is surprising regarding the above mentioned errors in building classes and the thinking breaks.

Regarding tool aspects (item 1.2 positive and item 2.5 negative in **Figure 6**) the plcUML tool seems to have some more problems.

Further subjective results gained from the questionnaires were: 1) Frustration levels were significantly higher in the UML group compared to the FBD group ($p = 0.02$); 2) The clearness of FBD was rated significantly higher than of UML ($p = 0.017$); 3) Behavior programming was rated significantly easier with FBD than with

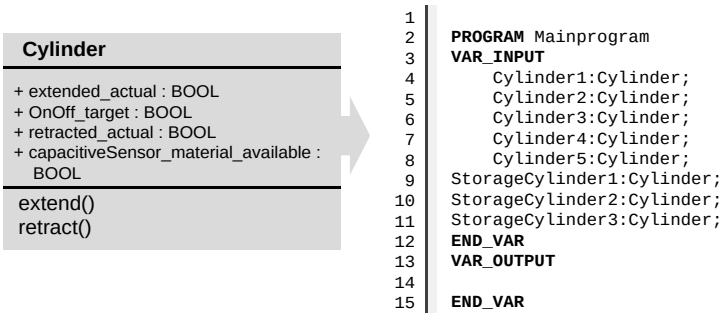


**Figure 5.** plcUML class diagram master model integrating storage and terminal cylinder and its IEC representation.

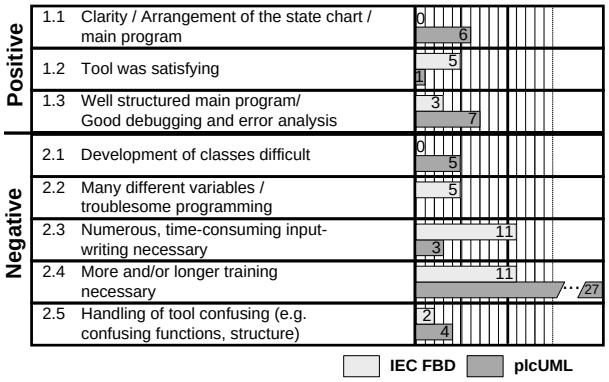


**Figure 6.** Subjective statements after the experiment.

UML ($p = 0.012$). And 4) subjective quality estimation and factual quality match far better with UML than with FBD ($p = 0.025$).

Because of the observed thinking breaks, we analyzed the modeling progress over time (points over time) in a random sample of only three subjects (with similar quality of model) we found differences in plcUML and IEC group, in the plcUML group there is a longer period of time until points referring to the master model are gathered and there is a clear ramp in points compared to the more steady increase of points in FBS group (**Figure 7**).

For further experiments detailed analysis of modeling progress is needed and, therefore, the cycle time of storing data needs to be reduced and a more efficient approach of analyzing subjects' modeling process over time needs to be developed. Details for the analysis and rating of subjects' models compared to the master model are given in Appendix A.

Subjects debugged at different times, some at the beginning and others at the end of the experiments with a nearly complete model. The analysis of debugging is necessary to find errors and will be focused in future work.

The design of the experiment including training and data analysis was appropriate delivering detailed relations between abilities and model quality, but revealing still shortcomings of plcUML as notation for apprentices in design tasks. Our assumption is that the necessary abstraction to build classes is too high for this group of subjects. These results fit to the notational complexity of class diagrams of Schalles. Therefore, in further experiments technicians and engineers will be included as subjects with a higher level of knowledge and experience in PLC programming. Additionally, different levels of task complexity will be tested.

## 4.7. Experiment E6-Maintenance Task in Early Phases of Notation Development with SysML-AT vs. Continuous Function Chart (CFC)

The research question of experiment E6 is how to evaluate three notations in a qualitative way in a very short period of time for training and experiment in an early phase of the development of a notation. E6 evaluates different modeling and programming notations (see also [16]), *i.e.* Parametric diagram (PD) of SysML-AT [26] vs. Continuous Function Chart (CFC) and IEC 61131-3 Structured Text (ST), regarding a maintenance task, *i.e.* understandability (analysis and interpretation according to [31]) of model contents in a qualitative way. The experiment was based on three different simple models of physical laws (about 4 - 5 sub-blocks and 7 - 8 variables), with each model described in every considered notation. Because the evaluation should take place in an early design phase and the time needed should be very short, tool support is not applicable. Bachelor students of mechanical engineering worked without a tool after a very short training, passing all three different notations and all three models. The sequence of the notations was permuted for each subject (see **Figure 8**) to eliminate learning effects.

As a software maintenance scenario, the subjects had to correctly interpret the models' contents, consisting of components (sub-blocks and variables) and data flows to answer questions regarding the model contents correctly. The mean of correctly answered questions was highest for the PD (68.25%) with a positive offset of 3.97% to ST (64.28%) and 4.76% to CFC (63.49%) (see **Table 1**, E6). The experiment shows, that even a short training with a short time for experiment and a small number of subjects delivers qualitative results. In accordance with

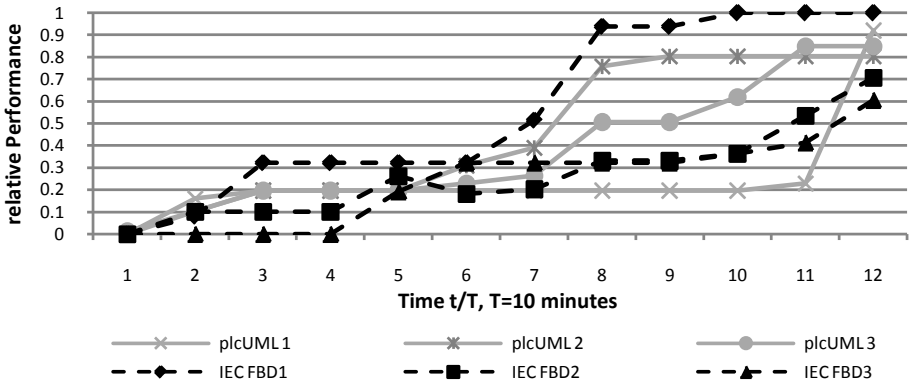


**Figure 7.** Modeling progress over time for 3 subjects of each group.

| Subject 1 | Subject 2 | Subject 3 | Subject 4 | Subject 5 | Subject 6 |
|---|---|---|---|---|---|
| Previous Knowledge Tests & Trainings | | | | | |
| Model A CFC | Model A CFC | Model A PD | Model A PD | Model A ST | Model A ST |
| Model B ST | Model B PD | Model B ST | Model B CFC | Model B PD | Model B CFC |
| Model C PD | Model C ST | Model C CFC | Model C ST | Model C CFC | Model C PD |
| Questionnaires for subjective cognitive demand & discussion | | | | | |

**Figure 8.** Experimental design of E6-maintenance task.

the results, in questionnaires that tested the subjective cognitive demand, the subjects rated the PD as the most understandable notation. Furthermore, all of the subjects answered, that they experienced a learning effect regardless of the different notations they used.

The results of a focus group that was conducted for additionally evaluating the SysML-AT [25] also indicated that the developed modeling approach is well suited for automation software modeling.

## 4.8. Experiment E7-Conceptual Engineering of Structural aspects of Distributed Networked Automation Systems (NAS)

The research question in this experiment is whether additional support in structural modeling of NAS realized with characteristics and pattern is beneficial in the conceptual design or whether the resulting complexity hinders the benefit. Besides the instruments regarding user acceptance are more elaborated and should give answers in more detail in relation to quality of models.

As the results of E5 and E6 show, plcUML and SysML-AT have positive influence on the programming of a PLC. Following Sierla [28] and the difficulties identified for engineering of distributed systems E7 evaluates a SysML-AT based notation and workflow vs. CFC for a high-level design of NAS in MS (see also [16] [45]). The evaluated approach focuses on the overall design of NAS integrating notation SysML-AT being the successor of plcUML. The SysML-AT based concept contains a workflow procedure referring to a life cycle model (following requirements E1) including communication and real time requirements for a hybrid control task (experiment E7 a). Additionally, characteristics (E7 b) and characteristics plus patterns in (E7 c) are compared to the pure notation and workflow procedure. Conditions b and c are only qualitative measures, because of the small group size.

The approach covers the modeling of automation hardware and software as well as of functional and non-functional requirements. From described requirements the functions that need to be implemented can be derived and captured within the same model. Hardware elements like sensors, actuators and nodes and their interfaces and properties are considered within the modeling approach as well. This enables the integration and linking of hardware and software models [79]. The notation is based on the SysML Block and Requirements diagram using ports to represent software and hardware interfaces (Refer also to [80]. Duration of experiment was not restricted, but taken as measure (mean given in **Table 1**, E7).

Characteristics as well as pattern supported subjects in solving the task, *i.e.* design of the automation concept

of a coking plant including belt synchronization without implementation. Main task of the experiment was to conceptually design a closed loop for speed synchronization which included three belts. This comprised all necessary functions, interfaces and relations to the sensors and actuators. The internal behavior and control algorithms were not required, *i.e.* the structural part of the model needs to be designed. Characteristics detail requirements as well as the later design solution including element relations. During the design the comparison of requirement characteristics and solution characteristics help to decide if the solution fits the requirements (cp. **Figure 9**).

Additionally patterns, divided into functional and deployment patterns, help to find a solution. Functional patterns include proposals and support the engineer in the development of the functional model. Deployment patterns indicate distribution alternatives of functions and support the engineer in the development of the deployment model [82].

The models subjects stored after finishing the given task were analyzed compared to a master model. The results show major difference between the subjects' solutions and master model, *i.e.* experts' best practice regarding module structure. Similar to Sierla [28] different possible solutions were detected, *i.e.* most subjects chose a functional oriented modeling approach, instead of a mechatronic approach taking modularity, reuse and architectural aspects of NAS into account. The experiment intended that students follow a mechatronic approach, therefore the master model was built realizing the mechatronic paradigm. In further experiments either the mechatronic approach needs to be integrated into the training or subjects' mental models need to be collected beforehand.

Nevertheless, the experiments show a significant benefit of SysML-AT compared to CFC (see **Table 1**, E7 a to c column results). Regarding notation with life cycle model, *i.e.* NM subjects gained significant better models compared to CFC (123.1 mean compared to 182 max. points, the best subject gaining 144 points, see appendix C). Using characteristics additionally (NMC), subjects improved their models again. But for those experiments only qualitative results are available due to the limited number of five subjects. Regarding user acceptance measures subjects stated less mental demand using pattern (see **Table 3**), *i.e.* NMCP has lowest mental demand with 10.75 of max 20 points. (the higher the more mental demand). The motivational factor "fear of failure" was most pronounced in the group with patterns and characteristics (NMCP). Furthermore, this group showed high external control beliefs meaning that subjects strongly related the outcome of the results to external circumstances and high fatalistic externality meaning that success is assessed as depending on fate, fortune, and chance and, however, subjects perceived low mental demand during task performance. In addition, according to usability aspects, suitability for task was best evaluated for group with characteristics (NMC). Suitability for individualization of patterns was rated significant lower than both other conditions.

Based on UML-PA and E2 as well as the experiences gained and rules derived for E5 (including task development, training and tool development) and experimental design in general (see 6) the experimental design of E7 was developed appropriately evaluating also the derived rules (see 6). E7 evaluated the benefit of NM for NAS and hybrid control with real time and communication requirements. Even relations to abilities realized in E5 could be further developed with a more advanced questionnaire. Results reveal more relations to human factors, e.g. mental workload, and usability measures. For further engineering support the challenge is to find a compromise between supports by characteristics and pattern and the approaches' complexity.

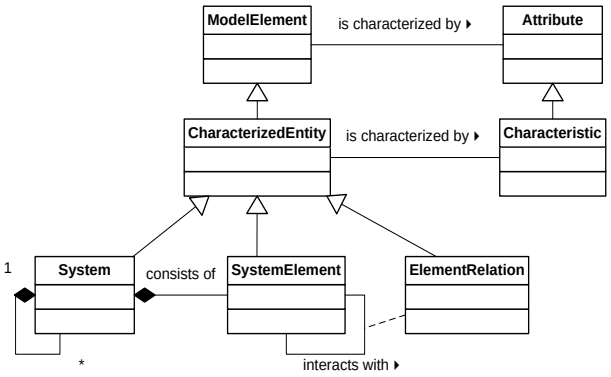


**Figure 9.** Characteristics meta-model [81].

**Table 3.** Results of human factors and usability measurement in E7.

| Variable | Mean | | | |
|---|---|---|---|---|
| | NM | NMC | NMCP | p |
| **Motivation** | | | | |
| Fear of failure | 14.26 | 15.00 | 23.67 | $p < 0.05$ |
| **Control beliefs** | | | | |
| Fatalistic externality | 45.95 | 55.20 | 60.75 | $p < 0.05$ |
| **Mental workload** | | | | |
| Mental demand | 14.28 | 14.80 | 10.75 | $p < 0.05$ |
| **Attributes for usability requirements** | | | | |
| Suitability for task | 27.35 | 30.00 | 20.75 | $p < 0.055$ |
| Suitability for individualization | 26.59 | 26.00 | 13.75 | $p < 0.001$ |

### 4.9. Summary of the Experiments

All experiments focus on the design phase besides E6, a centralized single PLC as control hardware besides E7 and students as subjects besides E1 and E5.

- E1 was the first experiment exploring the method of usability evaluation in logic design engineering with a single closed loop controller and compared pure UML 1.4 and PLC in a first attempt without the support of an engineering tool and with a large unstructured task.
- E2 focused on a hybrid automation task including communication with the focus to support deployment by simple UML-PA pattern compared to classical UML 2.0 with restricted tool support and a narrow engineering task.
- E3 focuses on error handling comparing plcUML State Chart to Sequential Function Chart (SFC) in IEC using a very simple automation sub-task and a short classical training.
- E4 focuses on SC vs. IEC 61131-3-SFC sequence of structure and behavior modeling in workflow using UML 2.0 with a didactically more elaborated but classically conducted training concepts in smaller sub groups with the goal to increase the quality of the structure model.
- E5 is similar to E1 also an exploratory experiment further developing the method of usability engineering experiments using a real software engineering tool with embedded UML the so called plcUML compared to IEC 61131-3 FBD with apprentices optimizing repetitive training, and exercise, an elaborate training environment and smaller automation task with reusable sub-process, including also human factors and prior knowledge.
- E6 focuses on a maintenance task in the early phases of notation development with SysML-AT vs. Continuous Function Chart (CFC) to show the benefit of easy and quick sub-experiments in the development process of the notation.
- E7 Conceptual Engineering of Structural aspects of distributed networked automation systems (NAS) including a procedure for life cycle support and characteristics for pattern selection and reuse with a detailed analysis of user acceptance including motivation.

In every single description of an experiment the research questions as well as the most important aspects of the experimental design, results and lessons learned regarding usability aspects are discussed as well as results for further development of MDE, *i.e.* notation, procedure and tool. Most experiments are based on prior experiments and notational development resulting from a prior experiment is tested in one of the following experiments.

## 5. Selected Results for Future Usability Experiments

The following section summarizes the best practice rules gathered to the best of our knowledge. At first the criteria for the selection and configuration of the affecting variables (see **Figure 1**) are discussed, e.g. the task, the

training and selecting a group of subjects. Afterwards the criteria for selecting the affected variables are discussed (see **Figure 2**).

## 5.1. Configuration of Affected Variables

### 5.1.1. Task Development

As affecting variable (see **Figure 1**), the type of the engineering task (maintenance or design) and automation task complexity and characteristics are key issues in relation to the complexity of the new notation or approach to be evaluated and the time available for training and the experiment itself.

**1) Automation Task**

To classify or rank the automation task complexity compared to other experiments and to estimate the time needed for training as well as the task itself in the experiment, the authors introduced some measures, *i.e.* number of I/O, number and type of control loops and depending on the used notation the WMC and number of states for OO design and the number of FBDs and variables for classical PLC programming using IEC 61131-3. Besides the task characteristics, *i.e.* real time, communication requirements and the tasks type as well as the inclusion of exception handling (E3) and mode of operation are relevant, too. In the above introduced experiments the WMC reaches from 3 in E2 a strongly restricted experiment using pattern to 43 in E5 and 45 in E1 in a more industrial related scenario.

It is obvious that a complete engineering task consists of a lot of decision points with different ways to a correct solution. These variation possibilities need to be covered by an evaluation scheme.

**2) Engineering Task**

Starting with HTA or GOMS, the required steps to fulfill the task are found. The quality of the HTA depends on the skills and experience (also industrial) of the experts conducting the HTA. Interviews with industrial experts are helpful to find appropriate subtasks as well as typical module libraries available to be provided in the experimental setup.

Modeling mostly consists of structural and behavioral aspects. In most of the experiments described above, structure and behavior were an issue (**Table 1**, column engineering task). All experiments besides E6 dealt with design and model creation (E2 model configuration). E6 highlights maintenance tasks and showed that tool support may be neglected for easy tasks as well as training may be very short compared to design tasks.

For both, modeling and training, the designer has to decide whether to provide a life cycle model or even a method and a tool. For more complex engineering tasks a tool is a prerequisite to gain subjects acceptance (not reached in E1, E4 and E6) and motivation. On the other hand, a prototypical tool (E3) leads to results that may be induced by the tool and not by the notation to be evaluated. Sophisticated tools need additional time for training. The prototype plcUML or SysML-AT, therefore, needs to be carefully tested by novices and persons belonging to the qualification group of future users before conducting the experiment, to ensure an effective detection of as many defects of the tool as possible prior to the experiment. Since otherwise frustration will rise and may act as disturbing factor in the experiment (E7 c).

### 5.1.2. Development of Training

As discussed above, an appropriate training is a prerequisite for meaningful results, but hard to achieve (E5 not E7 c)) in the first experiments. A hybrid learning environment is advantageous to reduce disturbances by individual trainers as in the pre experiment of E4. Furthermore, process simulation offers high benefits as to testing and debugging the software. For more complex notations and procedures, e.g. OO and UML, repetitive training with fade out is beneficial (E5). A training period of 1.5 days for OO with apprentices as subjects and 0.5 days for E7 a) with students as subjects was appropriate. With a very simple task or strictly focused hypothesis and a restrictive tool significantly shorter duration can be reached (E2 and E6).

### 5.1.3. Selection of Subjects

Besides E1, we decided for individual subjects to allow the identification of reasons and dependencies to individual abilities. This excludes to examine benefits of group work as found in Sierla [28]. In the field of MS engineering students are a typical group of subjects for design tasks as well as technicians and apprentices for maintenance tasks and simple design modifications at customer site. The necessary numbers of subjects per cell to gain quantitative results is minimum 15. Different skills and abilities, e.g. mathematics are often related to

results and act as disturbance factors.

Pre-tests are recommended to adjust distribution of subjects to groups regarding expertise and abilities. Different tests are available (E5) or adaptable (e.g. on general intelligence [83] or on previous knowledge). Missing or insufficient motivation may also be a disturbing factor as realized in experiment E7c. Also, mental workload, *i.e.* the cognitive demand perceived during modeling tasks is a critical factor for the probability of errors and, therefore, should be at an intermediate level (E5 and E7). When analyzing specific aspects of a notation in more detail after the main experiment, group sizes from 6 to 8 are regularly implemented to get qualitative results. In E6 the sequence of the notations was permuted for each subject to eliminate learning effects instead of using one notation for one group, which in case of E6 would have multiplied the necessary number of subjects by three.

## 5.2. Measuring Affected Variables/Usability Requirements

To analyze the gained result and to evaluate it, master models are recommended, developed by the designer of the experiment together with other experts.

### 5.2.1. Data Collection-Organizational and Technical Challenges

For the data analysis observation and recording of subjects' results are most important. The easiest way to observe subjects is to take a video, but the manual analysis of the video is time consuming. In engineering tasks using an engineering tool, the most often implemented strategy is to store the model cyclically with a selected time (all 2 or 5 minutes E5 and E7 5 min) or if a new input is typed in the model (E2). The cyclical storing strategy has the disadvantage of losing information in between storing intervals similar to the sampling of an analogue value. Storing the model with every subject's input has the disadvantage of large amounts of data, which need to be analyzed later. The strategy may not be integrated in real tools as necessary if using a *ß*-version of an industrial tool (in E5). The challenge is to implement storing strategies in the prototype or to get access to a market leading tool in case it should be used for evaluation. The CoDeSys implementation was easy to realize for the authors' team due to the gathered developer's knowledge of the plcUML-Plugin. Additionally to model analysis, human observers are advantageous especially in case of pre-experiments and to include additional information gathered by observation. Unfortunately, this is expensive, because the observers need to be trained; the observation needs to be documented in a standardized form and approximately 1 observer is required for 2 - 4 subjects. In E5 long periods of thinking breaks in the OO groups before building classes were found and included in further analysis. The analysis of results gained, *i.e.* model consolidation over time seems to be useful, but is depending on the availability of data and ease of analysis.

In psychology thinking aloud is an often implemented method, which is often not accepted and applicable by engineering students (E1). Another issue is to gain information why subjects make mistakes or chose a specific solution. To a certain degree this information may be gained by individual interviews or online questionnaires directly after the experiment (E4). In E4 subjects were asked to analyze their solutions compared with the master solution and give reasons, e.g. lack of time, translation problems, distraction etc., for their mistakes. The method is promising, but hard to realize with large groups of subjects because of possible interviewer effects with regard to the questions asked.

### 5.2.2. Effectiveness

Usability evaluation concerning affected variables, *i.e.* effectiveness, efficiency and user acceptance was realized with different methods. To assess effectiveness, completeness and correctness are measured by counting the numbers of correct steps compared to the master model, e.g. in the behavior model, e.g. a state chart the number of steps, in the structure model in FBD the number of variables, the number of classes and objects in a class diagram (for evaluation scheme E5 see Appendix A).

The difficulty in rating the results of an experiment is comparable to grading exams by distributing points for correct solutions, but more sophisticated. Points are given by two evaluators independently with a necessary interrater reliability of at least 65% (E5, E7 see Appendix).

### 5.2.3. Efficiency

To evaluate efficiency time stamps need to be included in the stored data and analyzed or as mentioned in 1) the cyclically stored data are taken to analyze efficiency over time. In most experiments efficiency is effectiveness

in the given period of time subjects got for the experiment. In most experiments time was limited due to organizational reasons, besides E7 where time was taken as a variable: When subjects felt ready they submitted their solution and the time needed was stored.

#### 5.2.4. User Acceptance

For evaluation of user acceptance in all of the above described experiments questionnaires based inter alia on the EN ISO 9241 and on recognized tests as RSME [83] and NASA-TLX [84] were implemented and further developed from one to the next experiment to analyze subjective values regarding modeling as such, the notation evaluated and/or the tool used, e.g. E1 and E5. Furthermore, extended evaluation of attributes for usability requirements examined by EN ISO 9241-110 questionnaire (E7) was additionally used to collect users' assessment of applicability of patterns and characteristics. Results revealed suitability for task and for individualization as appropriate indicators of difference. Questionnaires regarding the notation and tool may also reveal weaknesses of training and notation (E5 class concept).

## 6. Selected Results for the Development of Future Notations for Model Based Software Engineering

In MS, hybrid control tasks, real time and communication requirements of different complexity need to be engineered during design and maintained during operation covering structure and behavior in MS models.

From the results of E1, we realized that pure UML 1.4 with its five diagrams used in E1 is confusing and not appropriate especially for structure models. Additionally embedded tool support in PLC development environments and a procedure is requested by subjects. Forcing students to follow a specific modeling order, e.g. behavior or structure first (E4) is not helpful to improve structural models. The introduction of plcUML embedding class diagrams and state charts into an IEC 61131-3 tool enlarged with composite states for error handling showed benefit, but tool aspects as placement were criticized (E3). In E5 a more general, but simple logic design task with reuse revealed weaknesses of plcUML in design tasks for apprentices using a ß-Version of the tool. The challenge for apprentices was the necessary abstraction when building classes. Weaknesses in training and tool were criticized (**Figure 6**). The tool has been further developed and integrated in CoDeSys by industry in June 2013 now used in different industrial companies and research. Experiments focusing on maintenance tasks, evaluated in E6 with students of mechanical engineering, indicated that the SysML-AT PD has advantages compared to CFC and ST (qualitatively).

All these evaluations concentrated on the automation software of one centralized PLC. Regarding deployment and NAS two experiments were conducted, *i.e.* E2 and E7 including communication and real time requirements. In E2 a domain specific UML the UML-PA with reduced number of diagrams was beneficial in deployment of software to hardware devices like PLCs, using patterns with a very simple conceptual control task. The restricted tool was criticized, but the reduced number of diagrams was advantageous compared to UML 2.0. plcUML, consists of the Class Diagram for modeling software structure as well as the Activity Diagram and State Chart for modeling discrete software behavior using Activity Diagrams in the early phases of the software lifecycle for specification issues and the State Chart for detailed modeling of behaviour. Further developments of plcUML, namely the SysML-AT added the SysML Parametric Diagram for modeling constraints as mathematical equations to describe physical laws to the diagrams of plcUML. Although advantages of both notations were noticed, the results from E6 (focus group) indicate that a MDE approach for MS has to consider and support requirements analysis and architectural design and a supporting method. Especially for NAS the architectural design is even more important. Such a method was developed and positively evaluated in experiment E7 to be most appropriate for all typical requirements of automation in MS. Recent works currently develop an approach that contains the developed methodology for NAS and requirements modeling followed by software modeling and generation based on the plcUML and SysML-AT.

## 7. Conclusion and Outlook

MDE approaches should increase efficiency and quality in design and maintenance of software engineering for MS. The article showed results of usability experiments using pure UML, domain specific UML versions, *i.e.* UML-PA and UML E as well as domain specific SysML-AT for mainteance purposes and NAS. Summarizing the most important technical issues pure UML 1.4 or 2.0 is not appropriate, but plcUML with reduced number of

diagramms and a supporting modeling process integrated in an IEC 61131-environment to support roundtrip engineering. For error handling plcUML SC with commmposite states are beneficial compared to IEC 61131-3 FBD. Structural modeling using pure or even plcUML is still a challenge for many subjects as well as the creation of classes in the sense of abstraction used in computer science. Abstraction in automation and mechatronics is different to computer science, *i.e.* more related to physics, also in distributed systems application. Complexity of notation (class diagramm and E7) relates to difficulties in applying the notation in an experiment with time restrictions (2 days). For NAS the applicability of notation was positively and quantitaive evaluated and for characteristics and pattern further experiments and longer training time is needed. Ongoing research is looking at a detailed analysis of humans' mistakes trying to find reasons by interviewing subjects after the experiment.

Regarding real industrial software engineering tasks in MS all these experiments lack of experienced subjects, *i.e.* application engineers and the start-up phase with debugging. Real applications and some applications engineers are included in [85]. The classical debugging phase to find faults is not explicitly analyzed up to now even if Myers [86] provides an interesting approach to classify runtime faults and the underlying software errors. Debugging in E5 was limited to simulation and restricted due to given time. At the moment we implement inteviews after another experiment focusing on reuse of modules with apprentices to analyse faults categorized to Myers' classification.

Regarding usability aspects the presented experiments proofed the relevant affecting and affected variables (**Figure 1** and **Figure 2**) to be taken into account when designing the experiment.

To increase efficiency and quality of software in the development process of an industrial company in machine and plant manufacturing model based approaches using notations as UML and SysML are applicable and could be proven as partially quantitive beneficial. The prerequiste for a real benefit is the availability of an integrated tool support in the IEC 61131-3 especially for maintenance reasons to guarantee consistency of model and implemented code. Nevertheless, it is will not be easy to introduce and implement MDE using UML and SysML in an industrial company. Training and rules for application are necessary as well as a workflow to integrate existing legacy software developed in years. To integrate legacy software the existing software needs to be analyzed at first and modularity concepts need to be developed as a prerequisite for MDE. Variablilty analysis from software engineering should be implemented to maintain and evolve models and code synchronously.

Further research is also needed regarding the integration of more advanced controllers into the usability evaluation, e.g. modeled in Matlab/Simulink.

## Acknowledgements

The author gratefully acknowledges the support of the German Research Foundation (DFG) for the projects DisPA (Vo 937/2-1), KREAagentuse (VO 937/8-1) and FAVA (VO 937/13-1) and the support and fruitful discussions with Christoph Legat, Daniel Schütz, Kerstin Duschl, and Martin Obermeier.

## References

[1] Basile, F., Chiaccio, P. and Gerbasio, D. (2012) On the Implementation of Industrial Automation Systems Based on PLC. *IEEE Transactions on Automation Science and Engineering*, **10**, 990-1003. http://dx.doi.org/10.1109/TASE.2012.2226578

[2] International Electrotechnical Commission (2013) IEC International Standard IEC 61131-3: Programmable Logic Controllers, Part 3: Programming Languages. IEC, Geneva.

[3] Thramboulidis, K. (2010) The 3+1 SysML View-Model in Model Integrated Mechatronics. *Journal of Software Engineering & Applications*, **3**, 109-118. http://dx.doi.org/10.4236/jsea.2010.32014

[4] Rzevski, G. (2003) On Conceptual Design of Intelligent Mechatronic Systems. *Mechatronics*, **13**, 1029-1044. http://dx.doi.org/10.1016/S0957-4158(03)00041-2

[5] Zhabelova, G. and Vyatkin, V. (2012) Multiagent Smart Grid Automation Architecture Based on IEC 61850/61499 Intelligent Logical Nodes. *IEEE Transactions on Industrial Electronics*, **59**, 2351-2362. http://dx.doi.org/10.1109/TIE.2011.2167891

[6] Sauter, T. and Lobashov, M. (2011) End-to-End Communication Architecture for Smart Grids. *IEEE Transactions on Industrial Electronics*, **58**, 1218-1228. http://dx.doi.org/10.1109/TIE.2010.2070771

[7] Vyatkin, V. (2013) Software Engineering in Industrial Automation: State-of-the-Art Review. *IEEE Transactions on Industrial Informatics*, **9**, 1234-1249. http://dx.doi.org/10.1109/TII.2013.2258165

[8] Yang, C. and Vyatkin, V. (2012) Transformation of Simulink Models to IEC 61499 Function Blocks for Verification of Distributed Control Systems. *Control Engineering Practice*, **20**, 1259-1269. http://dx.doi.org/10.1016/j.conengprac.2012.06.008

[9] Dubinin, V., Vyatkin, V. and Pfeiffer, T. (2005) Engineering of Validatable Automation Systems Based on an Extension of UML Combined with Function Blocks of IEC 61499. *IEEE International Conference on Robotics and Automation* (*ICRA*), Barcelona, 18-22 April 2005, 3996-4001.

[10] Secchi, C., Bonfé, M. and Fantuzzi, C. (2007) On the Use of UML for Modeling Mechatronic Systems. *IEEE Transactions on Automation Science and Engineering*, **4**, 105-113. http://dx.doi.org/10.1109/TASE.2006.879686

[11] Bassi, L., Secchi, C., Bonfé, M. and Fantuzzi, C. (2011) A SysML-Based Methodology for Manufacturing Machinery Modeling and Design. *IEEE/ASME Transactions on Mechatronics*, **16**, 1049-1062. http://dx.doi.org/10.1109/TMECH.2010.2073480

[12] Bonfé, M., Fantuzzi, C. and Secchi, C. (2013) Design Patterns for Model-Based Automation Software Design and Implementation. *Control Engineering Practice*, **21**, 1608-1619. http://dx.doi.org/10.1016/j.conengprac.2012.03.017

[13] Thramboulidis, K. and Frey, G. (2011) Towards a Model-Driven IEC 61131-Based Development Process in Industrial Automation. *Journal of Software Engineering and Applications*, **4**, 217-226. http://dx.doi.org/10.4236/jsea.2011.44024

[14] Thramboulidis, K. (2012) IEC 61131 as Enabler of OO and MDD in Industrial Automation. *IEEE International Conference on Industrial Informatics* (*INDIN*), Beijing, 25-27 July 2012, 425-430.

[15] Obermeier, M., Braun, S. and Vogel-Heuser, B. (2014) A Model Driven Approach on Object Oriented PLC Programming for Manufacturing Systems with Regard to Usability. *IEEE Transactions on Industrial Informatics*, **1**. http://dx.doi.org/10.1109/TII.2014.2346133

[16] Vogel-Heuser, B., Schütz, D., Timo, F. and Legat, C. (2014) Model-Driven Engineering of Manufacturing Automation Software Projects—A SysML-Based Approach. *Mechatronics*, **24**, 883-897. http://dx.doi.org/10.1016/j.mechatronics.2014.05.003

[17] Estévez, E. and Marcos, M. (2012) Model-Based Validation of Industrial Control Systems. *IEEE Transactions on Industrial Informatics*, **8**, 302-310. http://dx.doi.org/10.1109/TII.2011.2174248

[18] Estévez, E., Marcos, M., Iriondo, N. and Orive, D. (2007) Graphical Modeling of PLC-Based Industrial Control Applications. *Proceedings of the* 2007 *American Control Conference*, New York, 11-13 July 2007, 220-225.

[19] Bartels, J. and Vogel, B. (2001) System Engineering Approach for Plant Automation (Systementwicklung für die Automatisierung im Anlagenbau). *At-Automatisierungstechnik*, **49**, 214-224. http://dx.doi.org/10.1524/auto.2001.49.5.214

[20] Land, M. and Horwood, J. (1995) Which Parts of the Road Guide Steering? *Nature*, **377**, 339-340. http://dx.doi.org/10.1038/377339a0

[21] Savioja, P. and Norros, L. (2013) Systems Usability Framework for Evaluating Tools in Safety-Critical Work. *Cognition, Technology & Work*, **15**, 255-275. http://dx.doi.org/10.1007/s10111-012-0224-9

[22] Siau, K. and Rossi, M. (2011) Evaluation Techniques for Systems Analysis and Design Modeling Methods—A Review and Comparative Analysis. *Information Systems Journal*, **21**, 249-268. http://dx.doi.org/10.1111/j.1365-2575.2007.00255.x

[23] Katzke, U., Vogel-Heuser, B. and Fischer, K. (2004) Analysis and State of the Art of Modules in Industrial Automation. *ATP International-Automation Technology in Practice International*, **46**, 23-31.

[24] Nielsen, J. (1993) Usability Engineering. Academic Press, Boston.

[25] Obermeier, M., Schütz, D. and Vogel-Heuser, B. (2012) Evaluation of a Newly Developed Model-Driven PLC Programming Approach for Machine and Plant Automation. 8*th IEEE International Conference on Systems, Man and Cybernetics* (*SMC*), Seoul, 14-17 October 2012, 1552-1557.

[26] Frank, U., Papenfort, J. and Schütz, D. (2011) Real-Time Capable Software Agents on IEC 61131 Systems-Developing a Tool Supported Method. *Proceedings of* 18*th IFAC World Congress*, Milan, 28 August-2 September 2011, 9164-9169.

[27] Strömman, M., Sierla, S. and Koskinen, K. (2005) Control Software Reuse Strategies with IEC 61499. 10*th IEEE International Conference on Emerging Technologies & Factory Automation* (*ETFA*), Catania, 19-22 September 2005, 749-756.

[28] Sierla, S., Christensen, J., Koskinen, K. and Peltola, J. (2007) Educational Approaches for the Industrial Acceptance of IEC 61499. *IEEE International Conference on Emerging Technologies & Factory Automation* (*ETFA*), Patras, 25-28 September 2007, 482-489.

[29] Patig, S. (2008) Preparing Meta-Analysis of Metamodel Understandability. *Workshop on Empirical Studies of Model-Driven Engineering* (*ESMDE* 2008), Toulouse, 29 September 2008, 11-20.

[30] Patig, S. (2008) A Practical Guide to Testing the Understandability of Notations. *Proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling*, Wollongong, **79**, 49-58.

[31] Gemino, A. and Wand, Y. (2004) A Framework for Empirical Evaluation of Conceptual Modeling Techniques. *Requirements Engineering*, **9**, 248-260. http://dx.doi.org/10.1007/s00766-004-0204-6

[32] Vogel-Heuser, B. and Sommer, K. (2011) A Methodological Approach to Evaluate the Benefit and Usability of Different Modeling Notations for Automation Systems. *Proceedings of the 7th IEEE International Conference on Automation Science and Engineering* (*CASE*), Trieste, 24-27 August 2011, 474-481.

[33] Lucas, M.R. and Tilbury, D.M. (2005) Methods of Measuring the Size and Complexity of PLC Programs in Different Logic Control Design Methodologies. *The International Journal of Advanced Manufacturing Technology*, **26**, 436-447. http://dx.doi.org/10.1007/s00170-003-1996-0

[34] Frey, G., Litz, L. and Klöckner, F. (2000) Complexity Metrics for Petri Net Based Logic Control Algorithms. *IEEE International Conference on Systems, Man, and Cybernetics*, Nashville, **2**, 1204-1209.

[35] Venkatesh, K., Zhou, M. and Caudill, R.J. (1994) Comparing Ladder Logic Diagrams and Petri Nets for Sequence Controller Design through a Discrete Manufacturing System. *IEEE Transactions on Industrial Electronics*, **41**, 611-619. http://dx.doi.org/10.1109/41.334578

[36] Lee, J.S. and Hsu, P.L. (2001) A New Approach to Evaluate Ladder Diagrams and Petri Nets via the IF-THEN Transformation. *IEEE International Conference on Systems, Man and Cybernetics*, Tucson, 7-10 October 2001, 2711-2716.

[37] Chidamber, S.R. and Kemerer, C.F. (1994) A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, **20**, 476-493. http://dx.doi.org/10.1109/32.295895

[38] Michura, J. and Capretz, M.A.M. (2005) Metrics Suite for Class Complexity. *IEEE International Conference on Information Technology*: *Coding and Computing* (*ITCC*), Las Vegas, 4-6 April 2005, 404-409.

[39] Fuchs, J., Feldmann, S., Legat, C. and Vogel-Heuser, B. (2014) Identification of Design Patterns for IEC 61131-3 in Machine and Plant Manufacturing. 19*th IFAC World Congress*, Cape Town, 24-29 August 2014, 6092-6097.

[40] Park, E., Tilbury, D.M. and Khargonekar, P.P. (2001) A Modeling and Analysis Methodology for Modular Logic Controllers of Machining Systems Using Petri Net Formalism. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*: *Applications and Reviews*, **31**, 168-188. http://dx.doi.org/10.1109/5326.941841

[41] Fuchs, J. (2011) Analyse und Neukonzeption der Softwaremodularität und deren Abbildung auf die maschinenbauliche Modularität am Beispiel eines Neuglasabschiebers der Lebensmittelindustrie. B.S. Thesis, Faculty of Mechanical engineering, TUM, Munich.

[42] Recker, J.C., zur Muehlen, M., Keng, S., Erickson, J. and Indulska, M. (2009) Measuring Method Complexity: UML versus BPMN. *Proceedings of the* 15*th Americas Conference on Information Systems*, San Francisco, 6-9 August 2009, 1-10.

[43] Rossi, M. and Brinkkemper, S. (1996) Complexity Metrics for Systems Development Methods and Techniques. *Information Systems*, **21**, 209-227. http://dx.doi.org/10.1016/0306-4379(96)00012-9

[44] Schalles, C. (2012) Usability Evaluation of Modeling Languages. Ph.D. Dissertation, Department of Computing, CIT, Cork.

[45] Frank, T. (2014) Entwicklung und Evaluation einer Modellierungssprache für den Architekturentwurf von verteilten Automatisierungsanlagen auf Basis der Systems Modeling Language (SysML). Ph.D. Dissertation, Institute of Automation and Information Systems, Technical University Munich, Munich.

[46] Hajarnavis, V. and Young, K. (2008) An Assessment of PLC Software Structure Suitability for the Support of Flexible Manufacturing Processes. *IEEE Transactions on Automation Science and Engineering*, **5**, 641-650. http://dx.doi.org/10.1109/TASE.2007.917135

[47] Cross, J. and Denning, P. (2001) Computing Curriculum 2001. The Joint Curriculum Task Force IEEE-CS/ACM Report. http://www.acm.org/education/curric_vols/cc2001.pdf

[48] Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C. and Verno, A. (2003) A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee. The Association for Computing Machinery, New York.

[49] Fitzgerald, S., McCauley, R., Hanks, B., Murphy, L., Simon, B. and Zander, C. (2010) Debugging from the Student Perspective. *IEEE Transactions on Education*, **53**, 390-396. http://dx.doi.org/10.1109/TE.2009.2025266

[50] Curtis, B. (1988) Five Paradigms in the Psychology of Programming. In: Helander, M., Ed., *Handbook of Human-Computer Interaction*, Elsevier North Holland, Amsterdam, 87-105. http://dx.doi.org/10.1016/B978-0-444-70536-5.50009-9

[51] Magenheim, J., Nelles, W., Rhode, T., Schaper, N., Schubert, S. and Stechert, P. (2010) Competencies for Informatics Systems and Modeling: Results of Qualitative Content Analysis of Expert Interviews. *IEEE Education Engineering*,

Madrid, 14-16 April 2010, 513-521.

[52] Ruocco, A.S. (2003) Experiences in Threading UML throughout a Computer Science Program. *IEEE Transactions on Education*, **46**, 226-228. http://dx.doi.org/10.1109/TE.2002.808263

[53] Wickens, C.D. and Hollands, J.G. (2000) Engineering Psychology and Human Performance. 3rd Edition, Prentice Hall, Upper Saddle River.

[54] Schweizer, K., Gramß, D., Mühlhausen, S. and Vogel-Heuser, B. (2009) Mental Models in Process Visualization—Could They Indicate the Effectiveness of an Operator's Training? *Engineering Psychology and Cognitive Ergonomics*, Springer, Berlin, Heidelberg, 297-306.

[55] Gravetter, F.J. and Wallnau, L.B. (2006) Statistics for the Behavioral Sciences. Thomson/Wadsworth, Belmont.

[56] Kim, J. and Lerch, F.J. (1992) Towards a Model of Cognitive Process in Logical Design: Comparing Object-Oriented and Traditional Functional Decomposition Software Methodologies. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Monterey, 3-7 July 1992, 489-498.

[57] Kim, S.H. and Jeon, J.W. (2009) Introduction for Freshmen to Embedded Systems Using LEGO Mindstorms. *IEEE Transactions on Education*, **52**, 99-108. http://dx.doi.org/10.1109/TE.2008.919809

[58] Berges, M. and Hubwieser, P. (2011) Minimally Invasive Programming Courses: Learning OOP with (out) Instruction. *Proceedings of the* 42*nd ACM Technical Symposium on Computer Science Education*, Dallas, 9-12 March 2011, 7-92.

[59] Faux, R. (2006) Impact of Preprogramming Course Curriculum on Learning in the First Programming Course. *IEEE Transactions on Education*, **49**, 11-15. http://dx.doi.org/10.1109/TE.2005.852593

[60] Jacobson, M.L., Said, R.A. and Rehman, H. (2006) Introducing Design Skills at the Freshman Level: Structured Design Experience. *IEEE Transactions on Education*, **49**, 247-253. http://dx.doi.org/10.1109/TE.2006.872403

[61] Verginis, I., Gogoulou, A., Gouli, E., Boubouka, M. and Grigoriadou, M. (2001) Enhancing Learning in Introductory Computer Science Courses through SCALE: An Empirical Study. *IEEE Transactions on Education*, **54**, 1-13. http://dx.doi.org/10.1109/TE.2010.2040477

[62] Lahtinen, E. (2007) A Categorization of Novice Programmers: A Cluster Analysis Study. *Proceedings of the* 19*th Annual Workshop of the Psychology of Programming Interest Group*, Joensuu, 2-6 July 2007, 32-41.

[63] Vogel-Heuser, B., Obermeier, M., Braun, S., Sommer, K., Jobst, F. and Schweizer, K. (2013) Evaluation of a UML-Based versus an IEC 61131-3-Based Software Engineering Approach for Teaching PLC Programming. *IEEE Transactions on Education*, **56**, 329-335. http://dx.doi.org/10.1109/TE.2012.2226035

[64] International Organization for Standardization (1999) Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs)-Part 11: Guidance on Usability, EN ISO 9241-11:1998. Beuth, Berlin.

[65] Bevan, N. (1995) Measuring Usability as Quality of Use. *Software Quality Journal*, **4**, 115-130. http://dx.doi.org/10.1007/BF00402715

[66] Annett, J. (2003) Hierarchical Task Analysis. In: Hollnagel, E., Ed., *Handbook of Cognitive Task Design*, Lawrence Erlbaum Assoc. Inc., Mahwah, 17-35.

[67] Lucas, M.R. and Tilbury, D.M. (2002) Quantitative and Qualitative Comparisons of PLC Programs for a Small Testbed with a Focus on Human Issues. *Proceedings of the American Control Conference*, Anchorage, **5**, 4165-4171.

[68] Lucas, M.R. (2003) Understanding and Assessing Logic Control Design Methodologies. Ph.D. Dissertation, Dept. Mechanical Eng., University of Michigan, Ann Arbor.

[69] Friedrich, D. and Vogel-Heuser, B. (2007) Benefit of System Modeling in Automation and Control Education. *American Control Conference* (*ACC*), New York, 9-13 July 2007, 2497-2502.

[70] Friedrich, D. (2009) Anwendbarkeit von Methoden und Werkzeugen des konventionellen Softwareengineerings zur Modellierung und Programmierung von Steuerungssystemen. Ph.D. Dissertation, University Kassel, Kassel.

[71] Friedrich, D. and Vogel-Heuser, B. (2005) Evaluating the Benefit of Modeling Notations on the Quality of PLC-Programming. 11*th International Conference on Human-Computer Interaction* (HCI), Las Vegas, 22-27 July 2005.

[72] Katzke, U. and Vogel-Heuser, B. (2005) UML-PA as an Engineering Model for Distributed Process Automation. *IFAC World Conference*, Prague, 3-8 July 2005, 129-134.

[73] Katzke, U. and Vogel-Heuser, B. (2009) Vergleich der Anwendbarkeit von UML und UML-PA in der anlagennahen Softwareentwicklung der Automatisierungstechnik-Beispiel einer vergleichenden empirischen Untersuchung von Modellierungssprachen. *at-Automatisierungstechnik*, **57**, 332-340. http://dx.doi.org/10.1524/auto.2009.0781

[74] Witsch, D., Ricken, M., Kormann, B. and Vogel-Heuser, B. (2010) PLC-Statecharts: An Approach to Integrate UML-Statecharts in Open-Loop Control Engineering. 8*th IEEE International Conference on Industrial Informatics* (*INDIN*), Osaka, 13-16 July 2010, 915-920.

[75] Witsch, D. and Vogel-Heuser, B. (2009) Close Integration between UML and IEC 61131-3: New Possibilities through

Object-Oriented Extensions. *IEEE International Conference on Emerging Technologies & Factory Automation* (*ETFA*), Mallorca, 22-26 September 2009, 1-6.

[76] Yang, S. and Sun, J.L. (2010) Modeling Traverse Feature in Concurrent Software System with UML Statecharts. *IEEE International Conference on Computational Intelligence and Software Engineering* (*CiSE*), Wuhan, 10-12 December 2010, 133-138.

[77] Witsch, D. (2012) Modellgetriebene Entwicklung von Steuerungssoftware auf Basis der UML unter Berücksichtigung der domänenspezifischen Anforderungen des Maschinen-und Anlagenbaus. Ph.D. Dissertation, Faculty of Mechanical Engineering, TUM, Munich.

[78] Vogel-Heuser, B., Seidel, T., Braun, S., Obermeier, M., Sommer, K. and Johannes, C. (2011) Modeling Order Effects on Errors in Object Oriented Modeling for Machine and Plant Automation from an Educational Point of View. 16*th IEEE International Conference on Emerging Technologies & Factory Automation* (*ETFA*), Toulouse, 5-9 September 2011.

[79] Frank, T., Eckert, K., Hadlich, T., Fay, A., Diederich, C. and Vogel-Heuser, B. (2012) Workflow and Decision Support for the Design of Distributed Automation Systems. *IEEE INDIN*: *International Conference on Industrial Informatics*, Beijing, 25-27 July 2012, 293-299. http://dx.doi.org/10.1109/INDIN.2012.6300859

[80] Frank, T., Hadlich, T., Eckert, K., Fay, A., Diedrich, C. and Vogel-Heuser, B. (2012) Using Contact Points to Integrate Discipline Spanning Real-Time Requirements in Modeling Networked Automation Systems for Manufacturing Systems. *IEEE International Conference on Automation Science and Engineering* (*CASE*), Seoul, 20-24 August 2012, 851-856.

[81] Frank, T., Eckert, K., Hadlich, T., Fay, A., Diederich, C. and Vogel-Heuser, B. (2013) Erweiterung des V-Modells® für den Entwurf von verteilten Automatisierungssystemen. *At-Automatisierungstechnik*, **61**, 79-91. http://dx.doi.org/10.1524/auto.2013.0009

[82] Eckert, K., Hadlich, T., Fay, A., Diederich, C. and Vogel-Heuser, B. (2012) Design Patterns for Distributed Automation Systems with Consideration of Non-Functional Requirements. 17*th IEEE International Conference on Emerging Technologies & Factory Automation* (*ETFA*), Krakow, 17-21 September 2012, 1-9.

[83] Oswald, W.D. and Roth, E. (1978) Der Zahlen-Verbindungs-Test (ZVT). Ein sprachfreier Intelligenz-Schnell-Test. Verlag für Psychologie Horgrefe, Göttingen.

[84] Hart, S. and Staveland, L. (1988) Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In: Hancock, P. and Meshkati, N., Eds., *Human Mental Workload*, North Holland, Amsterdam, 139-183.

[85] Vogel-Heuser, B., Braun, S., Kormann, B. and Friedrich, D. (2011) Implementation and Evaluation of UML as Modeling Notation in Object Oriented Software Engineering for Machine and Plant Automation. 18*th World Congress of International Federation of Automation Control* (*IFAC*), Milan, 28 August-2 September 2011, 9151-9157.

[86] Ko, A.J. and Myers, B.A. (2005) A Framework and Methodology for Studying the Causes of Software Errors in Programming Systems. *Journal of Visual Languages and Computing*, **16**, 41-84. http://dx.doi.org/10.1016/j.jvlc.2004.08.003

[87] McCabe, T. (1976) A Complexity Measure. *IEEE Transactions on Software Engineering*, **2**, 308-320. http://dx.doi.org/10.1109/TSE.1976.233837

## Appendix A. Example of Subject's UML Model (Evaluation Scheme E1)

One example of a modular UML model (**Figure A1**) is given. The results show the subject's problem to identify reusable parts of the plant. He decided to use the state chart (**Figure A1**, left) and the class diagram (**Figure A1**, right). He tried to build a modular state chart and formed a class "single out and transport to stamp". This shows the inadequate understanding of classes and state charts and their application. Unfortunately, none of the subjects realized a correct class model.

## Appendix B. (Evaluation Scheme E5 and WMC Calculation)

### B.1. Evaluation Scheme E

In the following the measurement from E5 for plcUML and FBD model quality is shown. First the evaluated model elements are discussed for structure and behavior modeling. Then typical subject's solutions for both notations are shown and the points given are depicted.

The model quality regarding the grade of task completion (correct model elements) for both notations was evaluated for structure and behavior through manual code/model inspection. As measure for structure in plcUML models, the number of correct attributes with a correct data type and the correct access modifier in the class diagram was counted (cf. **Figure B1**). As methods were not imperative in order to solve the given task, they were not included in the measurement.

Additionally the created object instances in the main program were counted for the structure model. *i.e.* each correct instantiation of a cylinder object was counted (cf. **Figure B2**).

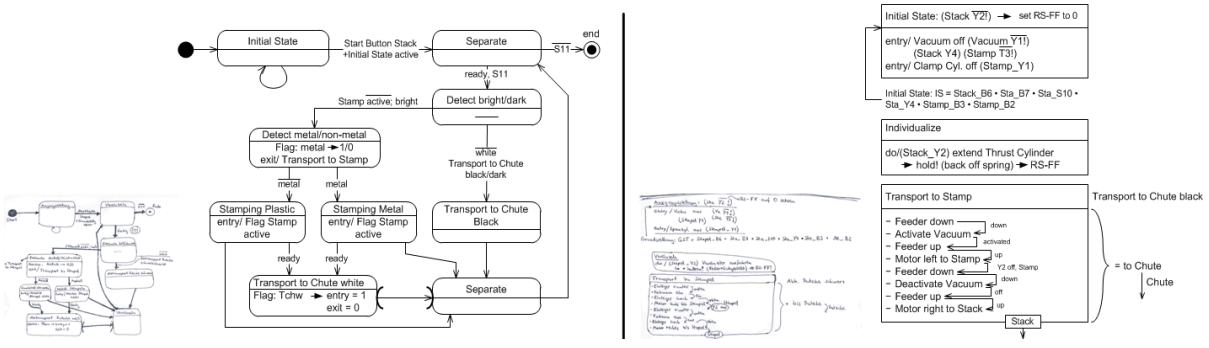


**Figure A1.** Modular UML behavior model of one subject (left: hand written model; right: translated model).

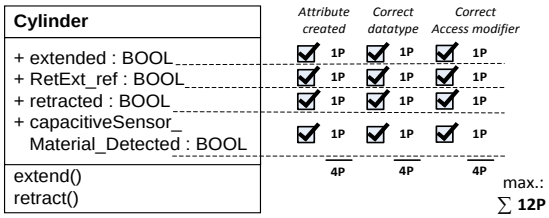


**Figure B1.** Structure model quality measurement plcUML: Class diagram.

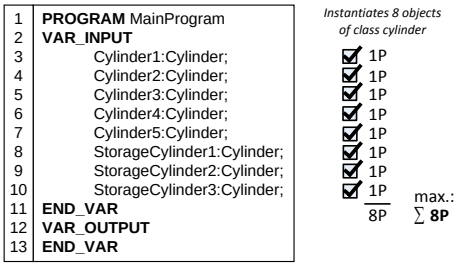


**Figure B2.** Structure model quality measurement plcUML: Object instantiation.

This results in a maximum of 20 points available for the structure model in plcUML. The plcUML behavior model quality was measured by identifying correct method calls, sequences of variable comparisons and states (cf. **Figure B3**). If the subsequent state after a logically correct variable comparison included a logically correct method call an additional point was given.

In **Figure B4(a)** and **Figure B4(b)** a complete example measurement for one student's model in UML is shown. Missing Points are depicted as Xs. The quality of the structure model (**Figure B4(a)**) is 13/20 or 65% and quality of the behavior model 24/67 or 35.82% (**Figure B4(b)**). The overall model quality is 37/87 or 42.53%.

Similar to the plcUML model quality measurement, the FBD program quality was evaluated. For the structure quality every necessary in- and output for the FBs was counted, cf. **Figure B5**. This results in a maximum of 32

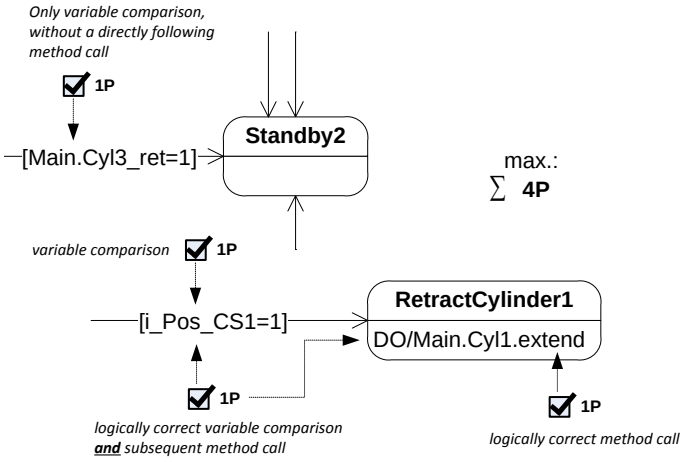


**Figure B3.** Behavior model quality measurement plcUML.

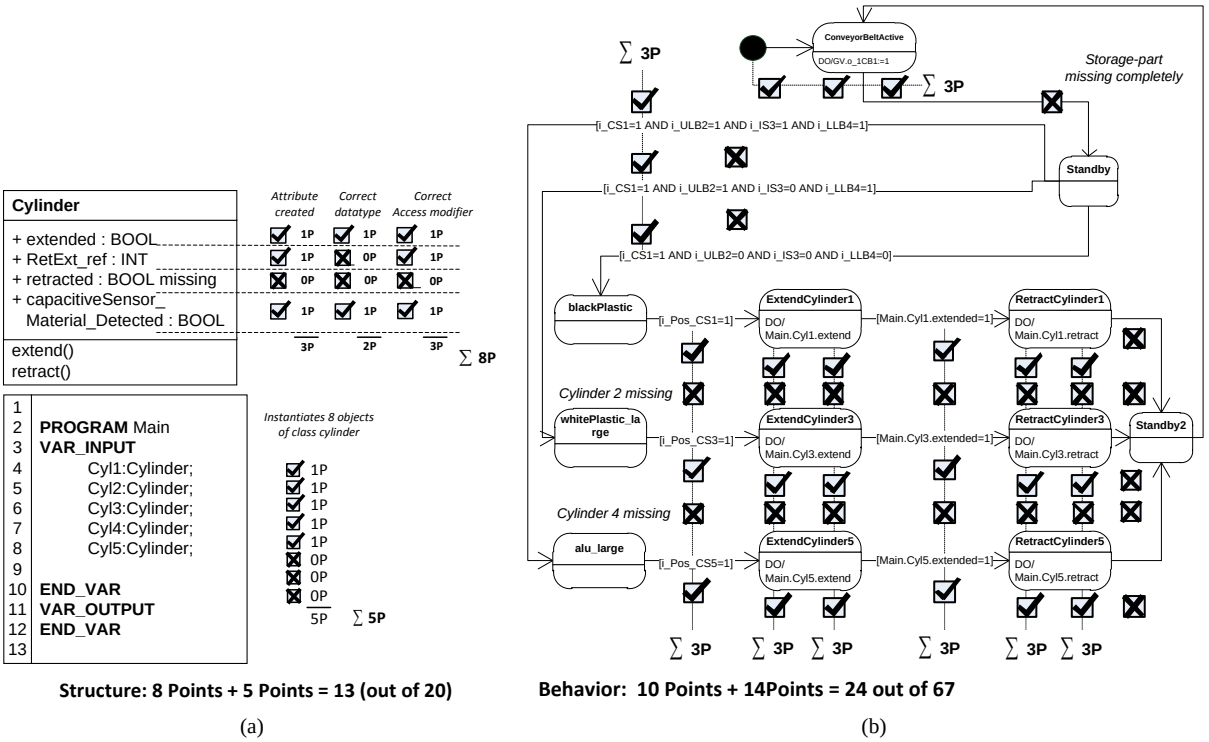


**Figure B4.** (a) plcUML structure model quality: 13/20 points; (b) plcUML behavior model example: model quality 47/67. Overall model quality: 13 + 24 = 37 Points/Relative overall modeling performance: 37/87 = 42.53%.

points for FBD model structure quality.

The FBD behavior model quality was measured by identifying correct FB or FC calls, sequences of variable comparisons and the connection of these elements, cf. **Figure B6**. If the subsequent call after a logically correct variable comparison included a logically correct FB or FC call an additional point was given.

## B.2. WMC Calculation

WMC is defined as the sum of Ci. Ci is the cyclomatic complexity of the ith Method, and is calculated by counting the conditions of the method +1, cf. McCabe 1976 [87]. In **Figure B7** an example for WMC calculation is given. In this case only the methods auto and manual of the example class are relevant. The auto method contains several conditions and therefore has a cyclomatic complexity corresponding to the number of included conditions +1 as defined by McCabe, resulting in Cauto = 10. The manual method does not contain any conditions resulting in a cyclomatic complexity Cmanual of 1. Finally these two complexity values sum up to an

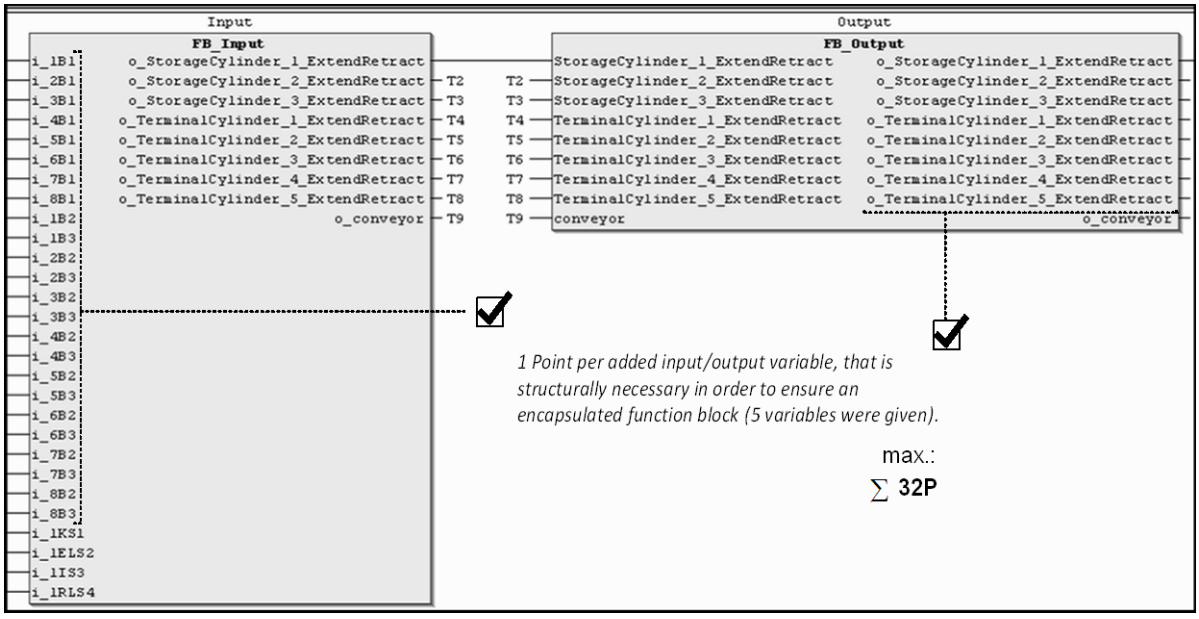


**Figure B5.** Structure model quality measurement FBD.

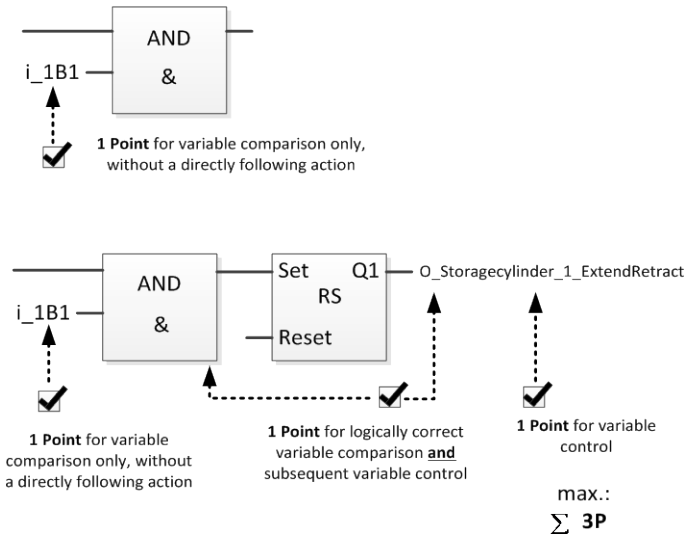


**Figure B6.** Behavior model quality measurement FBD.

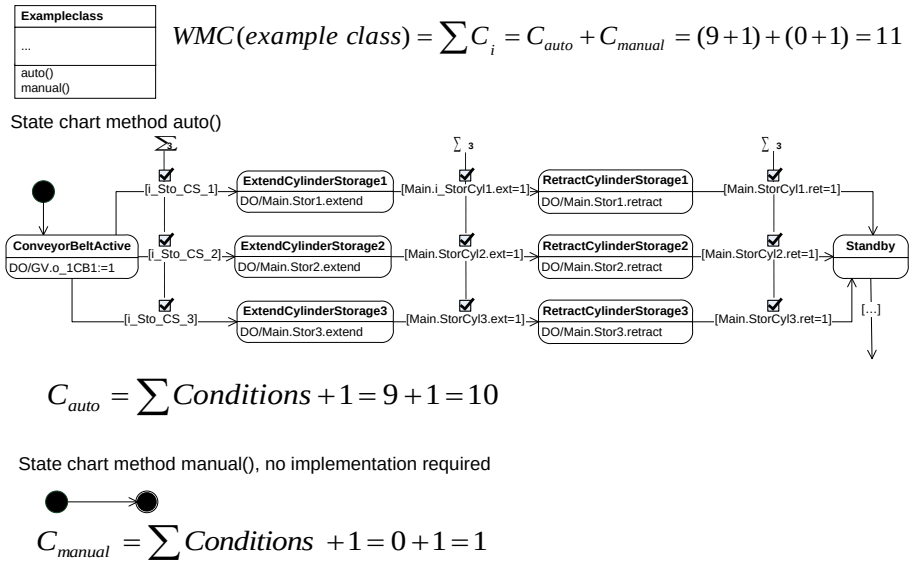


**Figure B7.** WMC calculation example.

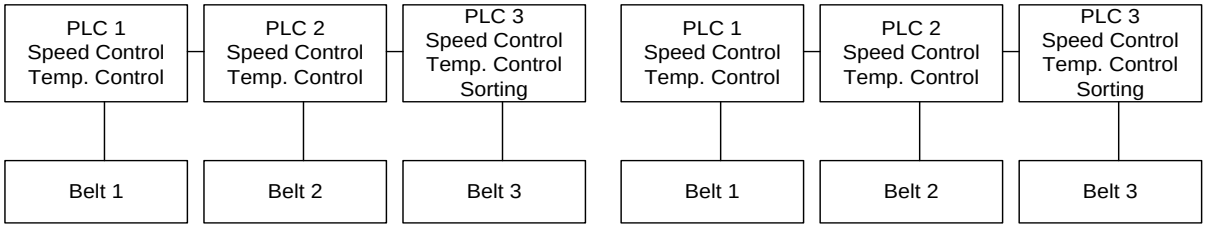


**Figure C1.** Most subject's functional oriented model (left) and Master model-mechatronic oriented model (right).

overall WMC of 11 for the example class.

## Appendix C. Master Model and Subjects' Solution (Evaluation Scheme E7)

Most subjects in E7 chose a functional oriented model deployment (**Figure C1**, left), *i.e.* deploying different functions (speed control, temperature control and sorting) on different PLCs, which is from an architectural perspective considering mechatronic modularity and reuse an inappropriate solution. NAS experts chose a different approach (**Figure C1**, right) to support reuse of existing modules. A comparison and evaluation of these different models was difficult as "how to build a module" was not included in the training, because the MDE approach should be intuitively applicable.

The best subject gained 144 out of 182 points building a mechatronic oriented model, but neglected requirements and other details.

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or Online Submission Portal.