Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Implementation and evaluation of UML as modeling notation in object oriented software engineering for machine and plant automation

Birgit Vogel-Heuser* Steven Braun* Benjamin Kormann*
David Friedrich**

** Institute of Automation and Information Systems,*
Technische Universität München, 85748 Garching, Germany
(e-mail: {vogel-heuser, braun, kormann}@ais.mw.tum.de)
*** Dept. of Design, Prototyping and Standards,*
Siempelkamp GmbH, 47803 Krefeld, Germany
(e-mail: david.friedrich@siempelkamp.com)

Abstract: Our goal is to increase efficiency and quality in automation engineering in machine and plant manufacturing industry by supporting modularity and reuse. This article proofs that object-oriented model-based design can beneficially be applied in industry and that the code automatically derived from the UML model can be implemented on industrial PLCs without additional effort. We had to solve the formal mapping from UML models to IEC 61131-3 program code in order to use an object-oriented approach with Unified Modeling Language (UML) as modeling notation integrated into a classical Programmable Logic Controller (PLC) programming environment (IEC 61131-3).

1. INTRODUCTION

This paper answers the following questions: Is an integrated UML modeling approach useful and beneficial for software engineering in automation in terms of software quality and development efficiency? What are the domain specific characteristics in UML diagrams and UML semantics needed to achieve an increased software quality and efficiency? We have used an object-oriented approach with Unified Modeling Language (UML) as modeling notation integrated into a Programmable Logic Controller (PLC) programming environment IEC 61131-3. We mainly focus on automation software development for machine and plant manufacturing industry. The IEC 61131-3, respective IEC 1131-3 by Bonfatti et al. (1997), with its five languages (Structured Text, Sequential Function Chart, Ladder Diagram, Function Block Diagram and Instruction List) is still accepted and applied as a worldwide industry standard in automation software engineering. Regarding the study of ARC Advisory Group (2009), the impact of non-IEC programming environments on embedded controllers is still minimal. This paper will briefly discuss the state of the art in automation software engineering for machine and plant manufacturing industry, the tools used and the applied experimental evaluation methods. The applicability of UML as notation will be discussed on three evaluation experiments of UML in PLC programming. Therefore we derived the concept and selected diagrams of our UML approach which are supported by the often used IEC 61131-3 engineering environment CoDeSys 3.0. The UML integration concept and two diagrams will be exemplarily introduced with respect to their implementation in code. The application of UML embedded into IEC 61131-3 will be evaluated, firstly by industrial companies as a beta-test and secondly in a lab experiment with students. Finally we will summarize the lessons learned during development and application in industry and estimate the necessary effort along with the available benefit for automation in machine and plant manufacturing. We conclude with necessary further development steps.

2. STATE OF THE ART

2.1 SW Engineering in Machine and Plant Automation

Modeling is barely applied in machine and plant automation, especially for hybrid processes, i.e. discrete and continuous process with a high degree of individual machines and plants. Though physical devices are modeled to ease start-up by simulation prior to runtime like proposed by Kain et al. (2009) or process modeling in process industry, but currently reuse in automation software is mostly applied on function block level, see Katzke et al. (2004) and Vogel-Heuser (2009a). The application engineer selects reusable elements, i.e. function blocks out of a library.

In Estévez, et al. (2007) a methodology for designing an industrial control system is detailed based on a management of model collaboration. This approach aims at achieving the exchange of information among the tools in order to support the development cycle of industrial control applications. Heverhagen et al. (2001) proposes the use of real-time Unified Modeling Language (UML) for defining IEC 61131-3 function blocks having code mapping as a main goal. Dietrich et al. (2006) applied UML in software development considering safety constraints for testing purposes of PROFIsafe. Licht (2004) proposed the integration of timed automata to verify the real time behavior. Secchi et al. (2007) introduce a unified framework for physical system and software modeling with the use of UML-RT profile.

This result is expanded in our project requirements analysis and by interviews with industry experts in Vogel-Heuser

(2009b). They all rate modularity and version management as well as change management as key requirements for successful engineering. The object-oriented paradigm has led to great success in application development, because it addresses the needs of concepts to handle today's complexity in software development and outperforms other approaches. The object-oriented extension of the IEC 61131-3 specification with classes and methods, interfaces and inheritance seems to emerge as a promising solution.

However this extension towards object-orientation does not include event mechanisms (as commonly used in Java, C#), but yet relies on synchronous program processing. It is not intended to replace the five well established IEC languages, because, among other reasons, it might discourage control engineers since support of legacy code would not be ensured. The cyclic execution pattern of PLCs still applies. In contrast, IEC 61499 in Sunder et al. (2006) follows an event-based paradigm with asynchronous communication and program execution. These differences regarding communication and execution mechanisms constitute (still) the main disparity between IEC 61499 and IEC 61131-3.

2.2 UML and Tool Support in Automation

The research groups Giese and Henkler (2006), Burmester et al. (2005), Gehrke (2005) included in SFB 614 adapted UML to mechatronic tasks and to real time requirements. A code generation to IEC 61131-3 ST is prototypically implemented, but not integrated into an IEC 61131-3 environment. A bi-directional mapping between the UML model and the generated code is an essential prerequisite for executable UML models in automation. It ensures consistency, keeps track of changes at plant site, and avoids a drift between model and code, since currently tests and improvements are mainly done at startup time. At the beginning we used IEC 61131-3 code generated out of UML models without any bi-directional mapping. But from an industrial point of view it is mandatory to edit during start-up, operation and maintenance as well as during redesign (see chapter 2.2). We showed that only a close integration of UML model and PLC code allows modification of code on plant site without the risk of inconsistency between model and code, see Witsch et al. (2009).

2.3 Usability Requirements

Industrial projects, their experts, as well as a lab experiments may be included to evaluate the benefits in terms of quality and time reduction based on criteria for the usability evaluation of programming notations and programming languages in lab experiments proposed by Gemino and Wand (2005) and Patig (2008). Katzke (2008) and Friedrich (2009) define an appropriate task to evaluate and include the user group, i.e. skilled workers or technicians for operation and maintenance in automation software engineering, see Vogel-Heuser (2009a).

3. UML AS NOTATION IN AUTOMATION SOFTWARE ENGINEERING

The UML (current version 2.3) is an established standard for software development. It is specified by the Object Management Group (OMG) and available as ISO/IEC 19501. UML can be used to create models from a software perspective or on a conceptual basis. The latter use of UML can be applied to almost any problem domain. Diagrams, such as use case diagrams are introduced for discussions to get a clear view of a system to be modeled and to enable the retrieval of requirements. Later on such diagrams serve as documentation and ease understanding. A major drawback in using UML sketches is the risk of becoming inconsistent with each other. If the UML is used from a software perspective, object-oriented features can be used for model construction. The complete specification consists of 16 different diagram types for structural and behavioral description (in version 2.3). In this chapter we focus on the usability requirements of a modeling notation.

3.1 Requirements from PLC Programming as a Task

A more detailed analysis of PLC programming as a task is helpful to derive the requirements for an appropriate UML-based support and its integration into an IEC 61131-3 engineering tool. We focus on hybrid processes in the production plant, which have specific constraints regarding qualification of personnel, necessity of program changes during run time as well as usually applied modularity and reuse concepts. Katzke et al. (2004) found three sizes of modules in industry, i.e. basic modules, application modules and plant modules and realized a wide spread type of reuse in plant manufacturing copy & paste, copy & modify and growing in the last years building of variants by parameters. To derive modules from a universal module and to use inheritance is still in its beginning.

3.2 UML in Automation -UML-PA Profile

Based on task analysis, Vogel-Heuser et al. (2005) and Friedrich (2009) conducted a series of controlled experiments to evaluate the benefit of modeling as such, and more specifically of UML for process modeling and subsequent PLC-programming. Two approaches based on the UML and the Idiomatic Control Language (ICL) got compared to the typical procedure of PLC programming without any notational support. One of the main results stated by the subjects was the fact, that no procedural method for the modeling task was provided. Subjects felt confused by the number of possible diagrams provided by the UML and were insecure about the correct sequence in modeling. Additionally, most of the subjects criticized the lack of tool support for the modeling task. They felt the use of pen and paper for the modeling task wasn't flexible enough. The transfer from the model to the program was not clear from a programming task point of view. Subjects proposed an automatic PLC code generation in case the model was created with a software tool.

UML-PA specified in Katzke (2008) and described in Katzke and Vogel-Heuser (2005) is, as the UML, composed out of various sub-languages. These languages describe different structures, interactions and behaviors. By selecting specific language elements, from approaches in the software development, the UML-PA is a customized modeling language for automation and was developed in DiSPA by Fischer et al. (2004). The applicability of a modeling language

in its practical use is as important as its formal specification. The UML-PA was designed at lower complexity with fewer types of diagrams and modeling elements. Software engineers using UML-PA should:

- decide faster for the use of a specific diagram type
- define the links between software objects and sensors/actuators of a system faster and easier
- summarize typical characteristics in objects to a class definition faster
- create behavioral descriptions out of interactions and scenarios faster and more precisely
- Determine the complete software system behavior out of the specific object behavior in individual scenarios.

3.3 Detailed Requirements from Industry

Experiments showed the necessity of a domain specific UML and a restricted number of diagrams, as well as support of a tool and a methodology. Support of legacy code is strongly required. Many lines of code exist in manufacturing and plant automation, which have already been tested. This legacy software often constitutes valuable know-how and needs to be reused due to economic and quality reasons. An advantage of its implementation is the ability to present the software structure in an intuitive way, importing and visualizing existing IEC 61131-3 code in the form of UML class diagrams should be possible (as a minimal requirement).

An implementation decoupled access of components is the fundamental basis for exchangeability on code level, e.g. a temperature sensor provides (aside from parameterization) a single method, which returns the measured temperature value. This way such a sensor can easily be replaced by any other interface compliant component without any modification on the method calling software. A component accessible by a standardized interface enables the possibility of maintainable software throughout the plant or machine lifecycle even if components must be replaced because of a technology change or maintenance. Diehm (2008) states the following central disadvantages of SFC lead to the requirements on UML state charts:

- Each SFC-Step iteration leads to a PLC cycle change. Thus detailed sequences cannot be modeled in SFC, even though it is based on a state automaton.
- Pseudo states must be inserted to jump before parallel sequences.
- Error handling routines lead to complex transition conditions.
- It is often necessary to insert jumps in order to realize a particular algorithm, which decreases readability.

The aspect of documentation during the project and support for communication between different disciplines was revealed as an important requirement.

4. UML EMBEDDED IN IEC 61131 ENVIRONMENT

Control engineers in automation must cope with challenges in all phases throughout development of control systems. Since existing technologies and paradigms are limited in their effectiveness, a new approach is needed for a higher productivity in modularity and reuse of software components other than on a function block level. Therefore the object-oriented features are interwoven with all other already existing language elements. The UML can even be made executable and used as a programming language. With the means of a model compiler, UML models become executable. The UML specification does not provide a formal definition on how model elements must be mapped into object-oriented programming languages, neither for discrete event systems nor for cyclic execution systems, such as PLCs. Executable UML models must be mature and complete for successful and productive usage and most importantly acceptance. Our UML editor is embedded into CoDeSys 3.0, the reference implementation of the object-oriented extension of IEC 61131-3. The editor supports three diagram types as there are class diagrams for structural description, state chart and activity diagram for behavioral description. All these diagrams are specified in a complete manner by Witsch et al. (2009, 2010). The editor provides modeling, coding and online debugging of UML models, where object-oriented elements can be mixed with traditional IEC language elements. Since the debugging functionality is entirely integrated in the IEC 61131-3 environment, the implementation can be monitored during runtime.

The class diagram enables a completely novel view of the control code structure. Class diagrams visualize the dependency and connection of components. A bi-directional mapping between the UML model and the generated code is integrated into the UML editor. The object-oriented paradigm encourages a modular design by different relation types, such as association among classes and inheritance for hierarchical structuring. The UML class diagram illustrates the structure of automation control software and helps to understand the interdependencies of components. Each of them has its own individual internal behavior. Detailed information on UML diagrams are not within the scope of this paper, but can be obtained in Witsch et al. (2008).

More complex design patterns like those already applied in classical application development have not yet been elaborated in automation control systems. It will be an important future research project to derive such patterns to reduce development time and cost while getting higher software quality. The object-oriented paradigm states that each instance of a component (as known as object) encapsulates state and behavior. UML state charts pick up that concept and provide a view to an implementation of an object in the UML editor. The UML state charts for industrial PLCs like introduced in Witsch et al. (2010) are formally specified as follows:

A PLC-state chart is a 6 tuple $SM = (S, T, \tau, O, s_0, s_f)$, where

- $S = \{S_1, \dots, S_n\}$ is a nonempty, finite set of states $S_i = (u, m)$ with
  - a time information $u \in \mathbb{N}$ and
  - $m \in \mathbb{Z}_2$ execution within a PLC-cycle (true, false)
- $T = (C, P, A, e)$ is a nonempty, finite set of transitions with
  - $C : V \longrightarrow \mathbb{Z}_2$ conditions over the set of variables V

- $P = \mathbb{N}$ is the priority of the transition regarding its source state,
- $A : 2^V \longrightarrow V$ actions
- $e \in \mathbb{Z}_2$ execution within a PLC-cycle (true, false)

• $\tau : (S \times T) \longrightarrow S$ is the transition function
• $O = \{(SM_{11}, \dots, SM_{1n}), \dots, (SM_{m1}, \dots, SM_{mn})\}$, $n > 1$ is a finite set of ordered orthogonal states with: $\forall i, j \in \{1, \dots, n\}, i \neq j \mid SM_{ik} \cap SM_{jk} = \emptyset$ with $SM_{ik} = (S_i, T_i, \tau, \emptyset, s_{i_0}, s_{i_f})$
• $s_0 \in S$ is the initial state
• $s_f \in S$ is the final state

Let $s_i \in S$ be an active state and let $T_i \subset T$ be all outgoing transitions of s_i, the transition function $\tau(s_i, t_i) = s_j$ fires, iff $t_i \in \left(T_{true} = \left\{t_j \in T_i \middle| c_{t_j} = true\right\}\right) \wedge \nexists t_k \in T_{true} \middle| p_{t_k} < p_{t_i}$ with $p_{t_l} \in P$.

State chart diagram is similar to IEC 61131-3 Sequential Function Chart, but more powerful with its modeling elements. Some of the main distinction between UML state chart and SFC are the missing support for failure concepts in SFC where state charts provide exception transitions out of composite-states for hierarchical state structures. Additionally SFC cannot be used for method implementations of the object oriented extension of the IEC 61131-3 standard and cyclic changes can only be expressed after each step in SFC whereas state charts provide so called fast states being executed in a single cyclic execution. A major improvement from a usability point of view is the totally freedom of state and transition placement in state charts. States can have multiple transition conditions and states can also be organized hierarchically. State machines are valid in the scope of objects. An object can take in exactly one state at a given point in time and it is able to transit to another state by method calls. The different behavior of the components is realized by method calls in the context of the corresponding object causing state transitions. A behavioral description is also needed for the interaction of components (objects). UML activity diagrams within the UML editor allow the specification of sequential and parallel tasks with Petri nets semantics. Each activity can be considered as real tasks totally independent to the PLC typical cyclic execution time, such as move out cylinder to its end position, see Witsch and Vogel-Heuser (2009).

5. APPLICATION EXAMPLE

The considered modular application example consists of a sorting belt, a stack depot, a crane and a stamp module (Fig. 1).

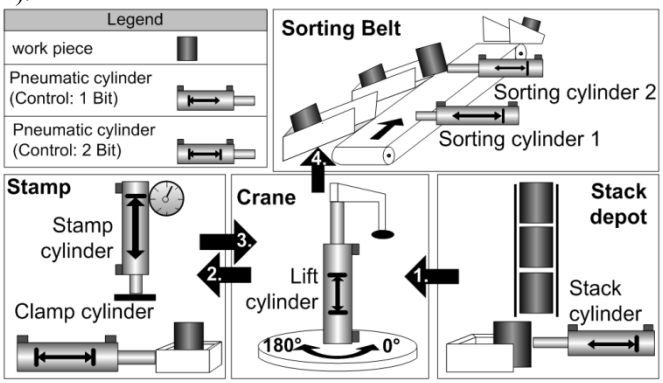

Fig. 1. Schematic illustration of the stamp and sort plant

In step 1 a work piece (WP) gets pushed from a stack depot into the handover position. The crane grabs the WP at 0°. The WP is handed over to the stamp at 180° in step 2. It clamps the work piece, stamps it with a configurable pressure and releases it afterwards. In step 3 the crane transports WP to the sorting belt at 90°. WP is handed over to the sorting belt and gets transported to an available (free) slide, where a cylinder pushes the work piece off the belt in the final step 4. All these components represent a larger functional module within the context of application.

There are six pneumatic cylinders, which are present in three different variants. As there are single acting (mono), double acting and controlled single acting cylinders. The modular composition of all components is directly mapped onto the object-oriented software structure. A module of the technical system equates to a class in the object-oriented environment. This picks up the concept of a mechatronic module from a software development point of view. Fig. 2 shows the realization of the work piece being pushed from a stack depot into handover position of the application example following the formal PLC state chart specification mentioned earlier.

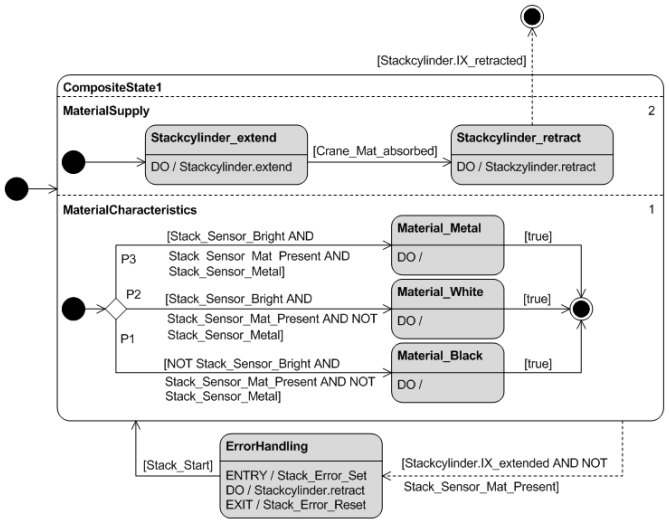

Fig. 2. State chart based implementation of material supply task

The outer left solid black state is considered as the starting state, whereas the upper framed solid black state is considered as the final state. The edges can optionally carry conditions, which if present need to be evaluated to true before they become active. The composite state enables a hierarchical grouping of states. In this particular example the lower region within the composite state is executed before the upper one, because it has a lower ordering number noted on the right hand side. The degree of all outgoing edges on the decision point is three, where the conditions are evaluated in the order of their priority from high to low (P1 highest). The transition (evaluated in order) which evaluates to true first will fire first. Each state can have optional entry, do and exit actions. A do action can run multiple times over multiple cycle times of the PLC whereas entry as well as exit actions are executed only once at the entrance and exit. This becomes clear in the formal definition of state charts, because the specific entry and exit actions are part of the state transition function T, hence these

actions are executed only once, either when leaving or entering a state. Dotted edges leaving the composite state are able to fire at any time. This is an important modeling technique for e.g. error handling, wherever necessary within the hierarchy. This particular state chart in figure 2 checks the material characteristics first and switches to the corresponding state in the lower region. It transits to the upper region of the composite state and starts the material supply part of the implementation

6. EVALUATION OF UML IN MDE FOR AUTOMATION

In this chapter the application of UML embedded into IEC 61131-3 will be evaluated by industrial companies as a beta-test and in a lab experiment with students.

6.1 Evaluation by Industry Application Experts

Our embedded UML concept and the UML editor were evaluated on different prototype states by the project consortium, but also by the companies of the industrial advisory board and others, i.e. externally (outside of the consortium) by application experts in the automation control industry, i.e. Bosch and HAUNI. BOSCH Atmo (Packaging Automation) did an analysis on the integration of the object-oriented features in IEC 61131 and UML compared to IEC 61499 into their coding guidelines for control software development. It showed that CoDeSys 3.0 proved advantages over procedural programming like encapsulation of multiply needed code, thus common copy paste errors on algorithm level can be excluded or at least reduced to a minimum. The number of LOC (Lines of Code) is reduced, which also increases the readability of PLC software code. It turned out, that state charts can easily be used to model sequences. Due to state charts visual similarity to SFC a smooth integration and application is given. Since quite a few libraries offering basic functionalities exist in IEC 61131-3 languages it would be a tremendous challenge to completely replace them all by their object-oriented counterparts. BOSCH came to the conclusion, that the integration of the object-oriented feature along with the UML editor is recommended, since executable UML provides a novel structural view on software providing enormous visual advantages, while the compatibility to the IEC 61131-3 languages are still given because of the complementary relation to state charts. Furthermore the already known chances of an object-oriented procedure, like reuse and modularity can be exploited for automation control software development. HAUNI AG claims a consequent encapsulation of data is a fundamental prerequisite to high software quality, which becomes true by the means of object-oriented analysis, design throughout the lifecycle. The UML editor showed significant performance in a demo project they implemented. The application of graphical programming along with the object-oriented concepts being compatible to well-known IEC 61131-3 programming languages was the key to success. However the editor must reach quality of a commercial product.

A well-defined and theoretically suitable modeling language does not need to be the universal key to success. It is inescapable to conduct experiments to rate its usability. Therefore an experimental setup was created to evaluate the effectiveness of UML state charts in PLC programming in comparison to similar and in the PLC context elaborated graphical modeling languages, here Sequential Function Chart (SFC) specified in IEC 61131.

6.2 Lab Evaluation with Students and Trainees

Besides the industrial evaluation and ß-test we wanted to proof the advantage of the embedded UML concept. That's why we decided to conduct another experiment to test one of the main advantages of the embedded UML in a limited experimental set-up, i.e. state charts to model error handling (see Fig. 2). 30 subjects (bachelor students in mechatronic engineering) were asked to compare PLC state charts available in the UML editor to SFCs, both embedded in a standard IEC 61131-3 development environment. The focus of this experiment was the implementation of error handling routines to existing control software. It incorporates two subtasks in order to complete successfully. The given code must be interpreted in the first place and extended by creating error routines. As the null hypothesis it was assumed, that there is no difference in the treatment, so that similar scores can be reached and similar time is needed for problem solving for PLC state charts or SFCs. The alternative hypothesis posits there is no significant difference between the two graphical notations. The subjects were grouped into two equally sized teams after they had passed some short capacity tests in short-term-memory, visual realization and combinatorics. In the first task of the application example introduced in chapter 5, the subjects had to extend the given piece of code with an error handling routine for a movement timeout of a pneumatic cylinder. In the second task a consistency check on two sensor variables had to be implemented. This task is predestinated to be elegantly solved with the means of enhanced features of PLC state charts such as composite states. The outcome of the first experiment showed no significant deviation between both languages from a quality and time prospective. A remarkable quality improvement of the outcome is noticeable in the identification of errors, setting status variable and process resumption with the use of PLC state charts. The SFC group achieved better results in error handling and resetting of status variable after the error had been solved. Hence it can be summarized, that PLC state charts and SFCs are equivalently appropriate for solving the given task, since there are no significant differences. The discriminating elements in PLC state charts for error handling (composite states and transitions) were not used in the first experiment as expected. The results are not discouraging, though. The majority with eleven of fifteen subjects using PLC state charts expressed their preference for a free placement of modeling elements, where only two would rather prefer a fixed alignment. Also the SFC group (60 % free, 14 % no preference, 26 % static) mostly preferred a free placement. Throughout the entire evaluation prototypical implementation of PLC state charts were used in contrast to a commercially fully-fledged SFC editor.

7. RESULTS AND DISCUSSION: STRENGTH AND SHORTCOMINGS OF UML IN AUTOMATION

The embedded UML derived from prior research on usability of UML and a domain specific UML has been evaluated by

the inner and outer consortium of industrial companies and will be available in November as part of a wide spread engineering environment based on IEC 61131-3 (CoDeSys 3.0, TwinCAT, etc.). Companies of the consortium already apply the tool successfully and are looking for further extensions. ELAU proclaims that the object-oriented approach using UML is a key technology to cope with the complexity of today's and future software engineering tasks in automation. The importance of an appropriate and reliable tool, which meets industrial requirements and gains users' acceptance, became more than obvious during the lab experiments. The decision to support only three diagrams and to implement the activity diagram and state charts could be verified as correct by the industrial evaluation. Because modularity and reuse is strongly connected to variants and versions, the documentation of this variants and import of legacy code, which is not an issue of UML, is a key success factor for industrial application as well as some additional features implemented in the embedded UML concept.

8. CONCLUSION AND OUTLOOK

The paper described the development of an embedded UML concept and tool derived from the requirements in manufacturing and plant automation (hybrid processes) and its industrial and lab evaluation. For an adequate engineering of the entire control task in hybrid systems the integration with Matlab/Simulink as modeling approach for closed loop control needs to be realized. On this basis we intend to develop a so called technology editor for process engineers. This concept supports process engineers to execute different series of experiments on the same machinery to gain and to explore e.g. variable material characteristics of thermo-mechanical forming in a familiar environment. The integration of a closed loop control library, which is directly generated from a Simulink model, allows to build-up a hybrid system. To increase modularity and reuse in industry we need to tackle educational aspects as well in engineering education in industry and in industrial education. Rules and check-lists for the development of good modules and good interfaces need to be developed, which are certainly not domain independent. Another direction of further development is to support testing, the derivation of test cases and the automation of tests based on the structural and behavioral description of the UML model. Regarding the model itself we are developing a modeling concept to integrate energy flow into the structural behavioral model.

ACKNOWLEDGEMENTS

Stiftung Industrieforschung together with 3S, Beckhoff, Teamtechnik, ELAU (Schneider Electric) and SIG Combibloc sponsored the research on UML in automation from 2006-2008 and from 2008-2010 the implementation was supported by 3S and Beckhoff. The usability research was not supported financially.

REFERENCES

ARC Advisory Group (2009). Programmable logic controllers worldwide outlook. Five year market analysis and technology forecast through 2013.

BESTVOR (2010). URL http://www.bestvor.de.

Bonfatti, F., Monari, P., and Sampietri, U. (1997). *IEC 1131-3 Programming Methodology: Software engineering methods for industrial automated systems*. ICS Triplex ISaGRAF, Essex.

Burmester, S., Giese, H., and Schäfer, W. (2005). Model-driven architecture for hard real-time systems: From platform independent models to code. In: *Proceedings of the European Conference on Model Driven Architecture - Foundations and Applications*, 25-40, Springer, Berlin.

Diedrich, C., Krause, J., and Franke, A. (2006): UML based software development under safety constraints. In: *Sicherheit* (Dittmann, J. (ED.)), 361-368, Magdeburg

Diehm, S. (2008). Anforderungen an die Modellierung von funktionsbausteinen mit UML aus Sicht eines Systemanbieters. In: *Automation & Embedded Systems* (Vogel-Heuser, B. (ED.)), Oldenbourg Industrieverlag, München.

Estévez, E., Marcos, M., and Orive, D. (2007). Automatic generation of PLC automation projects from component-based models. In: *Int. Journal of Advanced Manufacturing Technology*, **35**, Issue 6. 527-540, Springer, Berlin.

Fischer, K., Göhner, P., Gutbrodt, F., Katzke, U., and Vogel-Heuser, B. (2004). Conceptual design of an engineering model for product and plant automation. In: *Integration of Software Specification Techniques for Applications in Engineering* (Ehrig, H., Damm, W., Desel, J., Große-Rhode, M., Reif, W., Schnieder, E. and E. Westkämper (EDs.)), 301-321, Springer, Berlin.

Friedrich, A. (2009). *Anwendbarkeit von Methoden und Werkzeugen des konventionellen Softwareengineering zur Modellierung und Programmierung von Steuerungssystemen*, Ph.D. thesis, Universität Kassel.

Gehrke, M. (2005). *Entwurf mechatronischer Systeme auf Basis von Funktionshierarchien und Systemstrukturen,* Ph.D. thesis, Universität Paderborn.

Gemino, A. and Wand, Y. (2005). Complexity and clarity in conecptual modeling: comparison of mandatory and optional properties. In: *Data and Knowledge Engineering*, **3**, 301-326, Elsevier, München.

Giese, H. and Henkler, S. (2006). A survey of approaches for the visual model-driven development of next generation software-intensive systems. In: *Journal of Visual Languages and Computing*, **6**, 528-550, Elsevier, München.

Heverhagen, T., and Tracht, R. (2001). Integrating UML-RealTime and IEC 61131-3 with Function Block Adapters. In: *Proceedings of the IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, 395-402, Washington DC.

Katzke, U. (2008). *Spezifikation und Anwendung einer Modellierungssprache für die Automatisierungstechnik auf Basis der Unified Modeling Language (UML)*, Ph.D. thesis, Universität Kassel.

Katzke, U. and Vogel-Heuser, B. (2005). Design and application of an engineering model for distributed process automation. In: *Proceedings of the American Control Conference 2005*, 2960-2965, Minneapolis.

Katzke, U., Vogel-Heuser, B., and Fischer, K. (2004). Analysis and state of the art of modules in industrial automation. In: *Automation Technology in Practice*

international, **2**, Issue 1, 23-31, Oldenbourg-Verlag, München.

Licht, T. (2004). *Ein Verfahren zur zeitlichen Analyse von UML-Modellen beim Entwurf von Automatisierungs-systemen*, Ph.D. thesis, Universität Ilmenau.

Patig, S. (2008). A practical guide to testing the under-standability of notations. In: *Proceedings of the 5th on Asia-Pacific Conf. on Conceptual Modeling*, 49-58, Wollongong.

Secchi, C., Bonfè, M., and Fantuzzi, C. (2007). On the use of UML for modeling mechatronic systems. In: *IEEE Transactions on Automation Science and Engineering*, **4**, Issue 1, 105-113, Hyderabad.

Sunder, C., Zoitl, A., Christensen, J., Vyatkin, V., Brennan, R., Valentini, A., Ferrarini, L., Strasse, T., Lastra, J., and Auinger, F. (2006). Usability and interoperability of IEC 61499 based distributed automation systems. In: *Proceedings of the 4th IEEE International Conference on Industrial Informatics*, 31-37, Singapore.

Vogel-Heuser, B. (2009a). *Automation & Embedded Systems-Effzienzsteigerung im Engineering*, Kassel University Press.

Vogel-Heuser, B. (2009b). Visionen für das Engineering der Automatisierungstechnik 2020. In: *Automatisierungstechnische Praxis*, **51**, Issue 5, 49-56, Oldenbourg Industrieverlag, München.

Vogel-Heuser, B., Friedrich, D., Katzke, U., and Witsch, D. (2005). Usability and benefits of UML for plant automation- some research results. In: *Automation Technology in Practice international*, **3**, Issue 1, 52-60, Oldenbourg Industrieverlag, München.

Witsch, D., Ricken, M., and Kormann, B. (2010). PLC state charts: An approach to integrate UML state charts in open-loop control engineering. In: *Proceedings of the 8th IEEE Industrial Conference on Industrial Informatics*, 915-920, Osaka.

Witsch, D., Schünemann, U., and Vogel-Heuser, B. (2008). Steigerung der Effizienz und Qualität von Steuerungsprogrammen durch Objektorientierung und UML. In: *Automatisierungstechnische Praxis*, **50**, Issue 11, 42-47, Oldenbourg Industrieverlag, München.

Witsch, D., and Vogel-Heuser, B. (2009). Close integration between UML and IEC 61131-3: New possibilities through object-oriented extensions. In: *Proceedings of the 14th IEEE International Conference Emerging Technologies and Factory Automation*, 1-6, Mallorca.