

Usability and benefits of UML for plant automation – some research results

Birgit Vogel-Heuser, David Friedrich, Uwe Katzke and Daniel Witsch,
University of Wuppertal

An agile software development approach for embedded systems has been transferred to automation and process control. Using UML, an approach was developed, which allows to generate IEC 61131-3 code from an UML-model and to import it into soft-PLCs, automatically. The system architecture is part of the UML-model and by that fact bridges the gap between hard and software engineering. This work is embedded in a development of UML for process automation and usability test of UML with automation engineers.

UML / Code generation / PLC programming / Distributed systems / Embedded systems

1. Introduction

Today the development of software in plant industry is confronted with an increasing complexity of problems, but shortened project duration and project costs. Significant potential for optimization during the development of automation software and hardware exists in the following areas:

Improvement of software quality, cost reduction through reuse of software and hardware modules [1], improved communication between different groups of persons involved in the development, and integrated usability of tools and methods.

The Unified Modeling Language (UML) seems to be an adequate notation for such demands. This article will discuss on the basis of the plant automation's requirements, i.e. the process requirements, the requirements of automation concepts and devices as well as the requirements of the project (tool support, personell) what benefit the UML can provide.

An agile software development approach for embedded systems has been transferred to automation and process control. By using UML 1.x, an approach was developed, which allows to generate IEC 61131-3 code [2] from an UML-model and to import it into soft-PLCs, automatically. The generated IEC 61131 code consists of ST and SFC. Besides, the system architecture is part of the UML-model, which bridges the gap between hard and software engineering. This approach was proven using a sorting machine application and evaluated by automation experts. The work is embedded in a development of UML for process automation and usability test of UML with automation engineers.

2. Requirements

An overview of requirements of process automation is listed in Fig. 1 [3].

The criteria can be structured regarding process requirements, automation system architecture, and project. In process automation, different kinds of processes are possible. A process automation system represents a type of process, e.g. batch, continuous, or discrete. Sometimes processes are composed of different process types. They are called hybrid, due to the fact, that they consist of different process kinds. These process types require different control strategies and by that they require different modeling notation features, e.g. block diagrams or state charts.

Today plant manufacturing industry mostly installs standardized automation devices for automation systems, e.g. PLCs (Programmable Logic Controller), which are programmed in IEC 61131-3 [2]. Therefore, the transfer of modeling results into IEC 61131-3 is necessary.

The IEC 61131-3 contains languages, which follow a function-oriented or procedural-imperative paradigm. The increasing use of these languages has caused a growing dependency on the accepted standard, because existing implemented systems must be expanded and the developers are familiar with the practice of "accustomed" programming techniques.

Regarding an automation project, there are typically different engineers or technicians involved with different qualification levels and subjects (Fig. 1). By that fact, the notation has to be easy applicable to a certain level for process, mechanical, and electrical engineers as well as for technicians.

Depending on the project phase, different tools and methods are used. In the worst case, the corresponding data have to be re-entered during the transition from one phase to the next, because there is a lack of appropriate interfaces between the individual tools [4]. To reduce complexity of interfaces between different models and tools modules may be one appropriate solution. The modules integrate the different models and their tool representation for themselves.

The different models or tools offer one view on the module (Fig. 2).

The challenge is to integrate the views of different project phases and disciplines, e.g. basic engineering with P&I-diagrams and simulation, detail engineering with software and electrical hardware design including circuit diagrams, simulation, mechanical engineering, process engineering, human process interface, testing and commissioning etc.

A module with these views is an aggregation of information. The ideal to strive for would be a higher-level modeling notation and strategy supported by a tool that consistently provides all system information in one model or provides powerful and clearly specified interfaces to models of other disciplines or project phases.

In a prior study the module concepts of different industrial users were examined and a common model for modules in automation was evaluated [1]. "The degree of module usage differs dramatically in different companies. The advantages of module reuse is accepted, but the effort of time and money to develop this module without an already existing appropriate engineering tool is rated definitely to high" [1]. We found that all companies base their design on a number of basic modules (mostly 10–15). "The aspects of hardware (instrumentation, electrical schematics and process / machinery) have been hardly integrated. Despite the momentarily situation the benefits of modular software and hardware (instrumentation) could be demonstrated by one of the interviewed companies. The necessity of an effective and "easy-to-use" engineering support for reusable modules was rated as a prerequisite.

The question remains how such modules should be modeled by and for application engineers. Since the benefit of notations differs with the nature of the programming task [5], the selected notations have to provide elements for the specification of hardware and structure, sequence cascades, interlocking, control loops and modular design (Fig. 1). An overview and expert rating of existing notations for automation systems were given by VDI/VDE 3681 [6]. To gain the benefits of reuse an object oriented notation should be chosen.

For application development in areas other than automation technology the principles and methods of object orientation and their UML notation have been used for some time to meet the requirements of reusability and improvement of software quality.

categories/criteria		functionality/notation aspects
process	batch (continuous)	transfer functions, block diagrams, differential equation
	discrete	Status model, flow chart, continuous function chart, petri net
	hybrid	continuous and discrete process
automation system	heterogeneous or homogeneous	distribution, communication, network/central unit different hardware platforms different software platforms HMI, diagnostics, no screen
	time	hard an soft real time; time and event controlled systems
	implementation	IEC 61131-3 for PLC (embedded system), proprietary for DCS; C, C++
project	qualification	easy to handle for engineers and technicians
	system life cycle	top down design modularity, component base, object orientation reusability
	tool support	along entire life cycle

Fig. 1: Requirements in process automation [3].

In automation technology, the utilization of object orientation is still in its infancy.

Many existing notations provide solutions for isolated requirements. Indeed no established notation fits all of them. UML has advantages concerning its applicability across different development phases or the degree of familiarity among developers as well as users, which are also of value in embedded software engineering. But the main benefit is based on its extensibility constructs. It is possible to build specialized profiles for missing subjects. Using this mechanism UML for process automation (UML-PA) has been developed [1, 7].

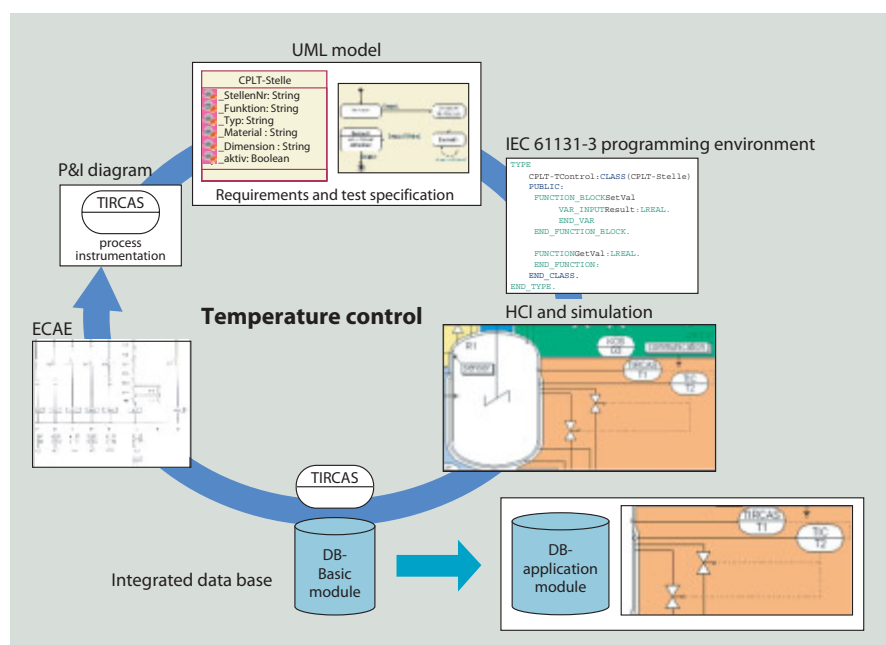


Fig. 2: Automation module temperature control with integrated views.

UML 2.0 specifies with the Profile for Schedulability, Performance and Time Specification some useful constructs for real time systems, but has no formal semantics [8]. Regarding system architecture it provides useful diagrams as composition structure diagram, which allows to model specific hardware and define symbols for that purpose. The development of the prototype is based on UML 1.x, because tool support for UML 2.0 is available recently. An evaluation with standard UML (1.x) will be introduced next.

Table 1: Design of the experiments (yellow – no significance).

Experimental design	Groups of experimentees	Group number					
		I modeling & programming			II programming with given model		
		ICL	UML	without	ICL	UML	text
students	BSc	6#	6#	6#	7	7	7
– Information Technologies	MSc	2	2	2	–	–	–
– Electrical Engineering							
Students in industrial practise		3	3	3	–	–	–
Trainees for skilled workers		–	–	–	4	4	4
Technicians		3	3	3	–	–	–
Application engineers		*	*	*	*	*	*

* To be done # groups with 2 members

3. Evaluation of UML for PLC programming

The appropriateness of UML for process control was evaluated from a usability and cognitive science point of view. Regarding Curtis' [5] mental model for programmers - two of six influencing factors of programming performance (knowledge base and cognition) - may be influenced by education and systematic software engineering methods. The benefit of modeling for programming complex distributed automation systems with real time requirements has not been evaluated yet.

If one of the notations should be applied for modeling in process control engineering it needs to be easy to use by engineers to specify the software functionality. The Idiomatic control language (ICL) was chosen because it is rated intuitively for process control engineering and includes well known and wide spread basic notational elements [9]. The ICL consists of three basic elements, i.e. a type of Sequential Function Chart, decision tables for conditions and transitions as well as idioms for controllers.

Experiments with students of electrical engineering and afterwards with technicians and skilled workers were designed and conducted. The goal was to evaluate the benefit of the UML and ICL for the use in software development regarding 'ease of learning and comprehensibility' and 'applicability to design a PLC program'.

The task was to create a model for a production process with UML or ICL and to use it for programming a PLC. For comparison, one group had to program without modeling. The procedure and the results (model, program) and time consumption of each experimentee was documented by an observer.

The results of the experimental design I and first pass (Tab.1, highlighted yellow) revealed problems building a model in the evaluated notations (Tab. 1). The students rated their notational knowledge and the ease of modeling usage as good in the questionnaire (post). The documented modeling results were analyzed and showed lacks in students knowledge. The models were analyzed regarding

- level of detail,
- notational correctness (e.g. mixture of state and transition (UML), missing transition details),
- modularity and reuse.

The IEC 61131 program was analyzed by means of code inspection (e.g. correctness of transition conditions, structure).

Some results are summarized as follows:

- The acceptance of abstract modeling raised with the complexity of the task.
- All experimentees rated modeling as useful, if the task was complex enough.
- The experimentees rated their notational knowledge as good and the applicability of the modeling notation as easy (questionnaire)

The correlation between the modeling and programming result and the questionnaire wasn't significant (values under 0,6) and the benefit of modeling prior to programming could not be shown. Possible reasons are groups instead of single test person (influence of group behavior and using averages) and the low quality of the created models.

Due to these results, the set-up was changed in experimental design II and second pass to reduce complexity of the experimental design. The test persons then worked separately and obtained a complete model or a textual description of the process. The first task was to answer questions about the process after working through the model. Afterward they had to use the model for programming the PLC.

As experiment II, second pass (highlighted green) has just been finished, results are available as a trend. Regarding the ability to understand the models, the textual specification and the ICL showed advantages in answering questions about the process. Nevertheless, the UML groups required less time for the transfer from model into program code, their transitions were specified more precise and they designed a more modular program code. Comparing to the textual specification UML seems to be more advantageous regarding the programming task.

The results of the experiments with the target group, i.e. skilled workers and technicians were conducted already. The group of trainees for skilled workers was tested due to experimental set-up II, because they showed significant lacks in UML during lectures and exercise prior to the experiment. The experiments showed so far, that standard UML is not significantly better than other notations for PLC programming.

- Reasons may be,
- the lack of a guideline or strategy, i.e. which diagram should be used for what purpose. A pre-experiment underlined this influence.
 - Standard UML needs to be adapted to support application in process automation,
 - Advantage of inheritance is weak in this application. Kim and Lee [10] showed that programmers using object oriented design "spent less time in absolute and relative terms in the task [application] domain." Object Oriented Design's "emphasis on abstraction helps to find the underlying problem structure";
 - lack in tool support and
 - lack in automatic code generation (IEC 61131),
 - experience using UML. With exception of some members of the students groups all experimentees used UML the first time. The familiarity with UML increases its advantages regarding time and quality of a model [11]. For application in industry this argument is weak, because under economic conditions the advantage needs to be visible in the first project.

The applicability of UML should be improved significantly by means of the six reasons mentioned above.

4. Automatic code generation from a UML model to IEC 61131-3

The different aspects of code generation from UML will be discussed starting from the principle of the code generation as an overview, the mechanism of code generation and the extension of the UML metamodel. The mapping of the UML model to the IEC 61131-3 as the software part of the UML model will be discussed in chapter 4.2. In chapter 4.3 the interfacing between software and hardware architecture will be explained. Finally the valuation by expert rating will be discussed.

The development of the prototype is based on UML 1.x, because tool support for UML 2.0 is available recently.

The structured analysis offers notations and models of operation for a top down design of automation systems. This concept follows a strategy of successive refinements. It identifies objects, but it has neither a mechanism nor a notation for concentrating common attributes and behavior to reusable classes.

A UML strategy designed for modeling embedded systems [12] was adapted to meet the requirements of automation technology. This adaptation considers hardware boundary conditions and real-time aspects and leads to a hierarchic model that pragmatically supports the system development process from comprehensive requirements analysis to technical soft-

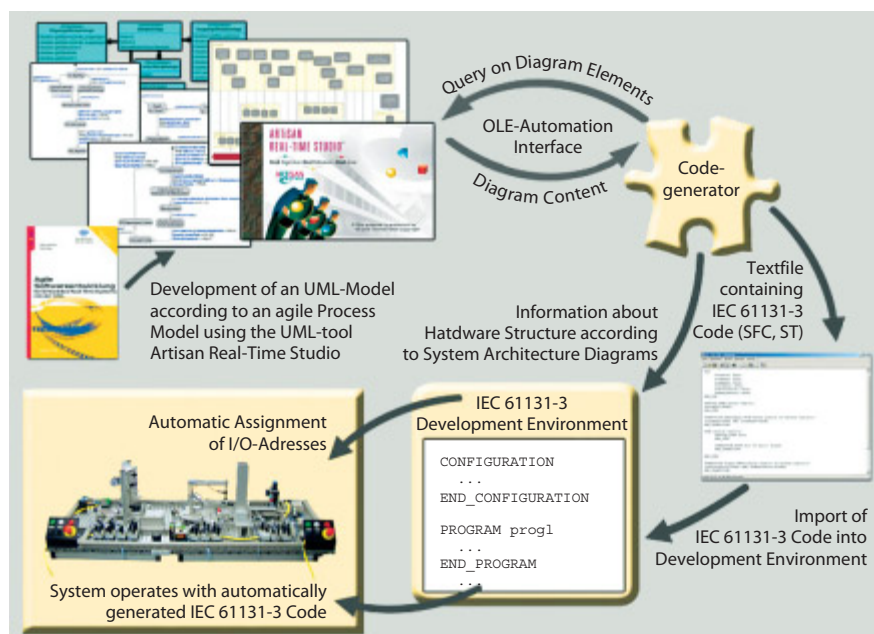


Fig. 3: Principle of code generation.

ware design. The real time aspects are discussed in [13] and is beyond the scope of this paper.

Real-Time Studio from Artisan [14] was used as modeling tool. This tool is particularly suitable due to its modeling flexibility and open software architecture, enabling simple integration of third-party tools via OLE interface as well as enlargement of the UML metamodel. Besides the tool provides system architecture diagrams, which allow to integrate systems hardware classified by using stereotypes parameterized with tagged values. Based on the above mentioned strategy an application example was modeled. From this model, a code generator automatically creates the IEC 61131-3 code (SFC and ST) and derives system architecture information, which are automatically imported into the PLC programming environment and into the system configuration tool. By that the application example is fully and correctly automated (Fig. 3). The code generator creates a plain text file with IEC 61131-3 code. The IEC 61131-3 programming environment has to support this import.

By using the graphical representation of the hardware architecture the mapping of I/O addresses to variables (software) could be realized.

All CoDeSys implementations [15] support this functionality. In this prototype TwinCAT [16] was used. The concept for an import to Siemens S7 has already been developed.

4.1. Mechanism for code generation

The OMG (Object Management Group) guarding the specification and development of UML defines the application of UML as follows: "The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems." "... However, users may sometimes require additional features beyond those defined in the UML standard. These needs are

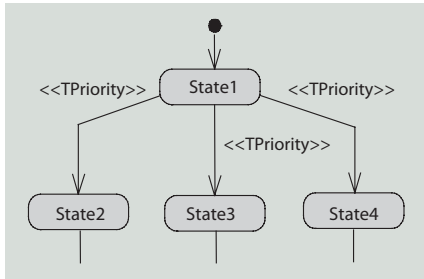


Fig. 4: Statechart with stereotype <<TPriority>>.

met in UML by its built-in extension mechanisms ...” [17]. Exactly these features are used for the automatic code generation. Exemplarily the usage of one extension mechanism will be explained here: the UML state chart with transitions which fire depending on guard conditions and priorities. These type of transitions is implemented in IEC 61131-3 SFC. To design this extension mechanism a stereotype <<TPriority>> was defined, which is associated to a transition (Fig. 4) and is restricted to UML elements of type transition.

The value of the priority will be specified with a tag *priority* type *Integer* added to the stereotype <<TPriority>>.

This extension mechanism affects the UML-Metamodel. On metamodel level a new class *transition* will be derived (Fig. 5) and by that a new UML element is created.

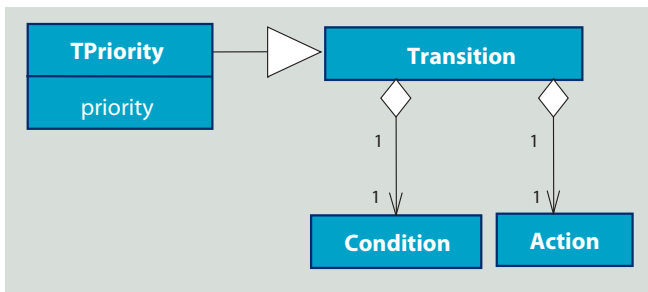


Fig. 5: Part of the enlarged meta model with the new class *transition*.

To this new UML element, which is derived from an existing element using a stereotype, attributes can be assigned. The attributes are named tags, their values are named tagged values. Tags may be data types, enumerations or references to other model elements. Using references allow to add associations in the metamodel. With stereotypes and tagged values the metamodel may be enlarged by new classes, attributes attached to these classes and by new connections between different UML elements. A domain or problem specific collection of stereotypes with their tags and constraints is called a profile (constraints are beyond the scope of this article). For example the *profile for schedulability, performance and time* [8] integrates real time aspects to UML.

4.2 Mapping the UML model to IEC project structure

Class diagrams are appropriate to describe the static code structure. Using stereotypes the class itself and its relation to

The UML metamodel

UML 1.x defines nine different diagrams to model the static structure, the dynamic behaviour and implementation aspects. Those diagrams offer a different point of view on the requirements. All these diagrams and diagram elements are joined together by the meta model. The meta-model specifies the relations between the different elements of the UML. The metamodel itself is specified through a set of class diagrams and can be understood as the model of UML.

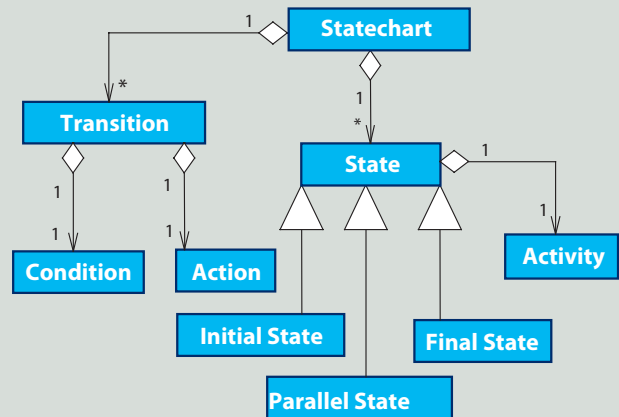


Fig.: Part of the meta model of state charts (simplified) [full specification see 17, p.2/141].

The state chart metamodel (Fig.) shows that transitions and states are associated to a state chart. Besides simple states a state chart can contain initial, final and parallel states which are derived from simple states. A state is related to an activity. A transition should have conditions and actions. In this way all UML elements and their interconnections are specified.

other classes can be modeled in its entirety. For the dynamic behavior activity diagrams and state charts are of interest. Collaboration (replaced by communication diagrams in UML 2.0 [18]) and sequence diagrams show only scenarios or use cases. The similarity of state charts in UML and sequential function charts (SFC) in IEC 61131-3 is obvious. Therefore state charts are chosen for the dynamic model.

The IEC 61131-3 language definition does not include classes yet [1, 2]. But other constructs in IEC 61131-3 offer similar mechanisms. Data type structures can be instantiated and allow to define structures. So nested data type structures are used to build classes with attributes and operations (Fig. 6).

The top level type definition named as the class includes the methods' and attributes' type definition. The methods type definition consists of FB instances, which implement the methods of the class. The attributes' types definition encapsulates all public attributes of the class with its' data type and a further type definition for the private attributes. The following example shows the instantiation and usage of attributes and operations of this class construct.

```

VAR
    MyClass : Class; (* instantiation *)
END_VAR
    
```

```

MyClass.Attributes.MyAttribute1:=TRUE;(* assignment*)
MyClass.Methods.MyMethod2(...); (* call *)
    
```

In the UML model, each task is represented by a package. A package contains a class diagram consisting of classes stereotyped according to their particular purpose. This could be the collection of data (entity-classes), reusable functions (service-classes) or the concentration of central control operations (service-classes) [12].

Each method within a class is assigned a state automaton (Fig. 7) that defines its dynamic behavior. Another state automaton describing the interaction of its methods is assigned to the control class itself. This state automaton becomes the main program of the task in the IEC 61131-3 project. This main program calls function blocks (FB) which result from the conversion of the control class's methods. These converted methods can in turn access the functions or function blocks which originate from methods of the service class for carrying out standard operations. This defines the call structure within the task. The state automatons of the UML are translated to Sequential Function Chart. The state automatons of the UML are translated to Sequential Function Chart. As this prototypical implementation is restricted to one PLC the modelling of the resources is implicit and access paths are unnecessary.

4.3 System architecture

For modeling hardware, Artisan Real-Time Studio offers so-called system architecture diagrams (SAD). SADs can be used in terms of pre-stereotyped UML deployment diagrams. The SADs are used to model hardware components, actuators and sensors. In order to use SADs as a basis for an automatic hardware configuration it was necessary to enlarge them by certain stereotypes for example *bus coupler* or *bus terminal* (Fig. 8). These stereotypes may be enriched with all relevant information available in a data sheet for this device. The integration of UML 2.0 with its system diagrams is one of the next working packages. In UML 2.0 the visual presentation of a stereotyped UML-element can be adapted to any supplier symbol.

The variables used in the class and state diagrams are assigned to the actuators and sensors using stereotypes with a reference tag.

The information which program variable is connected to which hard-

ware adress can automatically be transferred to the hardware configuration tool of the PLC programming environment, e.g. TwinCAT system manager. In this way a modification of the hardware structure can be realized by simply changing the associated arrow in the SAD (Fig. 8). The re-wiring is thus automatically performed correctly by the developed software.

In the prototype the SAD's similarity to a system architecture, network architecture or I/O-allocation in a circuit diagram is aimed at. Unfortunately up to now there is no common standard for hardware elements available and the different soft PLC tools (e.g. 3S, Beckhoff, KWSOFT and Softing) implement this in a different way.

One approach is to use the UML-model as a common basis for different engineering views and platforms. Which means modeling of the system's architecture using UML in order to generate different data, which can be consumed for example by ECAE-tools. The export or import to or from ECAE-systems can be performed via XML. This functionality is required by a number of system integrators. Another approach is to program interfaces between the different models and their tools.

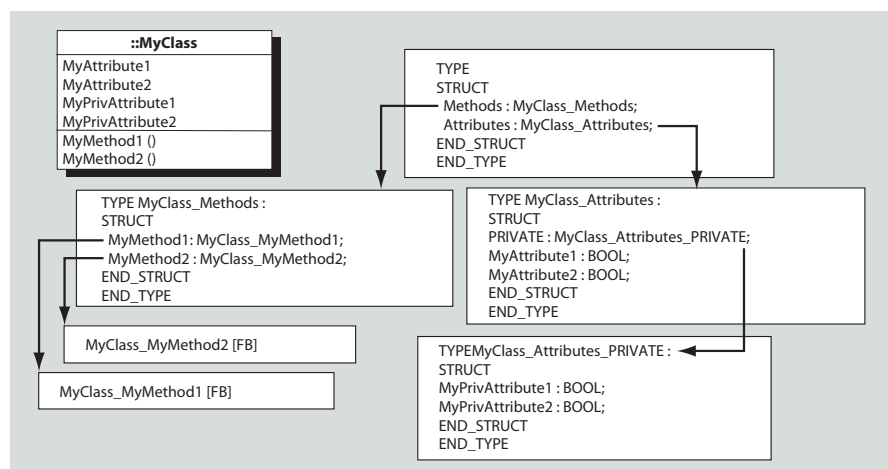


Fig. 6: Class construct in IEC 61131-3.

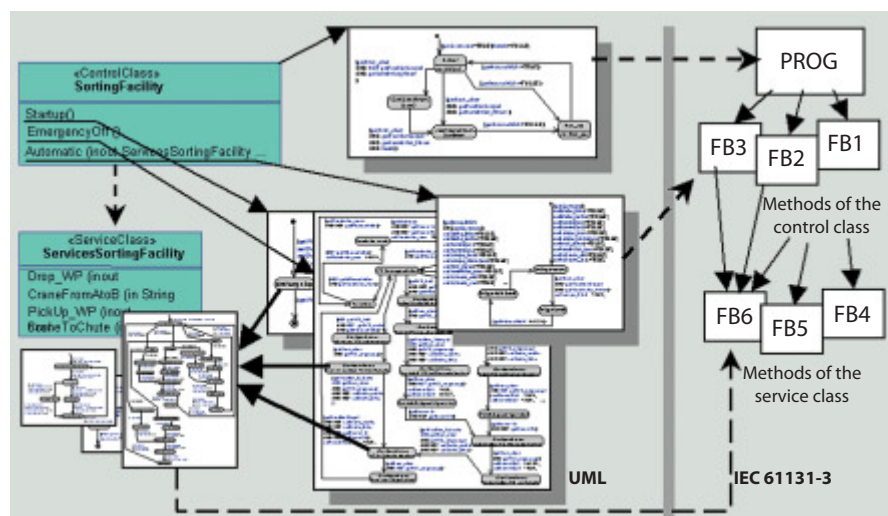


Fig. 7: Mapping of the UML model to IEC 61131-3.

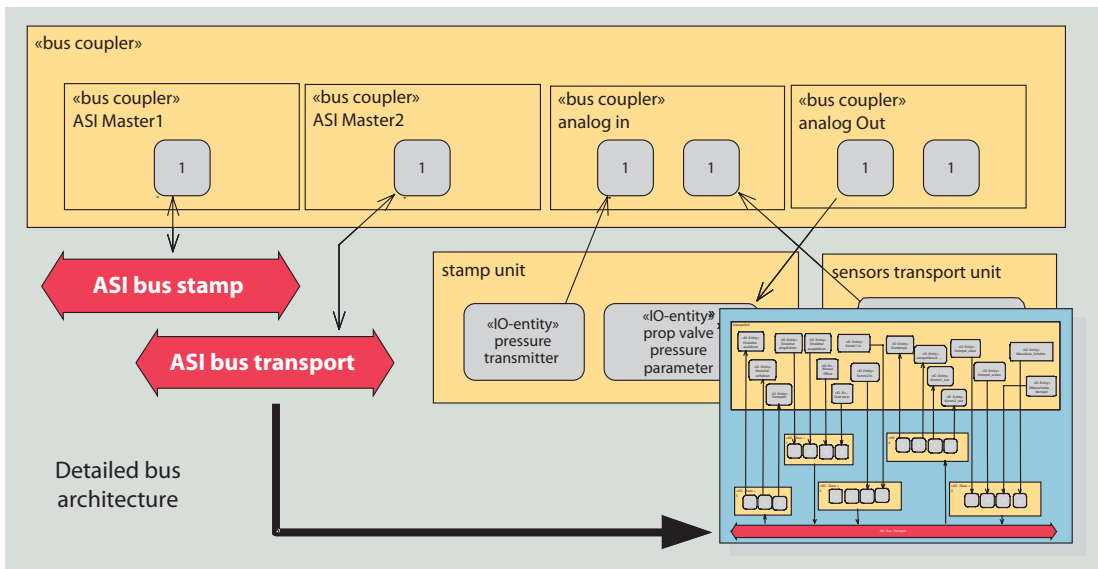


Fig. 8: System architecture and bus hierarchy.

4.4 Valuation of the prototype

The strategy and the code generator were evaluated with a simple application example representing a manufacturing process in which workpieces are processed differently depending on the results of a material analysis (inductive, optical). The “agile” strategy as well as the code generator could be proven.

The next evaluation step includes from automation industry experts and system architects for IEC 61131 environments and from system integrators applying such environments system architects and application engineers.

The result of this evaluation is summarized as follows. To profit from object orientation reuse by inheritance and module variation is targeted.

The application of UML in machine and plant automation is on the one hand appreciated for specification of automation functionality and architecture in general and for the specification of reusable modules. On the other hand most managers or head of design departments are pessimistic about its applicability for average application engineers.

Transparency was estimated low, when using inheritance and changes on the top level occur.

Another main argument against code generation from a model is the lack of back propagation. One requirement is, for example, to allow the start-up engineer to program in the case of malfunctions directly in the IEC 61131-3 code. Those changes need to be returned consistently to the overall UML model. With IEC 61131-3 in its present form this is only possible using a large number of rules. An extension of IEC 61131-3 with object orientation constructs is already under discussion [15]. If such constructs were included in the standard, back propagation from IEC 61131 to UML and application of UML without semantic breaks would be easier to realize. Until such time, the approach of simultaneous representation of the UML model and IEC 61131-3 code can provide concrete assistance.

The systems integrator group of experts is more optimistic, especially when using ST as one of the IEC 61131-3 pro-

gramming language or C. The connection between software and hardware was estimated most important. The acceptance of UML was rated as good, when appropriate tool support is available and a guideline with best practice as well as a library of standard modules is available to start with. This was reached by a developed import functionality of already existing IEC 61131-3 function blocks into the code generator.

The different disciplines require different information in depth and representation, they use different proprietary tools, but lack for an integrated project view to find interfaces in case of partial redesign (Fig. 9). UML may provide the basis, but therefore transformations between the different models and tools need to be realized. A commercial tool is required, which provides this functionality and open interfaces.

The code generator and the strategy could be proven successfully, but automation specific requirements need to be handled more efficiently and easier for application engineers. The adaption of UML for process automation is one approach.

4.4 Adapting UML for process automation (UML-PA)

UML-PA is an adaption of UML for process automation systems, i.e. modeling of controllers handles time constraints, dynamic redundancy, time triggered synchronization of distributed systems, communication structures between encapsulated modules (from the UML-RT profile), structures for clear multiple inheritance relations and an enlargement of the deployment diagram to describe the mapping of software and hardware architecture. All elements, which extend the UML to the UML-PA are realized through stereotypes and constraints. Therefore they are based on the extensibility constructs of the UML. In detail the UML-PA contains the following enhancements [7, 13]:

- time constraints on architectural level
- timed state machines

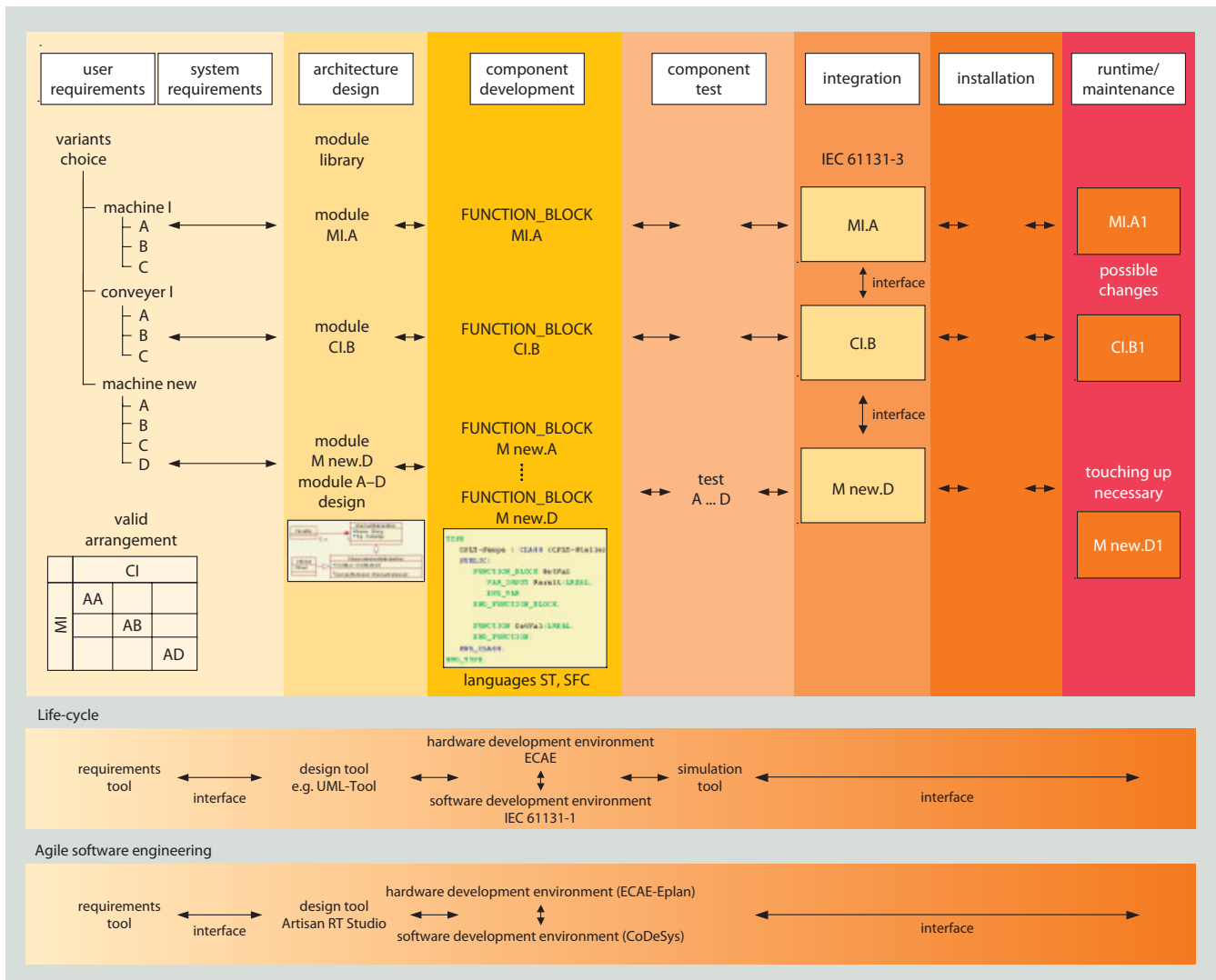


Fig. 9: Module development in plant manufacturing industry.

- communication by ports and protocols (which will not be discussed here) and
- mapping between software and hardware

Most UML-PA extensions can be realized through annotations within such tools. Based on the UML code generator the UML-PA should be interpreted. By this it will be implemented in such a tool.

5 Results and future work

The prototype was used to demonstrate that automatic code generation for automation technology can be achieved through pragmatic application of UML. However, the actual benefits of object orientation have not yet been exploited. The benefits – but also any problems – will become apparent during application in a more complex system, e.g. a real machinery for packaging including UML-PA Elements. This has been started already. In such application mechanisms such as inheritance will be used. This will significantly simplify the administration of variants and modules. The inte-

gration of network aspects is another target. For distributed systems the performance of networks and the mapping from software modules to different hardware structures depending on the project size, selected variants and the network performance is a prerequisite. The integration of UML 2.0 with its system diagrams has been started. Besides technical requirements a guideline for application engineers was developed, which provides an “agile” strategy for UML application.

For usability studies a huge field for research is still open, but the experimental design is difficult, but nevertheless only such experiments can help to understand what hinders the successful application of UML in process automation. This analysis allows to derive beneficial notations and engineering support for application engineers in the future. UML offers the opportunity to integrate several views of different project phases and disciplines, e.g. basic engineering with P&I-diagrams and simulation, detail engineering with software and electrical hardware design including circuit diagrams and simulation, mechanical engineering and process engineering, human process interface, testing and commissioning etc.

The different disciplines require different information in depth and representation, they use different proprietary tools, but lack for an integrated project view to find interfaces in case of partial redesign. UML may provide the basis, but therefore transformations between the different models and tools need to be realized.

References + Links

- [1] *Katzke, U., Vogel-Heuser, B. and Fischer, K.:* "Analysis and State of the Art of Modules in Industrial Automation", *atp international* 1 (2003), No. 1.
- [2] *Bonfatti, F., Monari, P. D., Sampietri, U.:* IEC 1131-3 Programming Methodology. *CJ International, Seyssins* (1997).
- [3] *Vogel-Heuser, B., et al.:* Conceptual Design of an Engineering Model for Product and Plant Automation. *Integration of Software Specification Techniques for Applications in Engineering* (Ehrig, et al., eds.), *Lecture Notes of Computer Science, 3147, Springer, Berlin* (2004).
- [4] *Vogel-Heuser, B.; Friedrich, D.:* Integrated Automation Engineering along the Life-Cycle. *Proceedings of IECON'02, 28th Annual Conference of the IEEE Industrial Electronics Society, Sevilla, November 2002.*
- [5] *Curtis, B.:* Five paradigms in the psychology of programming, *Handbook of Human-Computer Interaction* (M. Helander), (1988).
- [6] *VDI/VDE 3681, Ausgabe: 2004-01:* Classification and evaluation of means for description in automation and control technologies. *Beuth-Verlag, Berlin* (2004).
- [7] *Katzke, U., Vogel-Heuser, B.:* UML-PA as an engineering model for distributed process automation. *Accepted paper IFAC world conference 05.*
- [8] *OMG UML Profile for Schedulability, Performance, and Time, Version 1.1,* <http://www.omg.org/technology/documents/formal/schedulability.htm>
- [9] *Friedrich, D., Vogel-Heuser, B., Bristol, E.:* Evaluation of Modeling Notations for Basic Software Engineering in Process Control. *29th Annual Conference of the IEEE Industrial Electronics Society (IECON 03) in Roanoke, Virginia, USA, November 2003.*
- [10] *Kim, J., Lerch, F. J.:* Towards a Model of Cognitive Process in Logical Design: Comparing Object-Oriented and Traditional Functional Decomposition Software Methodologies. *Human Factors in Computing Systems (CHI 92) Proceedings, May 1992, pp.489-498.*
- [11] *Curtis, B.:* Objects of Our Desire: Empirical Research on Object-Oriented Development. *Human-Computer Interaction, 1995, Vol: 10, pp.337-344.*
- [12] *Hruschka, P., Rupp, C.:* Agile Softwareentwicklung für Embedded Real-Time Systems mit der UML. *Carl Hanser, München, Wien* (2002).
- [13] *Katzke, U., Vogel-Heuser, B.:* Design and Application of an Engineering Model for Distributed Process Automation. *Paper for the Invited Session Networked Control American Control Conference 05.*
- [14] *Artisan Real-Time Studio:* www.artisansw.com/products/professional_overview.asp
- [15] *CoDeSys:* www.3s-software.com/index.shtml?ProductTour
- [16] *Beckhoff TwinCat:* www.beckhoff.de/german/twincat/default.htm
- [17] *OMG Unified Modeling Language Specification, March 2003, Version 1.5* <http://www.omg.org/technology/documents/formal/uml.htm>
- [18] *OMG Unified Modeling Language Specification, Version 2.0,* [http://www.omg.org/cgi-bin/doc?ptc/2004-10-02.](http://www.omg.org/cgi-bin/doc?ptc/2004-10-02)

Received: December 7, 2004



Birgit Vogel-Heuser (Prof. Dr.-Ing., 43) is head of the chair of automation and process control engineering at the university of Wuppertal, Germany. Her research focus is systems and software engineering for automation and real time systems including modelling and HMI.

Address: Chair of Automation and process control engineering (FB E), University of Wuppertal, Rainer-Gruenter-Str. 21c, 42119 Wuppertal, Germany, phone +49 202 439-1945, fax -1944, e-mail: bvogel@uni-wuppertal.de



David Friedrich (Dipl.-Ing., 31) is a researcher and Ph.D. student at the chair of automation and process control engineering at the university of Wuppertal since 2001. His research covers engineering along the life cycle, the evaluation of the Unified Modeling Language in automation engineering and 3D data visualisation.

Address: see above, phone +49 202 439-1186, fax -1944, e-mail: friedri3@uni-wuppertal.de



Uwe Katzke (Dipl.-Inform., 42) is a researcher and Ph.D. student at the chair of automation and process control engineering at the university of Wuppertal since 2002. His research covers failure management systems, modularity and object oriented paradigms in automation engineering.

Address: see above, phone +49 202 439-1942, fax -1944, e-mail: katzke@uni-wuppertal.de



Daniel Witsch (Bsc, 25) joins since 2001 the chair of automation and process control engineering. 2004 Bachelor Degree Information Technologies. He is working on his Master's thesis at the ENS-Cachan, Paris in cooperation with the university of Wuppertal. His work covers UML-based engineering and code generation for control systems.

Address: see above, phone +49 202 439-1942, fax -1944, e-mail: witsch@uni-wuppertal.de

