Evaluation of an UML Software Engineering Tool by Means of a Distributed Real Time Application in Process Automation

Katja Fischer, Gregor Hordys, Birgit Vogel-Heuser

University of Wuppertal
Chair of automation and process control engineering
department of electrical, information and media engineering
Rainer-Gruenter-Str. 21
D-42119 Wuppertal
kfischer@uni-wuppertal.de, hordys@uni-wuppertal.de, bvogel@uni-wuppertal.de

Abstract: Today the development of software in process automation is a step by step strategy along the life cycle with different notations and different tools. The requirement analysis and the basic engineering could be described and structured with project management software using natural language. The software design is mostly function oriented and component based with IEC 61131-3 development environments and implemented on different targets. There is a high demand for modeling software using the UML has been evaluated for process automation regarding a typical real time application using one of the leading UML tools (Rhapsody from iLogix).

1 Introduction

Process control engineers need to discuss the functionality of a plant in an early phase of a project. A "language" to communicate between different skilled engineers is necessary, which is based on the requirements of the process itself. The quality of the notation is strongly depending on an appropriate modeling concept for the process characteristics.

Besides typically different engineers or technicians with different qualification levels and subjects are involved along the project life. For that reason, the notation has to be easy to use for process control and software engineers as well as for technicians and needs to support the entire engineering life cycle.

Software engineering in process control engineering and automation in addition has many deficiencies in method, notion and tool support. As a result, the use of software engineering methods, e.g. UML or object oriented approaches, is not wide spread in process control engineering or product automation. Nevertheless the effects regarding start-up times, additional costs and low software quality are immense.

In machine and plant automation huge application software and hardware has to be developed often with much more than 3000 input/output points (process variables),

which represent sensors and actuators. A plant is unique and therefore systematic approaches and modeling were neglected until now. Software and hardware were mainly tested and approved on site due to the lack of simulators and the fact that technology and mechanics are assembled for the first time on site.
Reduced time schedules and re-engineering on site with high costs and time pressure, lead to change also in this specific industry. Therefore an appropriate support for engineering has to be developed.
Based on a detailed requirement analysis for distributed systems in process automation including the specification of distributed systems, this research will derive a draft for an object oriented approach for this domain.
To achieve this target, several work packages have to be solved. Besides the analysis of UML itself and the enlargement with appropriate stereotypes, existing UML standards and UML tools need to be analyzed regarding the requirements of process automation. Because of applications size in plant automation modeling without an appropriate tool is not acceptable and economically suitable. As a pragmatic approach the tool analysis should show, whether one of the existing UML tools could be modified and adapted to the requirements of process automation and introduced in plant automation with slight changes. Or otherwise UML needs to be enlarged and on this enlargement tool development could start. This second approach is certainly the systematic approach on the other hand the time need for it is huge and in between the situation in process automation remains unsatisfying.
The first alternative will be discussed in this paper: the evaluation of an UML tool modeling a real time application.
The targets were the evaluation of an object oriented approach with UML and, as a result of the requirements, the evaluation of an UML tool along the entire life cycle. This should be done with modeling the prototypical application for the redesign of one component of a part out of a complex manufacturing plant. The result of the analysis should be directly applicable for design and implementation.
The characteristic requirements of process automation will be discussed at first and the application example will be introduced. After that it is described why UML should be used in process automation and why it is necessary to evaluate a tool. Before the tool is evaluated UML and the used tool are introduced.

2 Process Automation

2.1 Requirements

The requirements of process automation especially plant automation can be structured regarding requirements of the process, the system architecture of automation system and the project (table 1).
A plant consists of several parts of smaller plants, which may represent a type of process, e.g. batch, continuous, or discrete. The entire process is called hybrid, due to the fact, that it consists of different process types. These process types require different control strategies (closed and open loop control) and by that it requires different modeling notation features, e.g. block diagram or state machine.

Table 1 Summary of requirements [FV02]

Category / Criteria		Functionality / Notation Aspects
Hybrid Processes	batch	state transitions
	continuous	(sequential functions)
	discrete	closed/ open loop
		Interlocking
Automation System	heterogeneous	distribution, communication, network different platforms
		HMI and diagnosis
	time	hard and soft real time
	implementation	IEC 61131-3 for PLC, proprietary for DCS
Project	qualification level	easy to handle for engineers and technicians
	system lifecycle	specification top – down
		Modularity
		Reuse
	tool support	entire life cycle

Today plant manufacturing industry requires standardized automation devices for automation systems, e.g. PLCs (Programmable Logic Controller), which are programmed in IEC 61131-3 [BMS97]. Therefore the used tool should allow transferring of its modeling results into IEC 61131-3, because start-up, operation and maintenance need to support this standard PLC-programming language. For special tasks such as safety related tasks or hard real time requirements additional automation devices may be used e.g. process control computers with a real time operating system (RTOS). For hard real time systems specific requirements need to be realized. A list of implementation oriented real time requirements is shown in table 2. A modeling tool should also provide methods to deal with aspects of real time development like reactivity, multi-threading, time-based behaviour and real time environments. The constructs, which are listed in table 2, are based on notation constructs of the real time programming language PEARL [Sp04]. One main lack in modeling of real time application is the implementation aspect, which will decide, whether the hard real-time requirements will be met.
Interrupts and task dispatching are necessary for programming real time applications. Interrupts and task dispatching are essential already during the design of real time software. Therefor they are also needed to model real time systems, especially if source code should be generated automatically. An interesting question is whether those programming constructs are needed in a tool-supported development process directly, or whether they can be used hidden away under a certain abstraction level. But regarding the evaluation this aspect is necessary.

The communication is realized via different bus-systems:. In the field level field buses like PROFIbus DP and Interbus are deployed. Whereas the communication between and inside of process control level, plant management, and enterprise administration level normally is based on Ethernet (TCP/IP). For operation and maintenance a PC-based human machine interface is used. By that fact the architecture of the automation system is heterogeneous.

Table 2 Real time requirements

Useful and Necessary Language Constructs for Real Time Programming	**Useful and Necessary Description of Hardware Constructs**
Task dispatch	Connection between peripheral device and technical process
Transition control between different states of a task	
Scheduling	Modelling of input/ output
Synchronisation of tasks (Semaphores)	Description of different process computer units
Task activation (time/ event)	
Communication between tasks	Connection between different computers
Interrupts	

Regarding an automation project there are typically different engineers or technicians involved with different qualification levels and subjects. As a result the notation has to be easy to use for process, mechanical and electrical engineers as well as for technicians to a certain level. A more visionary requirement is to support the entire life cycle with one consistent model, but appropriate notation for each phase of the project.
In plant automation one criteria for a successful project is to develop the automation system precisely to customer requirements even if this functionality is mechanically under development. During design re-use of developed modules needs to be enabled. The test of this functionality prior to the start-up on plant site is the next challenge. Therefore the specification, which should include testing requirements for soft- and hardware is most important for the project.

2.2 Application Example

The target was to apply UML prototypically for the redesign of one component of a part of a manufacturing plant for timber industry. The whole plant mass-produces fibreboards. The so-called continuous thermo hydraulic press, which was the component with the most restrictive requirements, was viewed and should be modelled. Time critical closed loop control has to be combined with open loop control and switching to other control loops. For a better understanding one feature should be explained (simplified). The material, which is already mixed with glue, has to be pressed with a specific pressure to a certain distance due to the set value of the finished board’s thickness.

In figure 1 this part of the press is shown. Such a press could be composed of almost 80 frames. Every frame has two distance sensors and from two to five hydraulic systems, which consist of a valve for pressure increase and pressure decrease as well as a sensor. The distance control is realized by these hydraulic systems.

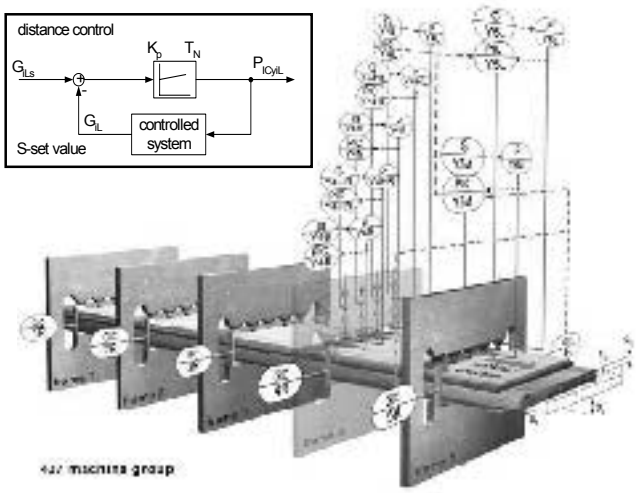

Figure 1 Application example with block diagram
(block diagram for frame i, i= 1- 80, L – left system)

During the process the pressure has to be kept in a certain limit, but the thickness of the material (i.e. distance of the press gap, GiLs) should be reached. A maximum pressure is set because of technological reasons. The real hydraulic pressure (PICyiL) and the real distance (GiL) is measured additionally.
The pressure has to be controlled in two modes: the distance control and the pressure control mode. Usually the hydraulic press runs in the distance control mode. The distance control mode is the mode in which the set value of the distance is reached with the pressure between the upper and the lower limit. If the distance could not be reached within these limits, the mode is switched to pressure control. The difficulty is that all frames have to switch synchronously into the other mode and only in this case, that all frames could change it. Otherwise the press would stop. The closed loop control of each frame has to be accomplished in 30 ms. Only several frames are controlled by one processor. The specific frames are connected to the processors and the specific processors among themselves via field bus.

3 Selection of the Modelling notation and application of UML

Thus far, in process automation, systems were developed function oriented e.g. with IEC 61131-3 [BMS97].
The question, which modelling approach and notation should be used, will be discussed next.

Schnieder et al. [Iv04] analyzed several modelling techniques and their suitability for different process characteristics. Fischer et al. evaluated UML/RT [FV02, SR98] and Friedrich et al. [FV03] worked on a comparison of modelling techniques for process control engineering. Biermann et al. [BV02] analyzed UML and Idiomatic Control Language (ICL) regarding decentralized systems.
The results of these approaches show the lack in an appropriate accepted modelling technique for the design of plant automation integrating hardware and software as well as architectural aspects with appropriate tool support.
However aspects as for example reusability and modularity are aimed but not achieved up to now in plant automation industry. Nested structures and encapsulation are not considered at all in industry until now. These aspects are well known and adapted in computer science. Indeed computer scientists deal with the same kind of computers and profit by the homogeneous structure of these systems. For uniform systems general structures can be designed easier. But in process automation industry – especially in machine and plant manufacturing – the developer deals with heterogeneous systems.
Nevertheless the solution appears to exist with an object oriented approach and UML and it seems, it only has to be mapped. Therefore an object oriented approach will be evaluated. In process automation UML is already accepted for documentation of use cases and as a common language between engineers for specification.
, which should support reusability. In industry a modelling notation comes only into operation, when there is a tool which supports such modelling. Therefore it is also necessary to evaluate tools.
The constraints and limitations of using UML in plant manufacturing industry to gain industrial acceptance in this domain needs to be analysed. For higher transparency the different diagrams of UML need to be proven due to necessity and applicability for automation projects. These constraints depend strongly on the applied tool.

3.1 Unified Modeling Language

The Unified Modeling Language (UML) [RJB99] was developed as an application independent formalism with the target to create a universal language for analysis and design of systems. It is based on different object oriented methods and notations of modelling like Object Modelling Technique (OMT) [Ru91], Object Oriented Software Engineering (OOSE), and Fusion [Pa98]. UML itself is a notation.
1997 UML Version 1.1 was declared as an international standard by the Object Management Group (OMG) [OMG03] and is further developed by them.
UML provides different diagrams with specific notations for specific views on a system. An overview is shown in table 3. The structure could be described in use case, class, component, and deployment diagrams. The dynamic behaviour could be modelled with state charts, activity, sequence, and collaboration diagrams. In addition, for management purposes UML offers a model management view. It describes the model itself and is visualised in class diagrams. It is composed of a set of packages, which could consist of classes, use cases, and state charts.
UML version 2.0 is adopted in 2003 [UML03] This version includes special constructs for performance, time, and scheduling. Today the specification is being worked out.

Table 3 UML Overview [FV02]

View	Diagram Type	Task
Structure	Use Case Diagram	• to describe user's view • to define the view of a system concerning the environment • to give a review of the functionality of a system
	Class Diagram	• to describe the structure of a system
	Component Diagram Deployment Diagram	• to define the physical architecture of a target
Dynamic	State Charts Activity Diagram Sequence Diagram Collaboration Diagram	• to describe the Dynamic of a system • to define relations between objects, activities • to define transitions between states of an object

3.2 UML Tool

For representative results different UML tools have been evaluated. At first Rose RT from Rational was discussed in [FV02]. Rhapsody, which is discussed in this paper is the second evaluated UML tool. The evaluation of Real Time Studio (ARTISAN, [AR03]) is in process.
Rhapsody 3.0.1 from iLogix is a visual design environment that enables engineers to use UML during the entire life cycle. For this purpose it provides all common types of constructive views except the deployment diagram. All diagrams use UML notation and most symbols have semantically precise meanings in the underlying model This is necessary for unambiguous implementation. Rhapsody also supports code generation as well as diagram animation. The diagram animation is offered at an early design level for state chart and sequence diagrams. This gives the ability to analyse and specify the intended behaviour of an application stepwise in the development cycle. The diagram animation, is a useful part of the design environment and helps to debug the system in the design phase rather than the executing model based on the generated source code. Sequence diagram comparison at runtime is also supported. It allows comparisons between hypothetical and real message sequences. Rhapsody provides methods to deal with aspects of real time development like reactivity, multi-threading, time-based behaviour and real time environments. The tool supports modelling active and reactive objects. Active objects are application objects with active concurrency (environment with several threads) that run on their own thread of control. They also own an event queue through which they process their incoming events. The reactive objects are application objects with sequential concurrency that run on the system thread. State charts define the reactive (discrete) behaviour of objects by specifying how they react to messages. A message can be either an event (asynchronous) or a triggered operation (synchronous) or a timeout trigger (time-based behaviour). Modelling of continuous

behaviour e.g. a PID controller is not directly supported by the Rhapsody graphical user interface (GUI). Rhapsody offers direct support for several real time operating systems (real time environments). Therefore it provides sets of pre-defined primitives for defining primitive concurrency and synchronisation objects like mutexes, semaphores or timers. These reactive object types are defined outside the system and cannot be modified.

3.3 Application Example

According to the requirements a simplified model of the hydraulic press using the tool Rhapsody C++ (3.0.1 Windows-based environment) is modelled. The model of the hydraulic press (figure 2) consists of several distributed controllers which run on different automation devices (resources). Therefore active objects with active concurrency are used for the design. Communication is realized by asynchronous and synchronous messages. Messages and triggered operations were used in the control state charts. The aggregations show the relationship between objects and map their communication channels. The operation control of the hydraulic press processes operator inputs (set values) and displays continuously its real values. The real process communication between the operation control and each individual frame is realized via field bus. At an early design phase the field bus is represented as a simple aggregation in the model.

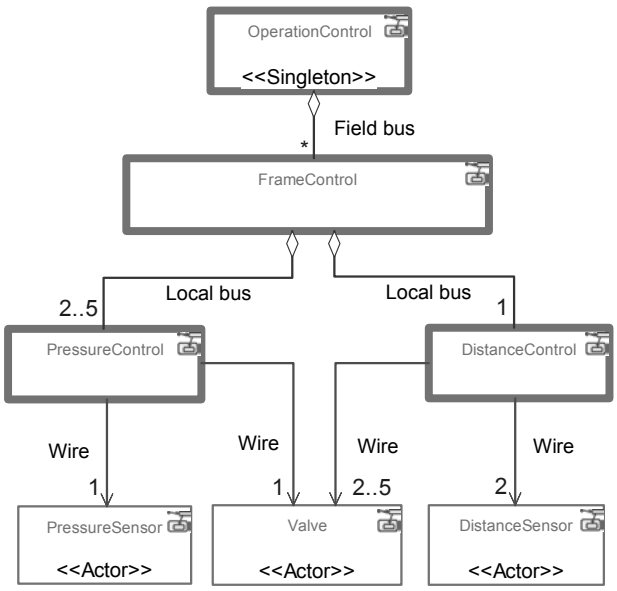

Figure 2 Class diagram of hydraulic press control

If a control mode change is necessary the operation control needs to synchronise all frames so that they switch at the same time to the same mode. The synchronisation process need to be synchronised in a specific time slot (hard real time) due to technological reasons .
Time requirements are modelled as software timeout triggers, which are provided by the tool. In case of failure all frames are switched off by the operation control. Every frame has got one frame control with several subordinated PI-controllers. The frame control is primarily responsible for state monitoring and mode switching. If, because of technological reasons, a change of control mode in one frame occurs, it must be synchronised within the frame by the frame control.
Usually the hydraulic press runs in the distance control mode. Dependent the hydraulic configuration one frame consists of two up to five hydraulic systems. Each hydraulic systems is controlled by one pressure controller. The frame is controlled by one distance controller. Standard multiplicity notation was used to consider the specific configuration of each frame. All PI-algorithms are directly implemented in C. For technological reasons there is a strongly defined time slot of 30 ms for the closed loop controller output (hard real time requirement). In the model this requirement is solved by the software timeout triggers. A PI-controller state chart is shown in figure 3. The pressure controller gets the input value from the pressure sensor of the hydraulic system, calculate new values and send them to the valve. The closed loop distance controller gets its input from the correspondent distance sensors (one or two) and sends up to five calculated output values, dependent on the current frame configuration, to the correspondent valves.

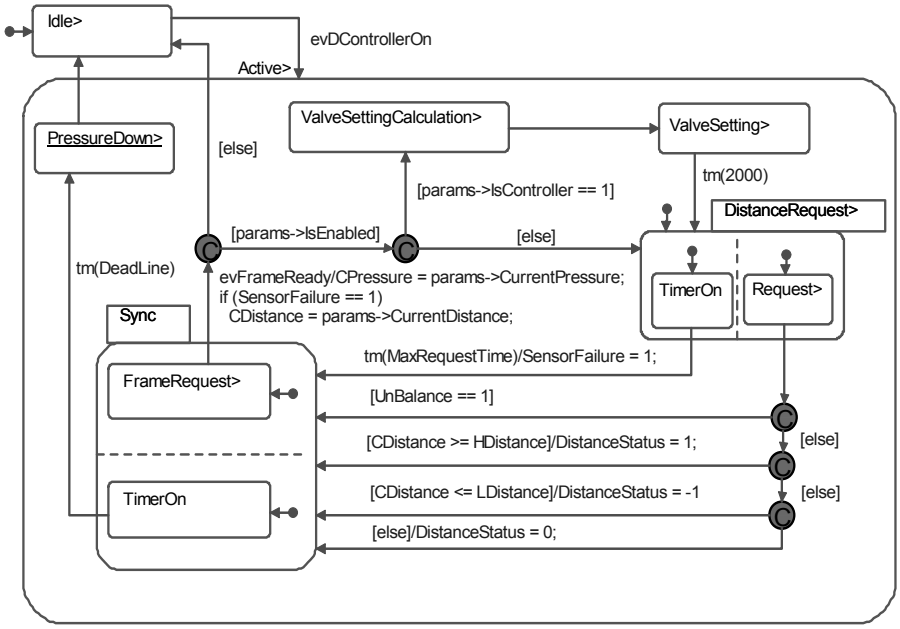

Figure 3 State chart of PI closed loop controller

Both controllers always run either in control mode or in observer mode. Either the pressure controller is in control mode or the distance controller. In the control mode a controller keeps a set value within a tolerance, in observer mode it only observes the appropriate measured value and informs the frame control in case of mode switch. Note that only one controller can access one valve at the same time, this can be mainly critical during the mode switching. The communication between controller and process periphery is released with synchronous messages (triggered operations). The using of mutex constructs within triggered operations allows the exclusive access required for valves in the model. Generally only the interfaces of the process periphery (sensors and actuators) are used in the model. Because of simulation reasons the sensors and actuators are built as reactive objects, which run on the system thread. The connections between the frame resource and the process periphery are mapped according to associations in the model. The complete communication between controllers is released with asynchronous messages. Rhapsody supports asynchronous messaging as events.

3.4 Evaluation

Rhapsody offers the so called browser view that manages all available UML diagrams and project resources. It allows already after a short training period an efficient and well-structured working with the graphical user interface (GUI). The variety of property settings seams to be confusing at first. The comparison of user defined sequence diagrams and generated sequence diagrams makes debugging easier. Further advantages are the modelling of concurrency within one state chart and that more than one component can run simultaneously in an animated session.
The usage of software time trigger is not recommended if an accurate timeout is needed. Time can be distorted if one task receiving the timer event is prevented from running by another task. Also the time until timeout events are consumed, depends on the number of events in the appropriate event queue. If an accurate timer is needed a timer class based on a hardware timer can be created. Rhapsody also supports the timer stereotype and some others as the so-called wrapper classes. These predefined reactive classes for defining simple concurrency and synchronisation objects only encapsulate the functionality of the underlying operating systems. The association class construct is a simple way to set the properties and functionality of the aggregation between the operation control and the frame controls, which corresponds to the physical field bus. In this case a field bus failure could also be modelled. But the tool does not support the association class construct. In this case the more complicated way to define a new additional field bus class can be used. The synchronisation and exclusive access are only supported by the wrapper classes, with all their expansion and flexibility problems. In this case the more efficient modelling way would be the usage of active classes with specific functionality. Rhapsody supports model debugging at run time by an animation tool that uses generated and compiled source code for object creation . The dynamic behaviour of the runtime objects could be observed in animated sequence diagrams and state charts. The runtime interpretation of associations between objects depends on the creation sequence of associations between their classes at design time. Therefore more than one runtime interpretation of the same model is possible. In this case the animation results are not repeatable and wrong interpretations may result. A system crash may

occur, if concurrency of several object instances occurs during the animation. The tool allows to run only one task at runtime. Thereby the created components run as threads within this task. This reduces the simulation opportunities.

Table 4 Real time requirements - valuation of Rhapsody

Useful and Necessary Language Constructs for Real Time Programming	**Rhapsody (C++)**	**Useful and Necessary Description of Hardware Constructs**	**Rhapsody (C++)**
Task dispatch	+	Connection between peripheral device and technical process	-
Transition control between different states of a task	-		
Scheduling	-	Modelling of input/ output	0
Synchronization of tasks (semaphores)	-	Description of different process computer units	-
Task activation (time/ event)	-		
Communication between tasks	-	Connection between different computers	-
Interrupts	-		

+	special language constructs
0	no special language constructs
-	not possible

Useful and necessary language constructs for real time programming are task set, transition control between different states of a task, scheduling, synchronization of tasks, task activation, communication between tasks and setting of interrupts. In the following these aspects are described (table 4). A component definition corresponds to a task definition (task dispatch), because a modelled component runs as a task at runtime. More than one components are linked to one executable task (.exe or .dll on windows system). Rhapsody offers stereotypes for scheduling and synchronization on thread level only. Rhapsody allows changing states of threads, the activation of threads as well as setting of priorities on thread level. This is only possible using predefined operations during implementation. There are no special modelling constructs for manipulation of threads. In addition the tool does not provide constructs for task activation or other task manipulations. The C++ version does not offer any support during description of hardware constructs (table 4). In addition Rhapsody supports automatic code generation from model in: C, C++ or Java. The required source code IEC 61131-3 implementation is not available, which was not a prerequisite for the tool selection. Because Structured Text (ST) one of the IEC 61131-3 is very close to PASCAL it might be possible to convert the generated code.

4 Conclusion

In the forefront of this evaluation we analysed the tool Rational Rose Real Time, which is based on UML/RT developed by Selic and Rumbaugh [SR98]. They introduced new constructs to make the modeling of real time systems easier, based on the UML extensibility constructs. The idea was, that these constructs, general UML concepts and diagrams would provide a toolset to design embedded real time systems. The result of this evaluation is, that this UML/ RT tool is not yet applicable in the application development of automation not only because of missing aspects of the UML specification, but also because usability lacks[FV02]. The differences of Rose RT (UML/RT) and Rhapsody (UML) will be discussed next. Rose RT based on UML/RT should offer special constructs for modelling real time systems. Overall Rhapsody gives adequate possibilities for modelling of real tine systems.
Rhapsody shows some of the same shortcomings as Rational Rose RT, e.g. the lack of usability for engineers. (chapter 3.4). The necessary improvement of UML as concept and notation is mostly realized in the UML 2.0 specification, which includes e.g. timing and communication diagrams and mutex and model driven architecture [JRH04]. Nevertheless the usual tools do not provide all possibilities of the UML specification. Another reason against adaptation in industry is the universal applicability of UML. UML allows modelling complex systems in different fields of application. For that it is a wide open standard, which is in general an advantage. But for training of mechanical and electrical engineers the capability of UML and the number of diagrams is a disadvantage. .
For this reasons an adaptation is necessary. Different possibilities are the development of design patterns, ‘best practice’ or the definition of new elements for process automation based on the extensibility constructs of UML similar to the development of UML/RT.
Closing it could be assumed that UML with iLogix Rhapsody allows the modelling of a specific process automation application, but it is not adequate for the software engineering workflow and the engineering personal in industry. Additionally the required IEC 61131-3 code generation is not supported neither yet by any UML tool.

5 Prospects

The necessary adaptation of UML is the field of further work within the Project DisPA [Dis04], DFG SPP 1064 [DFG03]. In this project we try to integrate techniques of software specification with aspects of engineering. Therefore the most important diagrams (class, structure, state and sequence) should be selected for an automation-specific derivation of UML. As another result the requirements of an UML based engineering tool should be listed and discussed. Therefore UML tools need to be evaluated.
Presently we evaluate the UML tool RealTime Studio from Artisan [Ar03]. Another interesting tool is Tau G2 from Telelogic, which will be evaluated next.
As described in chapter 3.1 a further version of UML with special constructs for performance, time, and scheduling is adapted [OMG03]. Nevertheless after the final draft, it is necessary to test a tool which supports modeling with UML 2.0.

Another field of work is the mapping between IEC61131-3 and UML widely discussed [BF03].

References

[Ar03] Artisan Software Tools, Inc., http://www.Artisan.de, 2003.

[BF03] Bonfe M., Fantuzzi C. , Design and Verification of Industrial Logic Controllers with UML and Statecharts. In: IEEE Conference on Control Application in Istanbul, Turkey, June 2003

[BMS97] Bonfatti, F., Monari, P. D., and Sampietri, U.: IEC 1131-3 Programming Methodology. CJ International, Seyssins, 1997.

[BV02] Biermann, C., Vogel-Heuser, B.: Requirements of a process control description language for distributed control systems (DCS) in process industry. In: Proceedings of IECON'02, 28th Annual Conference of the IEEE Industrial Electronics Society, Sevilla, November 2002.

[DFG03] DFG SPP 1064, http://tfs.cs.tu-berlin.de/SPP/, 2003.

[Dis04] http://www.lfa.uni-wuppertal.de/DisPA

[FV02] Fischer, K., Vogel-Heuser, B., UML for real-time applications in automation. In: Automatisierungs-technische Praxis (atp) 44 (2002); Heft 10,Oldenbourg, S. 63-69.

[FV03] Friedrich, David; Vogel-Heuser, Birgit; Bristol, Edgar: Evaluation of Modeling Notations for Basic Software Engineering in Process Control. In: 29 th Annual Conference of the IEEE Industrial Electronics Society (IECON 03) in Roanoke, Virginia, USA, November 2003.

[Gr00] Große-Rhode, M., Using a formal reference model for consistency checking and integration of UML diagrams. In: Tanik, M.M., Ertas, A. (eds), Proc. 5th World conference on Integrated Design and Process Technology. Society for Design and Process Science, Dallas, 2000.

[HP87] Hatley, D. J., Pirbhai, I. A., Strategies for Real-Time System Specification. Dorset House, New York, NY, 1987.

[Iv04] iVA, http://www.iva.ing.tu-bs.de

[JRH04] Jeckle, M.; Rupp, C.; Hahn, J.; Zengler, B.; Queins, S.: UML 2 – glasklar. Hanser Verlag, München, Wien, 2004.

[KFV03] Katzke, U.; Fischer, K.; Vogel-Heuser, B.: PEARL 2003

[OMG03] OMG: Object Management Group, http://www.omg.org, 2003.

[Pa98] Partsch, H., Requirements-Engineering systematisch, Springer-Verlag, Berlin, Heidelberg, 1998.

[RJB99] Rumbaugh, J., Jacobson, I., and Booch, G., The Unified Modeling Language Reference Manual, Addison-Wesley Longman, 1999.

[Ru91] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., Objektorientiertes Modellieren und Entwerfen, Hanser, München, 1991.

[Sp04] Sprachreport PEARL 90, http://www.real-time.de

[SR98] Selic, B., Rumbaugh, J., Using UML for Modeling Complex Real-Time Systems, http://www.rational.com\ whitepapers,1998.

[Te03] Telelogic, http://www.taug2.com/whytelelogic/presskit/ TauG2_Press_Release.pdf, 2003.

[UML03] UML 2.0, communityUML, http://www.community-ml.org/UML2.htm, 2003.

[VB01] Vogel, B., Bartels, J., Systementwicklung für die Automatisierungstechnik im Anlagenbau, at-Automatisierungstechnik Vol 5, Oldenbourg, München, 2001, pp. 214-224.