A Model-Based Method to Develop PLC Software for Machine Tools

M. F. Zaeh[1], C. Poernbacher[1]
[1]Institute for Machine Tools and Industrial Management
TU Muenchen, Garching, Germany
Submitted by J. Milberg (1), Garching, Germany

Abstract
It is difficult to develop PLC software for modern machine tools due to their increasing functionality and the resulting complexity. One approach to managing this is the model-based development and simulation-aided verification of control software. A suitable method for this is described in this paper. It focuses on a method for modeling the control functionality and a simulation system for commissioning the developed programs using a virtual model of the machine.

Keywords:
Concurrent engineering, Mechatronic, Virtual machine tool

1 INTRODUCTION

The development of highly productive machine tools is characterized by the need for high availability and an increased functionality and level of automation. This leads to more complex machines, especially as the software for triggering, coordinating and monitoring the individual machine functions is becoming complex. To master this complexity, it is necessary to use modern development and simulation tools and to redesign established methods [1]. While static and dynamic machine structure design using Finite-Element-Method and Multi-Body-System calculation is already the state of the art in industrial environments, a model-based development and a simulation-aided validation and optimization of machine tool control software still do not play any role in practice. This truly applies for the Programmable Logic Control (PLC) software that implements technological and safety-relevant functions. Reasons for this include the late integration of software engineering in machine development as well as the lack of suitable development tools. The situation is additionally aggravated by the insufficient coordination of the involved specialist disciplines and the great effort needed to create the necessary digital models. These problems result from a lack of suitable interdisciplinary means of description for specifying the machine tool and a uniform method for developing the control software [2].

2 KEY DEMANDS ON A MODEL-BASED DEVELOPMENT ENVIRONMENT

A model-based development of PLC software requires suitable means of description and development methods to be available and integrated in a common development platform. In addition, a practical approach must meet a number of requirements.

Due to the high complexity of modern machine tools, a one-sided, software-oriented perspective is no longer sufficient to develop PLC programs. Instead, the characteristics and the structural layout of a machine tool must already be considered at the beginning of development. The behavior of the electromechanical, electrical, hydraulic and pneumatic components of the machine (referred to as hardware components in the following) as well as the software components and their interaction must be modeled with sufficient precision and used to develop the control functionality.

The different perspectives of the involved disciplines must be taken into account in selecting the necessary means of description. A design engineer is primarily interested in the processes of the machine, the design of the machine components and ensuring their collision-free assembly. A software developer has a predominantly signal- and logic-oriented perspective. This person's focus is on clearly determining the possible machine states and designing trouble-shooting mechanisms. In the presented approach therefore, modeling techniques from the development of application software and embedded systems are used together with a three-dimensional kinematic model of the machine.

A software developer must already be able to validate and optimize the developed functions in early development phases. This requires simulation tools that allow the software to be verified and commissioned with a virtual model of the machine tool. For flexible adaptation to the assigned development task, the simulation must permit simple processes to be tested and visualized with a three-dimensional model and allow actual machine and control components to be integrated.

In order to minimize the modeling effort, it must be possible to reuse previously generated information and to gradually refine the model components during the course of development. This basically requires a formal or semi-formal representation of the machine tool system. As a result, mechanisms for checking the consistency and verifying the model can also be implemented [3].

3 BASIC CONCEPTS FOR MODELING MACHINE TOOLS

A system-theory approach is proposed for modeling machine tools. The ‘system’ machine tool, i.e. the PLC software and the hardware components of the machine tool, is structured hierarchically. Each of the subsystems has defined inputs and outputs that are connected with each

other according to the flow of information, material and energy between the individual subsystems. In addition, a defined behavior can be assigned to each subsystem. The most important notation elements selected for the proposed approach and their use in system modeling are described in the following. A simple example, showing the use of the described concepts and methods is shown in the following figures.

3.1 Modeling of the machine tool structure

To represent the structural layout of the machine tool, notation elements and diagrams of the Unified Modeling Language (UML), a standardized notation language used for developing application software and embedded systems [4], are employed (see Figure 1). Each subsystem is represented here by a component. Hierarchical structures can be represented by nesting several components. Each component has a number of interfaces through which it can exchange information with other system components. These interfaces are represented by ports. While the ports merely describe the information that is needed or provided by a component, the information flow in the system is represented by connectors. For this purpose, corresponding ports are connected to each other. The assignment of components to particular namespaces determines their affiliation within the machine tool. Physical interfaces between the PLC and the hardware components are displayed using the so-called lollipop notation.

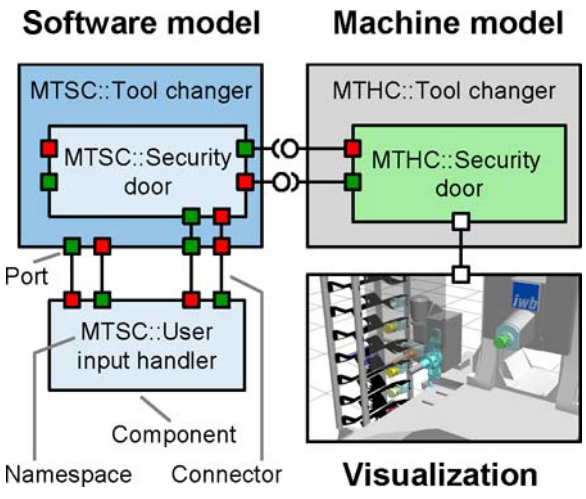

Figure 1: Modeling the machine tool structure.

Unlike existing approaches to developing automation systems, modeling strictly distinguishes between machine hardware and control software. This offers several advantages:

- The interface between the control and the hardware components of the machine is represented.
- Mixture between machine-specific and control software behavior is prevented. This simplifies the further use and reuse of the generated models to create hardware-specific control code [5] and the simulation-based testing of the developed programs.
- Discrete means of description are used to model the behavior of the PLC software.
- Hybrid means of description [6] are used to represent the behavior of the hardware components. They make it possible to specify continuous and discrete behavior characteristics (see section 3.2).

3.2 Modeling of the machine tool behavior

The PLC of a machine tool exhibits a discrete system behavior in accordance with its cyclic functioning. Finite automatons are therefore suitable for modeling the behavior of software components. The use of state machines [7] in particular is recommended (see Figure 2). These are also part of the UML and provide an adequate view of the machine tool to software developers.

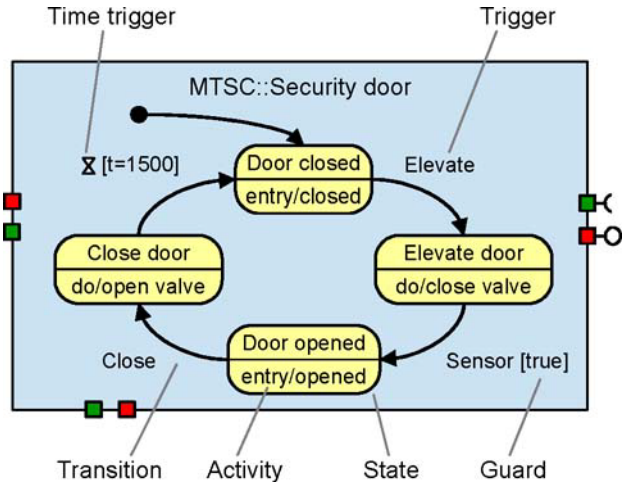

Figure 2: Modeling the software component behavior.

A state machine is defined by a finite number of states, transitions, triggers and guards. It has exactly one active state at any point in time. Every state is declared with a name that is clear in the structural context. The actions (activities) to be executed upon activation are annotated accordingly. These describe the information exchange among the components of the software model. They are also used to declare simple arithmetic and logic operations. According to their execution, three types of activities are distinguished. 'Entry' and 'exit' activities are performed once, when a state is entered or exited respectively. 'Do' activities, on the other hand, are repeated until the next state is entered.

Transitions, triggers and guards are used to specify the changeover from a source state to a target state. While transitions only define the states among which a state machine can change, triggers describe the events that are necessary for such a change. In PLC programs, these are changes in the signal states of a machine (sensor values), calls within the control, or user entries. So-called 'time triggers' are a special form of triggers. They can trigger a transition after or at a defined time. A trigger can be assigned additional constraints. These supplementary conditions, called 'guards', must be fulfilled for a state changeover to be possible.

When the PLC software of a machine tool is developed based on a model, the behavior of the hardware components must also be modeled sufficiently. Above all, the time behavior, the logical behavior and the interference behavior of the machine tool components are of interest. They cannot be suitably represented by discrete, graph-oriented means of description. Instead, a block-oriented modeling of the machine behavior using timing elements, simple mathematical functions, Boolean logic, lookup tables (discrete assignment of input values to output values) or their programming with script languages is favored (see Figure 3). Suitable modeling elements are selected according to the necessary degree of detail and the complexity of the system behavior to be represented.

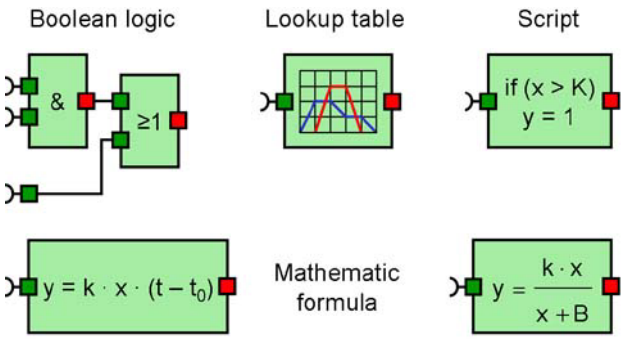

Figure 3: Modeling the hardware component behavior.

A three-dimensional visualization of the machine behavior coupled with the machine model makes it easier to specify and verify the developed control functions. Representing the motion behavior of the machine components facilitates the understanding and discussion of the machine functionality. This also makes it possible to detect collisions and perform construction space and achievability analyses, even in early development phases.

4 INTERDISCIPLINARY SPECIFICATION OF THE REQUIREMENTS FOR THE PLC SOFTWARE

The requirements for the functionalities to be implemented are determined and structured during the analysis phase. In practice, this is done with oral agreements and domain-specific documents. A different system understanding of mechanical design and software development as well as misunderstandings can lead to aberrations that are not detected until the software is implemented on the real machine. The coordination can be improved by an early, formal specification of the functionalities to be implemented. The use of sequence diagrams combined with a three-dimensional kinematic model and a model of the machine hardware components is recommended for this purpose. The behavior of the hardware components is represented as described in section 3.2. A three-dimensional visualization of the behavior clearly depicts the functionality of the machine to be implemented.

4.1 Sequence diagrams

Sequence diagrams are suitable for formally specifying the functions to be implemented. They provide a view of the machine tool model and classify behavior-specific aspects of the observed system in a structural context. They focus on representing the interaction among the individual system components. Sequence diagrams (see Figure 4) describe interactions in two dimensions. The communication partners are arranged from left to right. An imaginary time axis, the lifeline, runs from top to bottom. Communication among the subsystems takes place in the form of messages. These are represented by arrows and described by an annotated name. A message begins on the lifeline of its sender and ends on the lifeline of its recipient. If a message is received from a sender that is unknown or outside of the observed context or is sent to a corresponding recipient, it will not be displayed in the diagram.

Activities are displayed in sequence diagrams by a thickening of the lifeline. These markings are especially suited for representing the machine behavior in the context of specifying PLC functionality.

Preconditions for an interaction can be represented in a sequence diagram by so-called state invariants. Communication can therefore only take place when the sender or recipient is in the corresponding state. In addition, it is possible to specify time constraints in sequence diagrams. This makes it feasible to distribute time demands for particular machine functions that are determined by the customer to individual subfunctions. These demands can be used as test conditions in software verification.

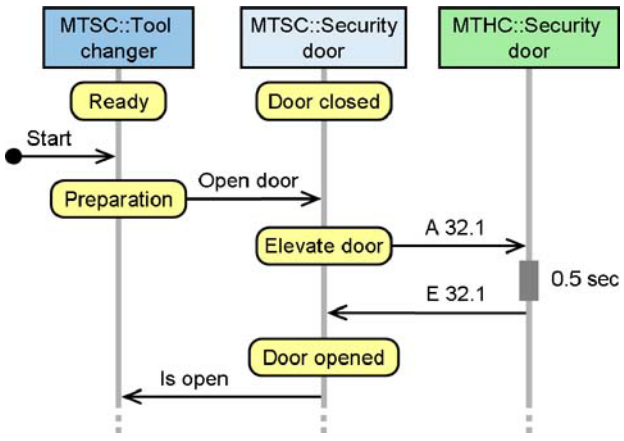

Figure 4: Sequence diagram.

The relationship of the sequence diagrams to the model elements of the state machines becomes clear upon closer consideration. Transitions, triggers and guards are represented by incoming and outgoing messages as well as by the state invariants that they pass through along the lifeline. The state invariants correspond to the states, and sent messages can be interpreted as activities.

4.2 Construction of sequence diagrams

The first step for defining a functionality to be implemented is to identify the involved system components and their structural arrangement in the machine tool. The key components are assembled in a structural model according to Figure 3. In a further step, the behavior of the hardware components is determined as described in section 3.2. If the hardware has not been permanently implemented yet, the machine behavior can be defined in a simplified way. Mathematical relationships that define the time-dependent position values are especially suited to representing motion-related functions. Elements of Boolean logic are suitable for displaying static functions such as the triggering of sensors.

These elements are subsequently coupled with a three-dimensional kinematic model of the machine. Motions are visualized by directly coupling position values to kinematic machine axes. The static functionality is represented by changing the visualization of the corresponding components in the kinematic model.

After the behavior of the hardware components of the machine has been determined, the actual specification of the software functionality can take place. State invariants, messages and activities are modeled step by step according to the functionality to be implemented.

Merely defining individual machine functions is not sufficient for modeling the PLC software and hardware components of a machine. The created models must be further detailed and refined over the course of development. This enables them to be used further for the subsequent phases of development. While the models of the software components serve as a basis for implementing the PLC programs, the models of the hardware components and the kinematic model of the machine tool can be used to validate, verify and optimize the developed control programs. A suitable system for this is presented in the following.

5 SOFTWARE VALIDATION AND OPTIMIZATION USING HARDWARE-IN-THE-LOOP SIMULATION

Until now, implemented PLC programs have usually been verified and optimized using an actual prototype of the machine. Errors in the software are identified and corrected on the spot. This is done because the control software only functions correctly when it interacts with the electromechanical, electrical, hydraulic and pneumatic components of the machine. A preliminary system test is therefore not possible [8].

This approach makes it difficult to extract knowledge from the software implementation for use in the machine development. The enormous time pressure in verifying and optimizing the developed programs also leads to a lower software quality and high consequential costs. The goal is therefore to commission the control software using a virtual model of the hardware components of the machine tool (see section 4).

Figure 5 shows the principal structure of a Hardware-in-the-Loop (HiL) simulation system that is partly based on commercial components and was developed at the Institute for Machine Tools and Industrial Management (*iwb*). The hardware used consists of a numerical control (NC) with integrated PLC, machine control panel (MCP) and operator panel (OP). With the exception of the drive control modules, the complete control hardware, including all user interfaces, is thus integrated in the simulation system. The electromechanical, electrical, hydraulic and pneumatic components of the machine tool are simulated on a standard PC. The real control hardware communicates with the simulation model via the PLC and an interface card integrated in the simulation PC. The behavior of the machine tool is visualized using a three-dimensional kinematic model of the machine.

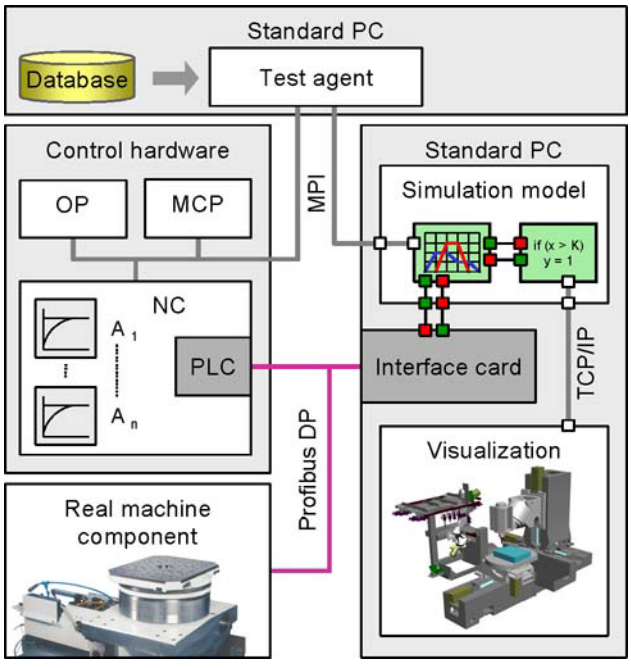

Figure 5: Principle structure of the simulation system.

The fieldbus system Profibus DP is the basis for communication between the control hardware and the simulation model. This standard bus system provides independence from the control hardware and enables real hardware components to be integrated.

The simulation models must have real-time capability, in order for real control hardware to be integrated in the system. This is made possible by the low performance requirements of the simulation and the decoupling of the three-dimensional visualization from the logical and time behavior of the simulation model. Experiments have shown that a cycle time of t_{SIM} = *10-20 ms* is attained when a milling center is simulated on a standard PC. Since the cycle time t_{PLC} of modern machine tool PLCs is within the range of *50-150 ms*, the requirement of real-time capability can be considered to be fulfilled. This requirement cannot be realized for the feed axes of a machine tool, however, due to the NC interpolation rate of t_{NC} = *1-4 ms.* The feed axes are therefore simulated within the NC by a PT1 element. To simulate the hardware components and visualize the feed axes movements, the actual position values are provided by a special software component integrated in the NC kernel. The simulation of feed axes based on an HiL simulation system is the subject of current research [9].

A virtual machine operator (test agent) is also integrated in the simulation system to automatically work through predefined test scenarios. The test agent has direct access to the human-machine interface and the control. This makes it possible to simulate the interaction between the control and the machine operator and to record their reactions. The test agent can also access the simulation model of the hardware components of the machine. It is thus possible to trigger specific interference situations in the virtual model of the machine tool. This allows troubleshooting functions of the control to be tested. Such experiments are not possible with a real machine tool because of the risk of machine damage.

6 SUMMARY

This paper has presented a new method for the model-based development of PLC software for machine tools. The focus was on describing a method for an interdisciplinary definition of the machine functions to be implemented and on a Hardware-in-the-Loop simulation system for testing and optimizing PLC software.

7 REFERENCES

[1] Weck, M., Queins, M., Witt, S., 2003, Effektive Entwicklung von Werkzeugmaschinen, VDI-Z, 10:32-36.

[2] Tomaszunas, J.-J., 1998, Komponentenbasierte Maschinenmodellierung zur Echtzeit-Simulation für den Steuerungstest, Dissertation, TU München.

[3] Fischer, K., Vogel-Heuser, B., 2002, UML in der automatisierungstechnischen Anwendung – Stärken und Schwächen, atp Automatisierungstechnische Praxis, 10:63-69.

[4] Selic, B., Rumbaugh, J., 1998, Using UML for Modeling Complex Real-Time Systems, www-136. ibm.com/developerworks/rational/library/139.html.

[5] Braatz, A., 2003, From Model to Code – Generation of PLC-Software from UML, SPS/IPC/DRIVES, 14:195-204.

[6] Bertenkötter, K., Bisanz, S., Hannemann, U., Peleska, J., 2004, Spezifikation von Echtzeit-Automatisierungssystemen mit HybridUML, atp Automatisierungstechnische Praxis, 8:54-60.

[7] Booch, G., Jacobson, I., Rumbaugh, J., 2004, The Unified Modeling Language Reference Manual, Addison-Wesley, Boston.

[8] Zaeh, M.-F., Ehrenstrasser, M., Poernbacher, C., Wuensch, G., 2003, Emerging Virtual Machine Tools, Shimada, K., eds., Design Automation Conference, ASME Design Engineering Technical Conferences, Paper No. DETC2003/DAC-48756.

[9] Pritschow, G., Roeck, S., 2004, “Hardware in the Loop” Simulation of Machine Tools, Annals of the CIRP, 53/1:295-298.