

Study regarding the benefits of object orientation in automation technology

→ Significant potential for optimization during the development of automation software exists in the following areas: Improvement of software quality, cost reduction through reusability of software components and modularization, improved communication between different groups of persons involved in the development, and integrated usability of tools and methods.

UML for control programming

A prototype implementation at the Faculty for Process Informatics at Bergische Universität Wuppertal was used to examine how object orientation and Unified Modeling Language (UML) can be used to improve the engineering process. Beckhoff hardware and software were used as automation components.

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

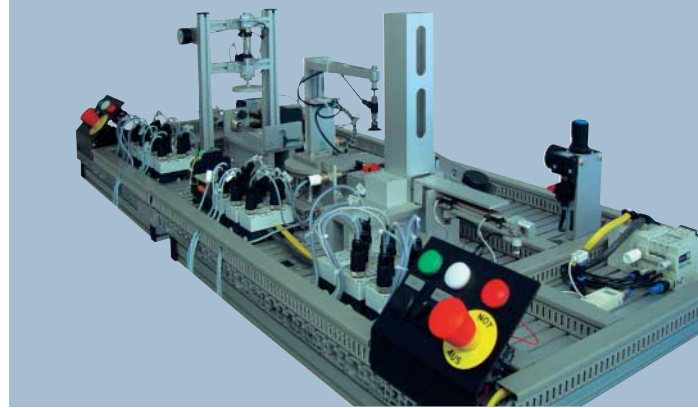
Depending on the project phase, different tools and methods are used. In the worst case, the corresponding data have to be re-entered during the transition from one phase to the next, because no appropriate interfaces between the individual tools are available. The ideal to strive for would be a higher-level tool that consistently provides all system information in a model and enables the design to be realized both at an abstract level and in a conventional environment (e.g. E-CAE, IEC 61131-3). However, considerable effort is required to achieve this. Part of the idea was implemented in a prototype with comparatively little effort.

The prototype enables a complete and executable IEC 61131-3 project including project design information for a system example to be generated from a UML model.

Principle of code generation in the system example

A UML strategy designed for modeling embedded systems 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 software design.

Real-Time Studio from Artisan was used as a 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 interfacing. Based on the above strategy, a system example was modeled using this UML tool. From this model, a code generator developed at Bergische Universität Wuppertal in Germany automatically creates the IEC 61131-3 code (SFC and ST) and derives project design information. The code and the project design information derived



from the UML model are automatically imported into the TwinCAT PLC programming environment or into the TwinCAT System Manager, fully and correctly automating the system example.

The UML model – mapping to IEC 61131-3

Three special class types with relevance to automation technology were used in the modeling: entity, control and service classes. Entity classes are used for central data management. They carry many attributes and few or no methods.

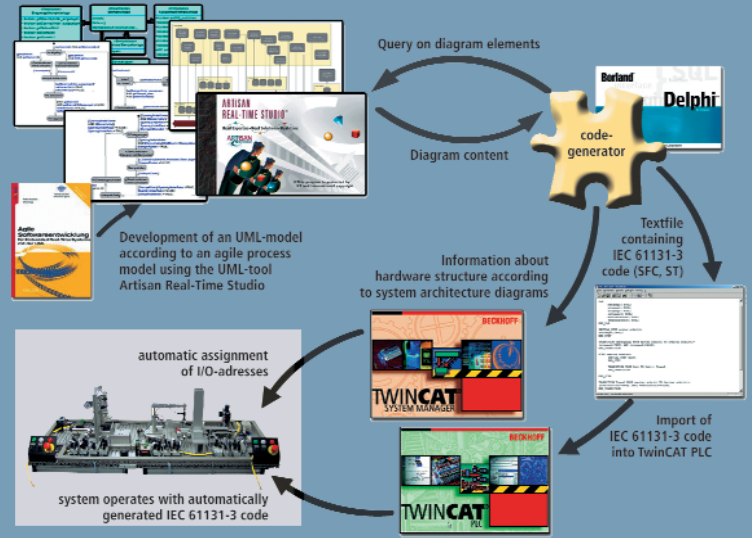
Unified Modeling Language (UML)

UML is a language for specification, graphic display, design and documentation of object-oriented software systems, business models and non-software systems. It aims to be a generally comprehensible basis for discussion between different persons involved in a project during system design and development. UML 1.x uses 9 diagram types for representing issues from different perspectives. Given the large number of available diagrams and options, using UML successfully requires an appropriate strategy and a suitable UML tool. The strategy determines which diagram should be used when and how.

Beckhoff cooperation with universities

Beckhoff has been supporting work at different universities and technical schools for some time. The core of the automatic code generation from a UML model was developed as part of a dissertation by Daniel Witsch. The code generator has already been developed further. This topic will be examined further in future.

Principle of code generation



This system example represents a manufacturing process in which workpieces are processed differently depending on the results of a material analysis (inductive, optical). Light and metallic workpieces are stamped on the left-hand side of the system; workpieces made of darker plastic are separated out. The crane on the right-hand side of the system is used to transport the workpieces within the system.

Reusable functionalities are consolidated in a service class. Examples are repeatedly required control operations, controllers or mathematical functions. Control classes deal with the central control of a task. Within the class diagram, they play a conductor role, using the variables from the entity classes and methods from the service class for delegating tasks.

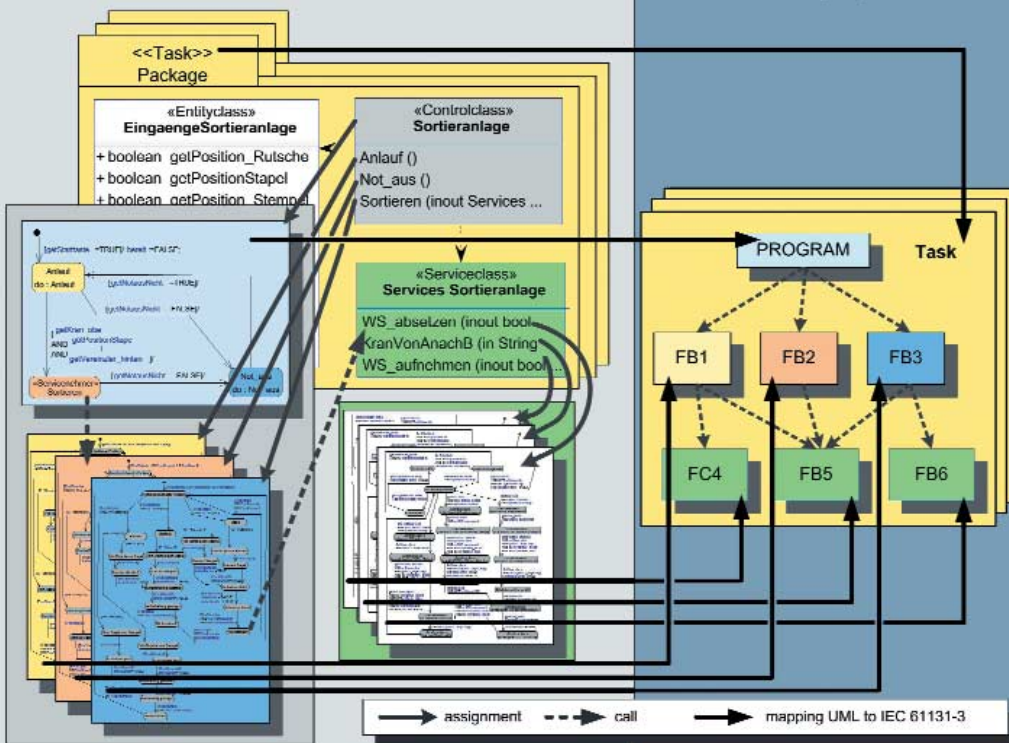
In the UML model, each task is represented by a package. A package contains a class diagram consisting of control, entity and service classes. This class diagram forms the static task model.

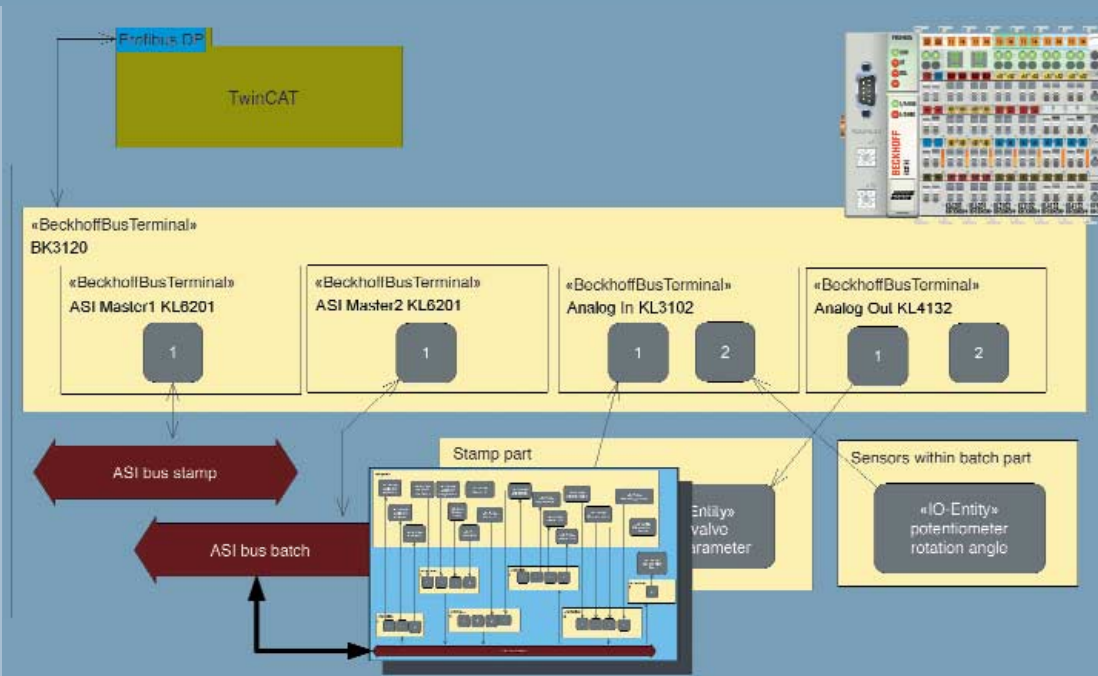
Each method within a class is assigned a state automaton (fig. at the bottom) that defines its dynamic behavior. Another state automaton describing the interaction of its methods is assigned to the control class itself. This state automaton becomes the main program of the task. The main program calls the methods of the control class, which in turn can access the 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.

UML model in Artisan Real-Time Studio

IEC 61131-3 project in TwinCAT

Mapping of the UML model to IEC 61131-3





System architecture description as a distribution diagram. Actuators and sensors coupled to a bus are shown in a separate diagram.

Integration of the hardware

For modeling of the hardware, Artisan Real-Time Studio offers so-called system architecture diagrams (SAD). These correspond to the distribution diagrams within the UML. The SADs are used to map the existing hardware components, actuators and sensors. The connection of the actuators and sensors with the inputs and outputs of the control hardware is indicated by arrows (fig. at the top). The variables used in the class and state diagrams are assigned to the actuators and sensors. If the system is re-wired, this information can automatically be transferred to the projecting tool of the TwinCAT environment, thus establishing the connection between hardware and software projecting.

Re-wiring of a sensor can be affected by changing an associated arrow in the SAD. The re-wiring is thus automatically considered correctly in the software.

Outlook

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 only become apparent during application in a more complex system. It is therefore necessary to expand the modeling aspect of this procedure to systems of any size and to consider mechanisms such as inheritance. This will significantly simplify the administration of variants and modules. Appropriate concepts are already available.

One main argument against code generation from a model is the lack of a return route. One requirement is, for example, for the commissioning engineer to be able to make changes directly in the IEC 61131-3 code, and for those changes to be returned consistently to the overall 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. If such constructs were included in the standard, re-conversion and application of UML without semantic breaches would be possible. Until such time, the approach of simultaneous representation of UML model and IEC 61131-3 code can provide concrete assistance.

The authors: Daniel Witsch and Univ.-Prof. Dr.-Ing. Birgit Vogel-Heuser

→ Faculty for Process Informatics, Bergische Universität Wuppertal
www.lfa.uni-wuppertal.de/uml2IEC61131