Efficient Control Software Design for Automated Material Handling Systems Based on a Two-Layer Architecture

Markus Spindler*, Thomas Aicher**, Daniel Schütz**, Birgit Vogel-Heuser**, Willibald A. Günthner*
*Institute for Materials Handling, Material Flow, Logistics, Technische Universität München, Germany
(e-mail: {spindler, kontakt}@fml.mw.tum.de)
**Institute of Automation and Information Systems, Technische Universität München, Germany
(e-mail: {aicher, schuetz, vogel-heuser}@ais.mw.tum.de)

***Abstract*—Combining high throughputs and good process quality, automated material handling systems are widely used in industry. In order to fully utilize their potential, automated material handling systems have to be highly specialized and custom-built, requiring individually implemented control software. Due to the fact that a modular structure of the software is often missing, individual implementations require extensive manual programming, which is labor intensive, error prone and leads to software structures which are hard to maintain. In this paper, a software architecture is presented that enables a modular composition of control software for automated material handling systems. This software architecture is based on a two-layer architecture and separates the decisions made by the material flow from the control of the conveyors. As a result of this separation, predefined modules for both layers can be created, therefore allowing an integrated modularization of the control software. Such an integrated modularization enables the creation of a software by interconnecting predefined modules and setting their parameters, and is therefore efficient in terms of time and labor.**

Keywords — Intralogistics, material handling systems, model driven engineering (MDE), automation, code generation

I. Introduction

Automated material handling systems (AMHS) are widely used in industry, distribution and transport due to their short throughput times, high handling capacity, and good process quality. As a result of increasing e-commerce and thus easy accessibility of consumers, the business of distributors is shifting from business-to-business (B2B) to business-to-consumer (B2C). This shift is characterized by greater shipment volumes and more demanding requirements with respect to process quality and delivery time, because end customers order in case of immediate need and therefore require a fast shipment. Since these requirements can be fulfilled best with AMHS, there is a growing demand for these systems, and thus an increasing need to effectively project such systems.

One important task when projecting AMHS is engineering the control software for controlling the material handling equipment. To enhance this engineering process, a concept for supporting the engineering of AMHS is being developed as part of our project aComA – Automated Code Generation for Modular Systems. The basis for such a concept is a corresponding software architecture that focuses on the special needs for AMHS and therefore supports the engineering of the control software in an optimal way. The focus within our aComA project lies on stationary AMHS. Stationary AMHS means that the material handling equipment is affixed to the ground and therefore the conveyors cannot move freely. Examples of components of stationary AMHS are roller conveyors and transverse shuttle cars. Freely movable conveyors such as forklift trucks and automated guided vehicles (AGV) are not considered. In line with this objective, in the remainder of the paper, the term "control" is used for the control software for material handling equipment and the term AMHS refers to stationary AMHS.

The project aComA – Automated Code Generation for Modular Systems – is supported by the Bavarian Research Foundation, project no. 1098/13

II. Boundary Conditions for AMHS and Requirements for the Control Software Design

This section starts with the presentation of the working and boundary conditions of AMHS, which are used to derive five requirements for the control software design engineering.

Two common operational purposes of AMHS are manufacturing and distribution, which have different boundary conditions [1]. Manufacturing systems are strongly affected by shorter and shorter product lifecycles, which leads to frequent changes with regard to the products and therefore the production processes. Because of these changes, the processes and system configurations are not predictable in advance and the conveying systems need to be flexible with respect to the layout, the processes, and the goods that are conveyed [2]. In contrast to manufacturing systems, AMHS in distribution are much less affected by changes. The transport units (TU) used in distribution centers are standardized palettes or containers, and once the processes and layout are set, they rarely change. Consequently, the states in a distribution center are predictable and can be determined in advance. The focus of this work is the application of AMHS for distribution processes.

AMHS within distribution processes are characterized by two basic properties of industrial plants. On the one hand, the

systems are custom-built and on the other hand, these custom-built systems are composed of standardized material handling equipment, comprising conveyors and handling systems. The custom-built systems are necessary because customized processes have to be realized, existing building structures have to be taken into account, and preceding and subsequent processes have to be integrated [3]. The conveyor and handling system market is characterized by many manufacturers, each of which has an extensive equipment catalogue [4]. Companies in the automated material handling equipment sector can be divided into three categories.

- Material handling equipment manufacturers: These companies produce and sell material handling equipment, but they do not build complex AMHS with superior controls.
- System integrators: These build complex AMHS with superior controls, however, they do not produce the equipment themselves, but rather purchase it instead.
- General contractors: These companies produce material handling equipment and build complex systems with superior controls with this equipment.

Both general contractors and system integrators need to build and commission systems on a tight schedule and in a cost effective manner. Therefore, it is essential to effectively design high quality control software. For system integrators, it is also essential to use their competitive advantage of being independent of a specific equipment manufacturer, because that enables them to choose the most suitable equipment for the each application. The drawback of using equipment from different manufacturers is that they need to be able to handle a wide range of sensors, actuators, and process principles of the material handling equipment, which makes the control software design a complex and time consuming task.

The mechanics and electronics of material handling equipment is long-lasting, with a lifespan of 15-20 years [5]. However, since the market environment of distribution centers is more fast-moving, it is necessary to modify systems, and therefore the control software, during the lifespan of the material handling equipment. Still, material handling equipment is being further developed due to technical progress and innovations, which need to be considered in terms of the engineering process of new systems. Additionally, AMHS are used in interconnections with external systems such as material flow computers or plant visualization, to mention just two of a range of many external systems. For each of these external systems, there are a lot of different vendors, and the communication interfaces are not standardized.

The control of a AMHS comprises two tasks: the control of the material handling equipment, meaning controlling the actuators and reading the sensors, and the material flow control, which is necessary within spatial subsections of the AMHS. Such subsections are specific implementations of logistics functions and are called *logistics elements*, see Fig. 1. The logistics functions include the following six functions: convey, merge/divert, handle, store, sort, and order picking [1]. Within the merge/divert logistics function, one *logistics element* is, for example, a T-intersection or H-intersection. Within the *logistics elements*, decisions have to be made, such

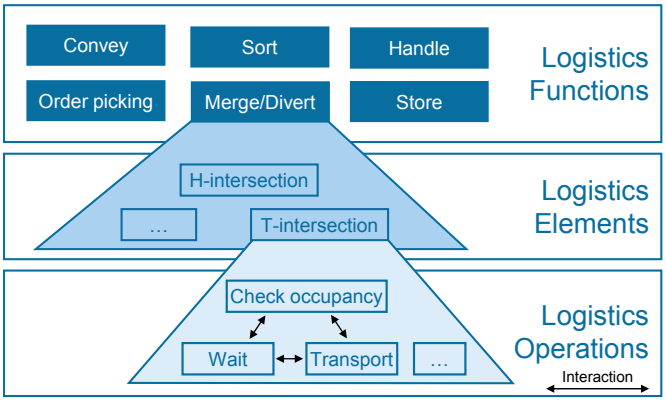

Fig. 1. Overview of the terms logistics functions, *logistics elements* and *logistics operations*.

as the right of way, which is performed by the process logic. This process logic consists of *logistics operations*, which are divisions of the logistics functions into individual operations. One example of a division is that the T-intersection consists of the following operations: transporting, waiting, and checking if a spot is occupied. The characteristic of the AMHS control is that a one-to-one correlation between the two control tasks is missing. One material handling component can be accessed by several material flow controllers, see Fig. 2, which can lead to conflicts.

Based on these working and boundary conditions, five requirements for the engineering of the control software are derived.

1. Low effort for designing individual high quality control software
2. Easy modification of existing control software
3. Highly standardized interfaces to external systems
4. Low effort needed for creating module libraries and implementing the engineering tool
5. Low effort for maintaining the module libraries

In the following section, the state of the art regarding the design of control software for AMHS and possibilities to improve software design are analyzed in order to develop concept elements that yield a suitable engineering concept for the control software design.

III. State of the Art

This section is divided into three parts. First, central and decentralized control methods for AMHS are compared with the conclusion that a central control is more suitable for the application considered in this paper. Second, the method of model-driven engineering (MDE), which can be used to enhance software design, is presented, and finally approaches in the field of AMHS for applying MDE are described.

A. Comparison of Central and Decentralized Control

The common method in industry for controlling AMHS is the use of central control. Central control has the advantage that all information is directly allocated for decision making, which enables optimal decisions, the execution of the control program is fast, since the programmable logic controller (PLC) has a powerful processor, and sensor reading and

commanding the actuators are optimized with regard to time. Moreover, all program states of a PLC are known and predictable, since all states need to be defined. The limitations of central controls are poor flexibility of the control concerning changes, the control cannot handle unpredicted states, and for large systems with a high range of possible states, optimal decision making is not possible within a reasonable timeframe. In order to overcome these limitations, decentralized control systems are developed, which aim at improving the aforementioned limitations.

Decentralized control approaches aim at developing autonomous components that interact flexibly within a system over a defined communication interface. Therefore, a component control for each component needs to be developed only once, the development of a system controller is omitted in such concepts, since the system configures itself autonomously, making the systems very flexible with regard to modifications. Examples of decentralized control approaches for AMHS are [2], [6]. Drawbacks of decentralized systems include long cycle times that result from the communication between the components as well as suboptimal decisions, resulting from solely local information, which leads to lower throughputs by the system [6], [7].

The flexibility advantages of decentralized control systems are most relevant for manufacturing conveying systems, since these systems frequently change. Distribution systems, which are more static, benefit less from the advantages of decentralized systems, but are significantly affected by the disadvantages, since they are optimized for high throughputs. Therefore, we focus on central control systems in this paper.

B. Enhancing Software Design through MDE

Software engineering productivity can be improved through four factors: abstraction, reuse, automation, and process improvements, which are all included in the concept of MDE [8]. The idea behind MDE is to develop a specific abstract system description for the considered domain and to create predefined reusable software modules based on this abstract system description. The custom-built system is modeled following a defined process and using the predefined modules. This modeled system is then used to automatically generate the control code for the specific control hardware. There are areas of automated systems for which domain-specific system descriptions exist, for example in the building automation or process engineering sectors [9], [10]. Moreover, there are various projects that develop methods and system descriptions to use MDE for manufacturing systems, but since they focus on production systems, they do not consider the specific requirements for AMHS [11], [12], [13], [14], [15]. To develop control software based on models, a supporting engineering tool is necessary. One example of an approved supporting engineering tool is CODESYS, which can be used for software design in various automation sectors [16]. So far, however, the domain-specific requirements of AMHS have not been covered and implemented.

C. Models for AMHS

The established standard for central controlled systems is an automation pyramid in which different tasks are assigned to different levels of the pyramid. With regard to AMHS, common levels are an enterprise level, a material flow level, a control level, and a level for controlling the sensors and actuators. The standard *SAIL – System Architecture for Intralogistics* defines interfaces between these control levels in order to standardize the communication [17]. Since SAIL applies to all kinds of conveying systems, stationary conveyors as well as freely movable conveyors, the interfaces are defined in an abstract way. Still, these two standards provide a valuable starting point for creating specific interfaces for automated stationary AMHS.

Besides the standards, there are research projects focused on the modularization of AMHS. Wilke proposes a modularization of automated AMHS called functional modularization. This means that each module should encapsulate a function and everything needed to perform that function, the hardware as well as the software, has to be within the module [18]. Wilke's work includes a general proposal for a modularization for all types of AMHS, but does not provide for a specific modularization for designing the control software for stationary AMHS. Ten Hompel presents a two-layer architecture for controlling decentralized AMHS, in which he separates a conveyor layer, responsible for controlling the sensors and actuators, and a logic layer, responsible for decision making [19]. He proposes this separation in order to be able to control the material flow with uniform logic modules and execute the tasks with a wide range of material handling equipment. In this work, only one-to-one correlations between logic and conveyor modules are considered, and therefore the conflict situations within one conveyor module are not considered. Moreover, Black and Lepuschitz consider the control software development of conveyer systems, but both assume a one-to-one correlation between software modules and conveyors [20], [21].

To summarize, in various sectors that use automated systems, MDE is already successfully utilized to enhance the engineering process of control software. The basis for MDE is a sector-specific system description, highly reusable modules, and a structured design process that minimizes manual programming. As shown in this section, a system description with reusable modules is missing for AMHS and is therefore developed within this paper. In the following section, different concepts for such a system description are discussed and evaluated.

IV. Concept Selection and Description

This section starts with the analysis and evaluation of potential concepts. This is followed by a detailed explanation of the concept that best fulfills the requirements.

A. Analysis and Evaluation of Potential Concepts

Each control of a AMHS has to have a part which is responsible for the direct control of the material handling equipment and a part that is responsible for the material flow decisions (see section II). However, there are three possible concepts with regard to how these two parts can be combined. The first possibility is to combine the two parts by *rigid linking*. This means that a particular module exists for each

specific *logistics element* in a specific realization with material handling equipment. The concept of *flexible linking* contains two independent module types for the two parts. These two independent module types are linked via a standardized interface. One module type contains the material flow control, which makes a decision and creates a command (for example, "wait"). This command is forwarded to the other module type controlling the material handling equipment. This separation into two module types also exists in the *manual linking concept*, however in this concept, there is no standardized interface between the two module types. The lack of this standardized interface increases the flexibility, since compromises have to be made for each standardization, but it also increases the effort needed to design the control software and is an additional source of errors.

The *rigid linking* concept does not sufficiently fulfill requirements no. 4 and no. 5, because material handling equipment from different manufacturers is considered, which leads to a disproportional growth of the library and thus to a sharp increase in the effort needed to create and maintain the library. The *manual linking* concept performs poorly with respect to requirements no. 1 and no. 2, because the *manual linking* requires manual programming, which leads to a higher potential for error. Moreover, manual programming results in non-standardized control codes and therefore maintenance and modifications are more difficult. The *flexible linking* concept fulfills all requirements properly, thus making it the most suitable concept. It is presented in detail in the following section.

B. Detailed Concept Presentation

The *flexible linking* concept uses a two-layer architecture for the engineering of the control software, see Fig. 2. The upper layer contains modules that are responsible for the material flow decision, called *logistics modules*. The bottom layer contains modules that are responsible for controlling actuators and sensors of the material handling equipment, called *conveyor modules*. The two layers are connected through a standardized communication interface, called *material handling interface* (MHI). This separation in terms of a material handling layout is depicted in Fig. 2.

Logistics modules contain one *logistics element*, the settings for this *logistics element*, optimization strategies, and an interface to external systems as well as *conveyor modules,* see Fig. 3. The *logistics element* specifies the process logic and both the required and optional MHI. The process logic controls the individual process steps of the *logistics element* that are necessary for the correct execution of the logistics function. This comprises adherence with the chronological sequence of the process steps and observing if the conditions for the individual process steps are true. Besides the process logic, the *logistics element* defines the MHI that is required for the basic functionality or optionally needed for specific settings of the *logistics element*. For example, at a merger, a requirement is that each inlet has to have a sensor to check if the inlet is occupied. If the absolute right of way setting should be used for a merger, an additional MHI is needed that passes the information if the gap in the priority line is large enough.

The behavior of the process logic, for example the right of way at a merger, is adjustable by parameters, enabling a fast design of the control software without manual programming. For some *logistics modules*, there are optimization measures regarding the material flow. Considering a transfer shuttle car without a subsequent scheduled job, an optimization measure is to move the shuttle car to the position that has the highest probability for the subsequent job. To gain information from a superior system, for example to which sink a branch should convey, the *logistics modules* have an interface. Via this interface, it is also possible to connect other external systems, like a plant visualization.

Logistics modules are only necessary at places in the layout where decisions concerning two or more material handling components have to be made, for example at branches or at a handover between two conveyors. At places where no component-overlapping decisions have to be made the material flow results from the boundary conditions and no *logistics modules* are necessary, see Fig. 2 detail A. Furthermore, it is possible that *logistics modules* can overlap, see Fig. 2 detail B, which can lead to conflicts. A conflict arises, for example, if one *logistics module* sends the command "transport" and another *logistics module*, due to an upcoming collision, sends the command to stop. Such conflicts are resolved within *conveyor modules*.

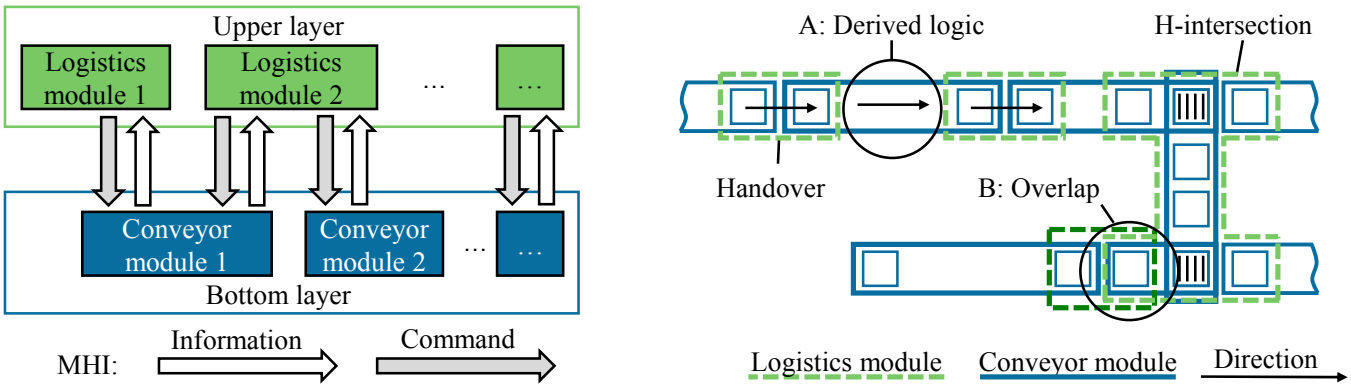

Fig. 2. Visualization of the two-layer architecture in a layer representation (left) and a layout representation (right).

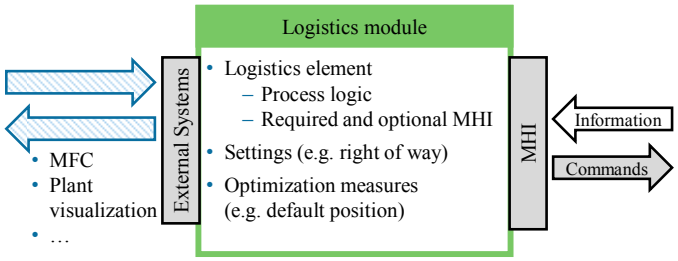

Fig. 3. Overview of the content and interfaces of *logistics modules*.

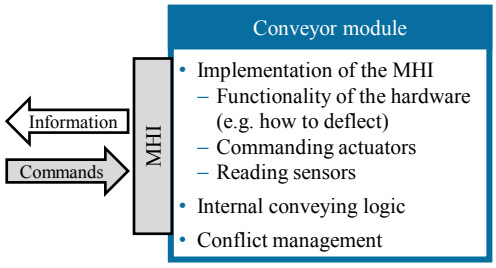

Fig. 4. Overview of the content and interfaces of *conveyor modules*.

Conveyor modules contain the logic for controlling material handling equipment, comparable with device drivers for computers. Via the MFI, these modules provide an interface to the *logistics modules* and they contain three elements, the implementation of the MHI, the internal conveying logic, and the conflict management, see Fig. 4. The MHI implementation comprises three sub-items, the functionality of the hardware, commanding actuators, and reading as well as interpreting sensors. The functionality of the hardware is needed, since several functioning principles exist for some functionalities, such as for deflections. Deflectors can consist of a one dimensional pusher or of chains, which first have to be lifted up and then turned on, to mention just two examples of functioning principles. The specific actuator implementation is used to be able to execute the commands from the *logistics modules* with various types of actuators. Moreover, the *conveyor modules* read the sensor signals, translate those signals into a command that is understandable for *logistics modules*, and broadcast the translated command to the corresponding *logistics module*. In addition to the implementation of the MHI, the internal conveying logic of the hardware components is implemented inside the *conveyor modules*. This internal conveying logic is used for conveying decisions that require only sensors and actuators of exactly one *conveyor module* and can therefore be performed autonomously by the *conveyor module*. Such conveying decisions include, for example, conveying from one end to the other end of a conveyor, see Fig. 2 detail A, or piling up TU on a zero pressure conveyor. The conflict management inside the *conveyor modules* is used to solve conflicts that can occur if one *conveyor module* is commanded by several *logistics modules*. For example, if one *logistics module* sends the command "stop" and another one sends the command "transport", the *conveyor module* has to stop, since ignoring a stop command can result in a collision. Since *conveyor modules* represent material handling equipment, they reproduce the layout without gaps.

To ensure unique communication between *logistics* and *conveyor modules* and to be able to flexibly combine the two module types, a standardized interface is necessary, the MHI. The MHI comprises all *logistics operations* that are transmitted between *logistics* and *conveyor modules*. Within the software implementation of the concept, this means that *logistics modules* refer to *conveyor modules*.

To connect external systems such as a material flow computer or a plant visualization to this two-layer architecture, interpreter modules (IM) are used. These IM know which commands the *logistics modules* understand. External systems send their commands to an IM, the IM translates the command so that the *logistics module* understands the command, and then the IM transmits the translated command to the *logistics module*. The reason for using this concept is that an IM for a specific external system has to be created only once and can then be reused. These connections to external systems are also used to obtain information about optimizations that comprise multiple *logistics elements*. Such optimizations are necessary, for instance, if several storage and retrieval machines (SRM) retrieve TU to a conveyor loop and each SRM should have a similar occupancy rate. Another example of an optimization is a branch that is supposed to divide incoming TU so that it results in a uniform occupancy of the succeeding accumulation conveyors. In both examples, the information that enables such an optimization could be provided by a material flow computer.

V. Application Example

In this section, the concept is explained using the example of a T-intersection, see Fig. 5. This example illustrates the functionality of the reference concept via the MFI and the task sharing between the modules. The conflict management, which is necessary if multiple *logistics modules* access one *conveyor module*, is not considered within the example.

The depicted T-intersection consists of three conveyors, and therefore three *conveyor modules*, and is controlled by one *logistics module*. Both the *logistics* and *conveyor modules* have positions for which the occupancy can be detected and for which the conveying direction is defined. These positions are called conveyor places (CP). The CP are labeled with letters and numbers for the *logistics* and *conveyor modules* respectively. For the interconnection between the *logistics* and *conveyor modules*, the individual CP of the *logistics* and *conveyor modules* are referenced to each other. In the depicted example, CP 2 of conveyor 1 (conv_1.CP_2) refers to CP B of the T-intersection (CP_B). This reference means that if a TU is detected on conv_1.CP_2, meaning conv_1.CP_2 = TRUE, the query of variable CP_B also returns the value TRUE. Additionally, this reference contains the assignment of the conveying commands, meaning if the *logistics module* sends a command to transport from CP B in the direction of CP C, conveyor 1 conveys in direction of conveyor 2. The other references in this example are conv_2.CP_1 = CP_C, conv_2.CP_2 = CP_A, and conv_3.CP_1 = CP_D. These references enable a process logic to be set within the *logistics module*, for example first come, first served, which is controlled within the *logistics module* and the commands for the execution are sent to the *conveyor modules*.

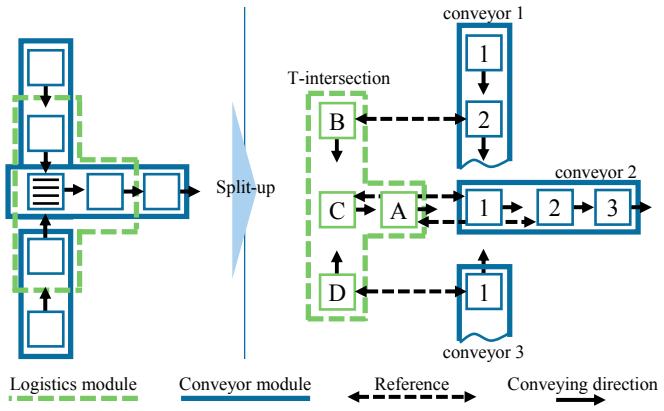

Fig. 5. *Logistics* and *conveyor modules* of a T-intersection.

This example was discussed with control engineers from a leading general contractor and system integrator. The experts approved the functional capability of both the reference concept and the task sharing. They also agreed that other material handling components, such as transfer shuttle cars or handling systems, can be integrated into the control software of AMHS in this way. The experts emphasized that a key factor for designing the control software is a fast design process, which can be additionally enhanced by providing common combinations of *logistics* and *conveyor modules* in a separate library and by keeping the graphical configuration at a minimum by focusing on spreadsheet configuration.

VI. Conclusion and Outlook

This article dealt with the design of control software for AMHS. These systems are widely used in industry because they provide high throughputs combined with high process quality. To use AMHS in an optimal way, custom-built systems are essential, which in turn requires individually implemented control software. Since there is no supporting engineering tool available, creating the control software includes manual programming, which is error-prone and labor intensive. To enhance the design process of the control software of AMHS, a supporting engineering tool is being developed within the aComA project.

The basis for such a tool is a modular software architecture, which is described in this article. For this purpose, first a list of requirements for designing the control software was derived. Next, potential concepts were explained and assessed. The result of this assessment was that a *flexible linking* between the material handling equipment and the material flow decisions is most suitable. This concept was presented in detail and illustrated by an application example.

The next steps will be an analysis of the material handling equipment and the *logistics elements*. Based on these analyses, the different MHI can be defined, the *logistics* and *conveyor modules* can be elaborated, and the conflict management can be analyzed and worked out. With the software architecture presented in this paper and the worked out modules, it is then possible to develop a engineering tool for designing the control software for AMHS. This engineering tool will then be evaluated by designing the control software for conveyor systems with industrial material handling equipment.

References

[1] W. A. Günthner, “Materialfluss und Logistik,” TU München, 2015.

[2] S. H. Mayer, “Development of a completely decentralized control system for modular continuous conveyor systems,” Ph. D. dissertation, Dept. Mech. Eng., KIT, Karlsruhe, 2009.

[3] M. ten Hompel and T. Schmidt, Warehouse Management: Automation and Organisation of Warehouse and Order Picking systems. Berlin: Springer, 2007.

[4] Materialfluss, “Materialfluss: Markt 2016,” Materialfluss, Dec., 2015.

[5] W. Degenhard, “Trendbericht: Automatische Lagertechnik,” FM Das Logistik Magazin, no. 06, 2014.

[6] S. Feldhorst, S. Libert, and M. ten Hompel, “Integration of a Legacy Automation System into a SOA for Devices,” in Proc. of 14th IEEE Int. Conf. on Emerging Technologies and Factory Autom., 2009, pp. 1-8.

[7] N. Klein, “The impact of decentral dispatching strategies on the performance of intralogistics transport systems,” Ph. D. dissertation, Dept. Mech. Eng., Tech. Univ. Dresden, 2013.

[8] S. A. Bohner and S. Mohan, “Model-Based Engineering of Software: Three Productivity Perspectives,” in Proc. of the 35th IEEE Software Engineering Workshop, 2009, pp. 35–44.

[9] S. Runde, A. Heidemann, A. Fay, and P. Schmidt, “Engineering of Building Automation Systems - State-of-the-Art, Deficits, Approaches,” in Proc. 15th IEEE Int. Conf. Emerging Technol. Factory Autom. (ETFA), Bilbao, Spain, 2010.

[10] D. Hästbacka, T. Vepsäläinen, and S. Kuikka, “Model-driven development of industrial process control applications,” J. of Syst. and Software, vol. 84, no. 7, pp. 1100–1113, July 2011.

[11] M. Vallee, M. Merdan, W. Lepuschitz, and G. Koppensteiner, “Decentralized Reconfiguration of a Flexible Transportation System,” IEEE Trans. Ind. Informat., vol. 7, no. 3, pp. 505–516, Aug. 2011.

[12] M. Bonfè, C. Fantuzzi, and C. Secchi, “Design patterns for model-based automation software design and implementation,” Control Engineering Practice, vol. 21, no. 11, pp. 1608–1619, Nov. 2013.

[13] L. Bassi, C. Secchi, M. Bonfe, and C. Fantuzzi, “A SysML-Based Methodology for Manufacturing Machinery Modeling and Design,” IEEE/ASME Trans. Mechatronics, vol. 16, no. 6, pp. 1049–1062, 2011.

[14] C. Brecher, J. A. Nittinger, and A. Karlberger, “Model-based Control of a Handling System with SysML,” Procedia Computer Science, vol. 16, pp. 197–205, 2013.

[15] E. Estevez, M. Marcos, I. Sarachaga, and D. Orive, “A Methodology for Multidisciplinary Modeling of Industrial Control Systems using UML,” in Proc. 5th IEEE Int. Conf. Ind. Inf., Austria, 2007, pp. 171–176.

[16] 3S - Smart Software Solutions, CODESYS Application Composer. Available: www.codesys.com

[17] Systemarchitektur für die Intralogistik (SAIL), VDI/VDAM 5100, 2011.

[18] M. Wilke, “Wandelbare automatisierte Materialflusssysteme für dynamische Produktionsstrukturen,” Ph. D dissertation, Dept. Mech. Eng., TU München, 2006.

[19] M. ten Hompel, S. Libert, and U. Sondhof, “Distributed Control Nodes for Material Flow System Controls on the Example of Unit Load Conveyor and Sorter Facilities,” Logistics Journal, pp. 1–9, 2006.

[20] G. Black and V. Vyatkin, “Intelligent Component-Based Automation of Baggage Handling Systems With IEC 61499,” IEEE Trans. Automat. Sci. Eng., vol. 7, no. 2, pp. 337–351, April 2010.

[21] W. Lepuschitz, V. Jirkovsky, P. Kadera, and P. Vrba, “A Multi-Layer Approach for Failure Detection in a Manufacturing System Based on Automation Agents,” in Proc. 9th Int. Conf. on Inform. Technol.: New Generations (ITNG), 2012, pp. 1–6.