Design Pattern for Decomposition or Aggregation of Automation Systems into Hierarchy Levels

Benjamin Brandenbourger
CAX-Service GmbH
Siedlungsweg 14, 85258 Weichs, Germany
brandenbourger@cax-service.de

Friedrich Durand
afag Automation AG
Fiechtenstrasse 32, 4950 Huttwil, Switzerland
friedrich.durand@afag.com

***Abstract*—In variable automation systems the encapsulation of sub-systems is a key factor for fast reconfiguration of modern production plants. This contribution proposes a generic and recursive design pattern for decomposing or aggregating automation systems into standardized abstract units. Depending on the use case, the top-down or bottom-up methodology is favorable for classifying the identified entities in different hierarchy levels. The intuitive, cross-domain approach creates commonly understandable, abstracted digital twins of the entities in form of integrated mechatronic models. The encapsulation of the entities allows a better exchangeability on each hierarchical level and paves the way for faster commissioning in variable automation systems. The approach is evaluated by means of a production cell presented at the automatica 2018 trade fair.**

I. Introduction

With the shift from mass production towards customer-specic products the manufacturing industry currently experiences drastic changes [1]. Companies are forced to react with fast refitting machines due to the increasing, dynamic, and variant nature of customer requirements [2]. One response to this requirement are Cyber Physical Systems (CPS) with functional interfaces [3]. A CPS is a set of physical devices, objects, and equipments that interacts through a communication network.

However, the evolvability in agile manufacturing systems needs more than only distributed intelligence in form of decentralized control units. Similar to software engineering, a conglomerate of CPS without a mature architecture is hard to maintain and to exchange parts of it. Therefore, based on the functional engineering approach this work introduces a design pattern for decomposing or aggregating automation systems in standardized abstract units. The units may be understood as digital twins representing a virtual abstract copy of the physical system [4]. The design pattern is intuitive, comprehensible, and applicable on the different disciplines involved in the engineering of automation plants. The recursive approach creates abstract digital twins of automation systems in form of integrated mechatronic models (IMM) as presented in [5]. Furthermore, the resulting logical units are hierarchically classified in levels, easily exchanged, and allow more flexibility when designing new systems.

In this work a generic design pattern for decomposing or aggregating automation systems is presented. Depending on the use case the approach is used as a recursive top-down or bottom-up methodology. We believe the proposed design pattern is applicable to a broad range of applications and facilitates the deployment and integration of new services in evolvable manufacturing systems. The design pattern has been developed and evaluated on two demonstrators of the German research projects OPAK and DEVEKOS. OPAK and DEVEKOS have been supported by the German Federal Ministry of Economic Affairs and Technology (BMWi) as part of the research program *Autonomik Industrie 4.0* and *PAiCE*.

The remainder of this paper is structured as follows: Section II gives an overview of available work and background knowledge in the field of functional engineering and abstraction in automation systems. An application example in form of a demonstrator is presented in Section III. Section IV describes the theoretical approach for decomposing or aggregating automation systems into standardized abstract units. Different hierarchy levels, three types of abstract units, and a design pattern are introduced. The theoretical approach is evaluated in Section V by means of the application example. Finally, Section VI concludes the paper.

II. Background

There are many efforts developing system architectures in general.

ZVEI (Zentralverband Elektrotechnik- und Elektronikindustrie e.V.) defined in 2015 a reference architecture RAMI 4.0 (Referenzarchitekturmodell Industrie 4.0) [6]. This architecture proposes seven hierarchy levels based on IEC 62264 and IEC 61512 reaching from the product-level to the connected world. However, the granularity of the levels in a production plant is low, as work centers are recursively decomposed only in stations, control devices, and field devices.

Alam & Saddik present and describe in [7] an architecture reference model called C2PS for cloud-based CPS. Pointing on the situation of reconfiguring, such as an unexpected demand, their model helps in identifying various degrees of basic and hybrid computation-interaction modes.

The work of Keddis [8] investigates in a similar direction on how to adapt manufacturing systems based on product recipes. The scheduled operations help to reduce human intervention and thus increases the autonomy. An algorithm for mapping production steps on a given plant topology is presented in her work and is taken up here in later sections. For this purpose, abstract transport units are introduced to interlink the value-adding processes. Nevertheless, the approach does not focus

978-1-5386-7108-5/18/$31.00 ©2018 IEEE

in classifying all the appearing automation components in standardized units.

Rosen et at. state in [4] that a digital twin represents a virtual abstract copy of the physical system, also referred to as a digital shadow containing all or partial information and knowledge of the considered system. Preusse et al. [9] propose a graphic-based method which enables the user to create a formal specification of behavior description with the use of symbolic timing diagrams. The authors of both works follow a functional approach for controlling the plant's actuators. The same functional approach is also taken up by Sierla et al. [11] and Brandenbourger et al. [10] by creating a domain-overarching integrated mechatronic model using a metamodel implemented in AutomationML. In [5] and [12] the same authors expand the integrated mechatronic model by a behavior model of automation components using cross-domain interdependencies. The components' and system's functions are encapsulated in so-called skills which start a procedure when called [13] [14] [15]. Skills are comparable to standardized service-interfaces provided by a device [16]. Frei defines in [17] skills as a module's capabilities which are offered to other modules. She categorizes skills in basic skills offered by single modules and composite skills composed of several basic skills by coalitions of modules. Helbig et al. introduces in [3] a standardized classification for those skills. Additionally, Frei introduces different types of agentified modules such as feeder, order, product, and part agent, which are comparable to the entities presented in a later section. All these works give ideas about possible abstraction of automation components and systems which build the relevant foundation for our approach explained in later sections.

However, no design-pattern or structured approach for creating abstract digital twins of automation systems in form of integrated mechatronic models and classifying them in an hierarchical architecture exists up to now.

III. Application Example

Figure 1 depicts a production cell used for assembling fidget spinners. For better credibility and evaluation, the application example is not a hypothetical construct but a real life and complex automation system created by system integrators.

The cell has been presented in 2018 at the trade fair automatica in Munich (Germany) [18] and was conceived by a large consortium consisting of component manufacturers, system integrators, and software developers. The VDMA (Verband der Deutschen Maschinen- und Anlagenbauer) [19] initiated and coordinated the tentative to create a vendor-independent, interoperable, and skill-based demonstrator. The communication within the VDMA R&A Demonstrator relies on OPC UA. Following the functional approach presented in [10], skills are modeled as OPC UA-objects containing an internal state machine and standardized methods such as *start()*, *stop()* etc.

The VDMA R+A Demonstrator consists of different stations assembling a customer-specific product, namely a fidget spinner. During the order process the trade fair visitor personalizes

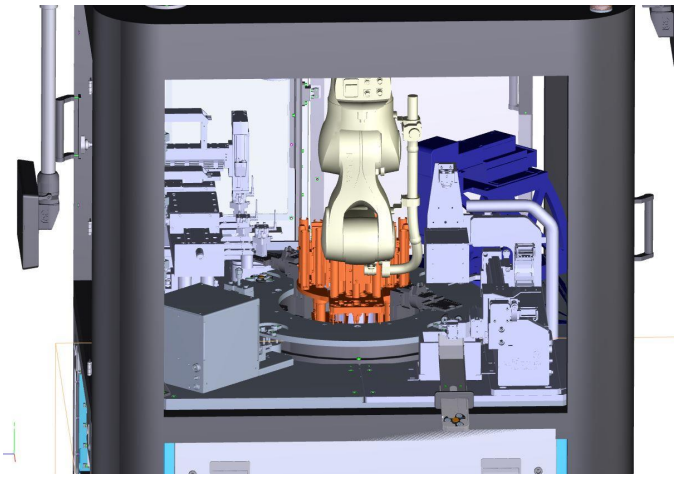

Fig. 1. VDMA R+A Demonstrator presented at automatica 2018 trade fair

the product by choosing the color of the body and the color of each caps located on both sides of the centric ball bearing. Furthermore, the inscription on one cap is free to be defined by the visitor.

After finalizing the order, the production starts in Station 1 by turning the central rotary indexing table into a defined position. Each magazine (highlighted in orange in Figure 1) on the central rotary indexing table contains stacked bodies of a specific color and ball bearings. In the next step an articulated arm robot places the isolated body and four ball bearings in the nest of the external rotary indexing table. A substation of Station 1 consists of three linear cylinders used for forcing the ball bearings into the body. After rotating the external rotary indexing table to the next position, Station 2 places one cap on each side of the centric ball bearing. The caps are provided on a bi-translational feeding unit allowing transportation, distribution, and flipping of the goods by means of a vibration system. A camera system detects the position, orientation, and colors of the caps and transmits the information to a pick&place unit, which grabs and presses the caps on the centric ball bearing. An additional cylinder with a rotary gripper unit is used for tilting the workpiece in order to access both sides of the centric ball bearing. Another pulse of the external rotary indexing table conveys the workpiece to Station 3 where a YAG-laser engraves the personalized text on one cap and the VDMA-logo on the other cap. Tilting the workpiece is performed just as in Station 2. The last pulse of the external rotary indexing table conveys the workpiece to Station 4. This station performs a quality check of the workpiece by controlling the colors and the engraving. The lettering detected by OCR, the color of the body, and the color of the two caps are compared with the information entered during the order process. Depending on the results of the quality check the workpiece is handed over to the visitor or discharged.

The complexity of the VDMA R+A Demonstrator has been abstracted for better understanding.

IV. Approach

In this work a design pattern for decomposing automation systems or aggregating automations components is presented. The basis for this recursive approach are standardized abstract units which are created during decomposition or aggregated to more complex units during composition. The generic abstract units, also referred to as entities, can be understood as digital twins of parts of the considered automation systems. They communicate on an event-triggered base by calling the entities' skills.

A. Engineering approach

The functional engineering approach presented in [10] and [5] forms the base of this work. An envisioned production process can be split up in production steps which are performed by standardized functionalities, also referred to as skills. These skills are offered by the automation components or subsystems incorporated in the considered automation system.

Figure 2 depicts the decomposition of the production process in standardized productions steps and their mapping onto standardized skills offered by automation components within the application example. The top part shows the envisioned abstract main production process and the corresponding production steps. Only value-adding process steps are listed here, as the resulting sequence of production steps is application-oriented and output-driven.

Depending on the plant topology, the abstract production process is extended with steps not creating additional value. These include plant-specific transport routes and information concerning material dependencies (see center part of Figure 2). The extension with non-value-adding process steps can either be performed manually by the application engineer or can be recognized automatically based on the plant topology [8].

Now that the plant-specific process steps have been determined, a decomposition into the corresponding standardized skills needs to be performed. For this purpose the standards VDI 2860 and DIN 8580 offer a wide range of generic skills. Each process step is split up in multiple generic skills.

The last step of the engineering approach consists in mapping a required generic skill with the corresponding specific skill offered by an automation component (see bottom part of Figure 2).

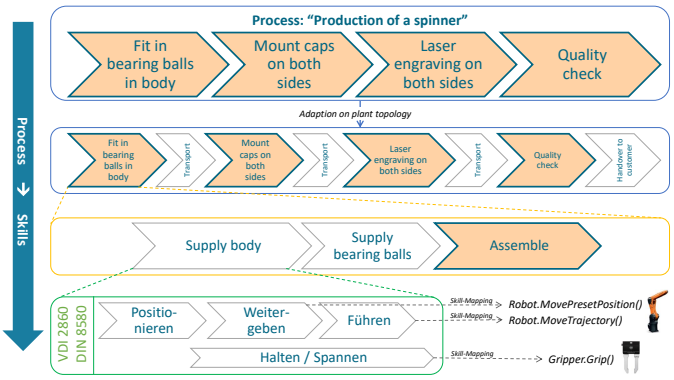

Fig. 2. Decomposition of a production process into steps and mapping on standardized skills

B. Hierarchy levels

Skill-mapping must not only be done on the lowest level as shown in the subsection before. Skills can also be aggregated to composed skills and offered on a higher level. Therefore a logical hierarchical classification needs to be introduced. This subsection proposes a generic systems architecture.

The presented levels must be seen as a best practice and are the result of several evaluations on different demonstrators e.g. of research projects. Applying the design pattern presented in this work does not necessarily entails the implementation of all hierarchy levels. There exist applications which could need less or additional hierarchy levels. The main feature of the classification in hierarchy levels is that an automation system can be completely decomposed in devices of a lower hierarchy level as well as single automation components can be aggregated to more complex automation systems on a higher hierarchy level. One-to-one relations between physical and virtual devices are maintained through different architecture levels, whereby a virtual device reflects the compositional nature of complex physical systems. The accuracy of the resulting model depends on the envisioned granularity. As the design pattern is recursive, the number of hierarchy levels is flexible and therefore the architecture may have different complexity granularities. This has the effect that the manifestation of an architecture is very simple, very complex, or even shows multiple granularities. Furthermore, the logical hierarchy levels remain unaffected by the shift from an hierarchical automation pyramid to a distributed automation net. A source-sink relation and therefore a logical hierarchy will still persist.

Based on the results of several evaluations, the following hierarchy levels of automation systems are suggested:

- **Line**: A production line is a production system containing several value-adding manufacturing units such as cells which are connected by material handling systems.
- **Cell**: A cell is a processing machine performing partial or full processing on the workpiece and changing tools as required. It is spacially separated from other cells. A cell can be decomposed in different stations.
- **Station**: A station is a self-contained, mostly value-adding unit designed for a specific process step. Stations meet customer-specific requirements and are spatially separated from each other. Furthermore, stations are interconnected by a transportation system within the cell.
- **Substation**: A substation is a partial structure of a station. They are used for logically and physically decomposing complex stations in manageable units. The application-independent reusability of substations is high.
- **Module**: A module is a logical grouping of components and can exist within a substation or station. This hierarchy level is highly application-specific and therefore optional.
- **Component**: A component is an automation unit performing a specific simple automation task such as a trans-

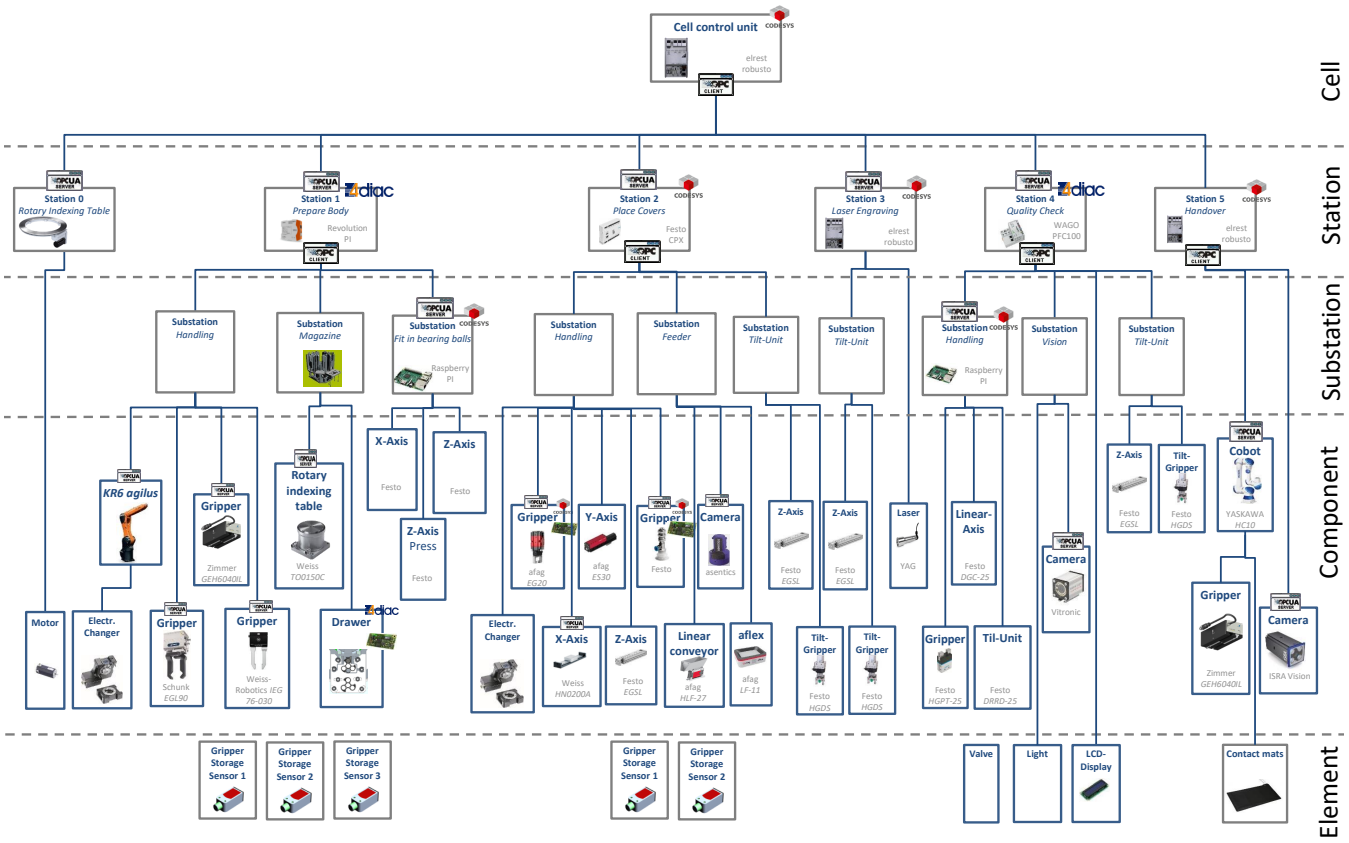

Fig. 3. Hierarchical classification of units in VDMA R+A Demonstrator

latory movement. Components are combined to create complex systems on the level of modules, substations, and stations. Furthermore, components may contain elements. Examples for components are axis or grippers, while there is no difference whether the components are legacy or novel integrated devices including an own processing unit.

- **Element**: An element is a basic automation unit which needs to be connected to a control unit. Examples for elements are sensors and valves. Passive mechanical structures are also assigned to the level of elements. Elements can not be split up in smaller units.

Apart from the last level, each hierarchy level possesses a process sequence as exemplary depicted in Figure 2. The process sequence from the highest level to the modules level are modeled as composed skills whereas components offer basic skills. Furthermore, this approach supports a finite state machine as presented in [5] defining the behavior of the modeled resources.

C. Entities

Entities are generic, standardized, and abstract units of automation systems. They can be understood as a holistic logical segmentation of a complex system. As the presented approach is recursive, the decomposition of a system into entities or the aggregation to a system of systems can be performed on any hierarchy level. A compromise between complexity originating from the decomposition or aggregation and flexibility originating from the encapsulation is expedient. The presented approach describes specific properties of the entities such that every mechatronic unit is assignable to one type of entity.

This work introduces three types of entities:

Transporter - TR (incl. Decider): A decomposed system contains exactly one TR which can optionally be split up in an input- and output-TR. The TR-entity transports the main workpiece to be produced. The same TR can be used in different parallel decomposed systems as a TR-entity supporting concurrent access on one resource. Furthermore, this entity may be casted to a SP-entity for the next station in the processing order. For influencing transportation rules or choosing the right recipe for the workpiece, a TR-entity may include a Decider-entity (DC). A DC is a sensor with internal or external processing influencing the behavior of TR. The DC-entity can also exist only in a virtual manner (e.g. as a logical function). Examples for DC-entities are stopper-modules, turnouts, fixing-unit, etc. Examples for TR-entities are conveyors, rotary tables, clocked conveyors, passive mechanical slides, etc.

Worker - WK: A decomposed system contains zero to many Workers (WK).The WK-entity is the actual working unit of the decomposed system and incorporates in most cases a

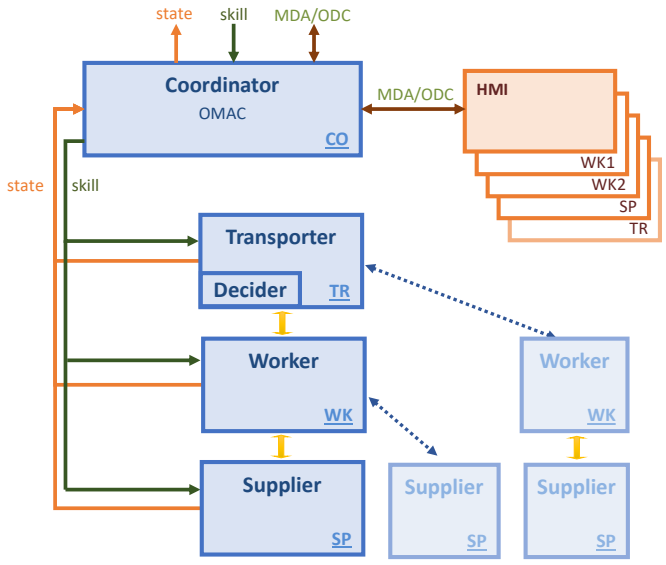

Fig. 4. Logical decomposition of a system into standardized entities

value-adding process. Examples for WK-entities are press-in units, check units, engraving units, drilling units etc.

Supplier - SP: A decomposed system contains zero to many Suppliers (SP). A SP-entity feeds the considered system with material (a.k.a."material influx") whereby it should not to be confused with a TR-entity transporting the workpiece. A SP can be recursively decomposed in further sub-entities. Examples for SP-entities are stocks, conveyors, humans, other substations, etc. Conveyors and other transport units may be casted as TR or SP, depending on the main material flow.

Figure 4 depicts the correlation between the entities. A logical coordinator regulates the interaction between the entities. The logical coordinator can be located on a separate physical resource or on one of the entities. The entities offer skills which can be orchestrated to composed skills on the level of the coordinator.

Classifying a decomposed system in entities enables better exchangeability of the entities without changing the general sequence of the orchestrator. Only the internal sequence of the exchanged entity needs to be implemented. This exchangeability offers the advantage of easily replacing faulty entities by spare parts or even by a human performing the requested task such as transporting the workpiece or supplying a machine with raw material.

D. Design Pattern

Based on the work introduced in Subsection IV-B and IV-C a design pattern is presented in this Subsection. The main goal of the design pattern is structuring an automation plant into hierarchy levels and assigning types of entities to sub-units. The design pattern is recursive, generic, flexible, and may be applied with a top-down or a bottom-up approach.

1) Top-Down approach for existing automation systems: The top-down approach is suitable for decomposing already existing automation systems. Beginning from the top level (e.g. *Cell*) the considered system is decomposed in units of the next hierarchy level (e.g. *Station*). These logical units are seen as black boxes without further in-depth knowledge about their structure or skills. The decomposition is holistic such that the complete considered automation system is modeled in the sum of the decomposed units. If a decomposition in the next hierarchy level is not possible the after next hierarchy level is used. A decomposition in the next hierarchy level is not be possible if the classification of the decomposed unit is clearly a lower hierarchy level (e.g. a *Station* containing *Elements* such as sensors). The decomposition is repeated recursively for each unit until no further decomposition is necessary as the envisioned granularity of the model is reached. The result is a hierarchical classification of all mechanical units of the considered automation system.

Decomposing the considered system with a top-down approach without in-depth knowledge about the decomposed units could lead to a good mechanical but bad logical decomposition. As the single units in the hierarchy levels should represent logical clusters of lower units, the logical structure of the considered automation system must be taken into consideration. Hereto the material flow must be respected. It is described in Section IV-B that the value-adding process steps are modeled in *Stations*. Therefore, one way to easily decompose a *Cell* into *Stations* is by creating the same amount of stations as the number of process steps (see top part of Figure 2). Applying recursively this decomposition leads to *Stations* which are decomposed in *Substations* (or lower hierarchy levels) and so on.

After having decomposed and classified all units of the considered automation system the second step of the design pattern consists in assigning the types of entities (see Section IV-C) to the *Substations*. Each *Station* features at least one *Substation* as a Transporter (TR) and usually one *Substation* as a Worker (WK). The *Substation* Supplier (SP) is optional and depends on the *Station*.

2) Bottom-Up approach for new automation systems: In contrast to the top-down approach, when applying the bottom-up approach the envisioned automation system is not necessarily known. This approach recursively aggregates single units to more complex units resulting in a system of systems architecture. *Elements* and *Components* are grouped to *Modules* or *Substations* which are in turn aggregated to *Stations* depending on the required production steps if already known. Similar to the top-down approach, the second step of the design pattern consists in assigning the types of entities to the *Substations*.

The bottom-up approach is especially useful for plant engineers wanting to create encapsulated production units which are preconfigured and ready to use. At a later date, during the engineering phase of an automation system, those preconfigured entities and their interaction can easily be planed in a plant's structure.

V. Evaluation

The approach presented in section IV is evaluated in this section by means of the VDMA R+A Demonstrator presented at the automatica 2018 trade fair. The demonstrator is a real-world example of an automation system as it was conceptualized by machine builders for realistic production. Figure 2 depicts the main production process of the demonstrator and how this process is split up in single process steps which are in turn mapped on the skills offered by the installed automation components. The evaluation on the VDMA R+A Demonstrator has shown that the proposed methodology is able to provide a standardized approach while maintaining exchangeability and component reusability.

Applying the top-down approach of the design pattern only on the highest hierarchy level of the VDMA R+A Demonstrator, namely the cell, results in a decomposition with two entities. The first entity is a *Transporter* split up in input- and output-TR and performed by a human being reloading the magazines and receiving the final product. The second entity is a *Worker* which is the complete cell seen as a black box.

Recursively applying the same approach to the black box, namely to the interior of the complete cell, results in three types of entities: an entity *Transport* implemented by the external rotatory indexing table, four entities *Worker* implemented by all the working units aligned around the external rotatory indexing table, and one entity *Supplier* implemented by the internal rotatory indexing table with the magazines containing the bodies and ball bearings.

The external rotatory indexing table is a special *Transport* entity and can be understood as several clocked conveyor belts. Each segment of the external rotatory indexing table used by a station corresponds to one clocked conveyor belt.

The granularity of the decomposition is still not sufficient such that the next cycle consists in decomposing the cell in several stations as depicted in Figure 5 and Figure 3. Each station contains again entities of different types. Applying for example the design pattern on Station 1 results in four entities: the first entity *Transport* is realized by the segment of the external rotatory indexing table located beneath the robot. The second entity *Worker* is implemented by the axis pressing the ball bearings into the body and therefore performing the value-adding process step of this station. The third entity *Supplier* consists on the one side of the internal rotatory indexing table with the magazines containing the bodies and ball bearings. On the other side the articulated robot moving the bodies and ball bearings is also part of the entity *Supplier*. A further decomposition of these entities is not meaningful.

Each entity offers skills implemented in the VDMA R+A Demonstrator as OPC UA objects containing OPC UA methods. These skills are implemented on the control units of the entities resources. Applying the design pattern for decomposition on another station may result in some resources with no integrated control unit. In this case, the superordinate, coordinating resource creates a wrapper for the necessary skill [16].

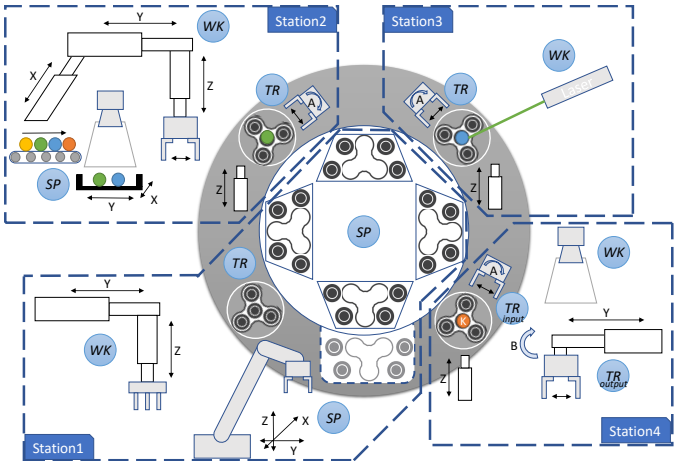

Fig. 5. Schematic perspective of VDMA R+A Demonstrator and decomposition in stations and entity types

VI. Conclusion and Future Work

In this paper, we propose a generic design pattern for decomposing or aggregating automation systems in an hierarchical architecture. Depending on the use case the approach can be used as a recursive top-down or bottom-up methodology allowing component reusability and modularity. Furthermore, the identified units are classified in one of three types of entities allowing a better exchangeability. Lessons learned show a compromise between complexity originating from the decomposition or aggregation, and flexibility originating from the encapsulation of the devices.

Future research will investigate how to make this approach feasible for process manufacturing systems. We also plan on extending the approach to define a standardized interaction between the entities. That means to have a predefined generally valid process e.g. for substations and thus increase the exchangeability of single units. This is necessary for reducing the engineering and programming effort in highly variable automation systems Another direction is to store the entities' type as meta-information in a units' model implemented in AutomationML. Finally, as the concept of systems architecture matures and evolves, we will investigate different optimization criteria for classifying devices in entity types.

References

[1] H. Kühnle, "Post mass production paradigm trajectories," *Journal of Manufacturing Technology Management*, vol. 18, pp. 1022–1037, 2007.

[2] G. Da Silveira, D. Borenstein, and F. S. Fogliatto, "Mass customization: Literature review and research directions," *International journal of production economics*, vol. 72, no. 1, pp. 1–13, 2001.

[3] T. Helbig, S. Henning, and J. Hoos, "Efficient engineering in special purpose machinery through automated control code synthesis based on a functional categorisation," *Machine learning for cyber physical systems*, 2015.

[4] R. Rosen, G. von Wichert, G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567–572, 2015.

[5] B. Brandenbourger, M. Vathoopan, and A. Zoitl, "Behavior Modeling of Automation Components using cross-domain Interdependencies," *Int. Conf. Emerging Technologies & Factory Automation (ETFA)*, 2016.

[6] ZVEI, "RAMI 4.0 (Referenzarchitekturmodell Industrie 4.0)," *Zentralverband Elektrotechnik- und Elektronikindustrie e.V.*, 2015.
[7] K. M. Alam and A. El Saddik, "C2PS - A digital Twin Architecture Reference Model for the cloud-based cyber-physical systems," *IEEE Access*, vol. 5, pp. 2050–2062, 2017.
[8] N. Keddis, "Capability-based Planning and Scheduling of Workows for Adaptable Manufacturing Systems," *Dissertation*, 2016.
[9] S. Preusse and H.-M. Hanisch, "Specification and verification of technical plant behavior with symbolic timing diagrams," *3rd international Design and Test Workshop*, 2008.
[10] B. Brandenbourger, M. Vathoopan, and A. Zoitl, "Engineering of Automation Systems using a Metamodel implemented in AutomationML," *Int. Conf. on Industrial Informatics (INDIN)*, 2016.
[11] S. Sierla, V. Kyrki, P. Aarnio, and V. Vyatkin, "Autonomic assembly planning based on digital product descriptions," *Computers in Industry*, vol. 97, pp. 34–46, 2018.
[12] B. Brandenbourger, M. Vathoopan, and A. Zoitl, "Modeling and verifying behavioral constraints for automation systems," *Int. Conf. on Industrial Informatics (INDIN)*, 2017.
[13] P. Ferreira and N. Lohse, "Configuration model for evolvable assembly systems," *4th Int. Conf. Conference On Assembly Technologies And Systems (CIRP)*, 2012.
[14] M. Onori, N. Lohse, J. Barata, and C. Hanisch, "The IDEAS project: Plug & produce at shop-floor level," *Assembly Automation*, vol. 32, no. 2, pp. 124–134, 2012.
[15] J. Pfrommer, M. Schleipen, and J. Beyerer, "PPRS: production skills and their relation to product, process, and resource," *Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–4, 2013.
[16] K. Dorofeev, C.-H. Cheng, P. Ferreira, M. Guedes, S. Profanter, and A. Zoitl, "Device Adapter Concept towards Enabling Plug&Produce Production Environments," *Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*, 2017.
[17] Frei Santos Barbosa, Regina Maria, "Self-organisation in evolvable assembly systems," *Dissertation*, 2010.
[18] Trade fair, "automatica," *https://automatica-munich.com/*, 2018.
[19] VDMA, "Verband der Deutschen Maschinen- und Anlagenbauer," *Science of computer programming*.