Orchestration of Services in Modular Process Plants

Henry Bloch, Tobias Grebner, Alexander Fay
Institute of Automation Technology
Helmut Schmidt University
Hamburg, Germany
{henry.bloch; grebner.tobias; alexander.fay}@hsu-hh.de

Mario Hoernicke
ABB Corporate Research
Ladenburg, Germany
Mario.hoernicke@de.abb.com

Stephan Hensel, Anna Menschner (née Hahn), Leon Urbas
Chair of Process Control Systems
Process Systems Engineering Group
Technische Universität Dresden
Dresden, Germany
{stephan.hensel; anna.menschner; leon.urbas}@tu-dresden.de

Torsten Knohl, Jens Bernshausen
Bayer AG
Leverkusen, Germany
{torsten.knohl; jens.bernshausen}@bayer.com

***Abstract*—Modularization of process plants is seen as one approach to face the increasing flexibility requirements in the process industry. Therefore, modules are combined to a modular process plant to achieve a higher flexibility of the production. Conventional distributed control systems do not support flexible production systems adequately. Hence, modules should encapsulate process functions as services. The services should be invoked and controlled by state-based control. This contribution presents an approach for the state-based control of services, following a defined state-model. The approach has been tested in two case studies and two different implementations: In a simulation environment based on Matlab Simulink Stateflow, and in an industrial implementation of a pump and a controller. This contribution provides an overview of the developed concepts and their capability for the use in industrial environments.**

Keywords— State models; process control; simulation; modular process plants

I. INTRODUCTION

A. Modularization in process plants

The global process industry faces a changing business environment. Trends such as shorter product lifecycles and high-volatile markets result in the need of a flexible production environment [1, 2]. One possible enabler to achieve a flexible process plant is the modularization of the plant. This also requires the modularization of the process control system [1]. A modular plant consists of different process modules which offer encapsulated process functions as services to the so-called superior control system (SCS). Within the SCS, services are orchestrated by the plant operator to achieve the desired production process. Thereby all services are controlled by means of a state model, which has been defined in the NAMUR (User Association of Automation Technology in Process Industries, www.namur.net) working group 2.3.1, in accordance with [3]. The state model is defined as a statechart of states and transition, whereby all states and transitions have a unique enumeration to identify active states and to fire transitions. To base the control of modules on a state model has been decided by the German Electrical and Electronic Manufacturer's Association (ZVEI, www.zvei.org) [4]. The ZVEI, the NAMUR and the VDI/VDE-GMA (The Association of German Engineers/Association of Measurement and Automation Technology, www.vdi.de/gma) work together in several working groups and define a standardized description of modules, the so-called Module Type Package (MTP), in the first joint VDI/VDE/NAMUR standard in history. The MTP shall contain all necessary information to integrate a module into a modular plant, such as the communication, the services, a human machine interface (HMI) description and maintenance information. Further aspects of the module will be defined by subsequent parts of the standard [5]. The authors of this paper contribute to the standardization process and are members of all related working groups of NAMUR, ZVEI and VDI/VDE-GMA.

Within this paper, an approach is presented to control process services of modules via a defined state model by means of an orchestration unit in the SCS. Additionally, an approach for the representation of the services and the orchestration unit in the central HMI is presented. To test the approaches, the authors have designed and implemented a simulation environment in Matlab Simulink Stateflow in combination with a graphical user interface (GUI). This simulation environment allows to test the approaches in defined case studies in parallel to the ongoing standardization work. Within this paper an example defined by working group 1.12. of NAMUR and ZVEI is taken as the first case study. Its implementation within Matlab Simulink Stateflow is presented. As second case study, an implementation of a Dosing service in an industrial environment is presented.

978-1-5090-6684-1/18/$31.00 ©2018 IEEE

B. *Structure of the paper*

This paper is structured as follows: In Section 2 the general setup of modular process plants, the concept of process services and the superior control is introduced. In Section 3, the state-based control of process services is presented. In this section, the focus is set on the state model and the invocation of services by the orchestration unit of the SCS. The simulation environment for the first case study is introduced in Section 4, which is followed by both case studies in Section 5. The contribution concludes in Section 6 and gives an outlook on future work of the authors.

II. MODULAR PROCESS AUTOMATION

A. *General setup of modular process plants*

The structure of a modular process plant is divided into different modules, which can each be provided by different vendors. However, each module provides its own control as it has its own controller, which contains the programs for the basic automation of the module itself. The basic automation equals the encapsulated process functions of the module, which are offered as services to the environment of the module. This concept has been defined by the NAMUR as an intelligent module [6]. To orchestrate all modules, they are connected to a so-called backbone, which provides all needed physical media, energy supplies, and information networks and contains the SCS. The SCS itself comprises an orchestration unit which fulfils the function of the conductor of the plant. The orchestration unit triggers the production process, collects all feedback of the services, handles the information and returns the commands for each service to the controllers of the modules. The services of a module then send commands to the field devices within this module and receive sensor signals from within this module in return. The general setup is shown in Fig. 1. As a modular plant is expected to run in an orchestrated operation, all information about the current state of the plant is displayed on an HMI in the SCS.

B. *Process services in modules*

The core elements of modular process automation are the process services. Services are encapsulated process functions which are offered by the module, e.g. mixing, filling, dosing. These services form a new layer in the communication architecture of modular process automation compared to the communication architecture of classical process plants and limit the depth of intrusion of the SCS into the module (see Fig. 1). Field devices are no longer addressed directly by the SCS, but only via services, to ensure the integrity of the module and also to protect the module suppliers' know-how.

The suitability of service-oriented architecture approaches and the Microservice approach for modular process automation have been analyzed already by the authors in [7, 8].

C. *Superior control system*

The presence of an SCS in the architecture of modular process automation is required by the plant owners [6]. The task of the SCS does not differ from the typical tasks of process control systems as listed below, except for the additionally

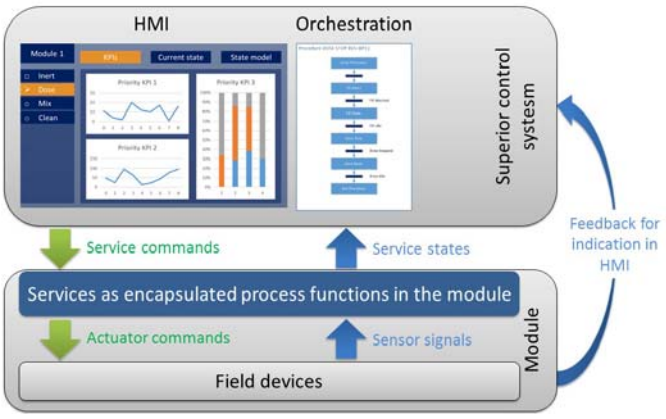

Fig.1 General setup of modular process plants

required flexibility: whenever the physical topology of the modular plant is changed by a new arrangement of the modules or the inclusion of new modules or the removal of modules, the SCS has to adapt to the new module configuration. The SCS has to fulfil the tasks of a process control system, such as [9]:

- Provision of an HMI to allow the operator to influence the technical process
- Control and adjustment of the technical process
- Provision of a recipe/batch control interface
- Supervision and alarm management
- Archiving of parameters and process data
- Data analysis
- Long-time recording of signals

In this contribution, the authors focus on the first three topics: the process control, the provision of a batch interface and the HMI. The batch interface is provided by an orchestration unit, which is structured in accordance with the recipe standard [10]. In [11] the authors presented already the model-based engineering of modules and recipes based on the MTP.

III. STATE-BASED CONTROL OF MODULAR PROCESS PLANTS

In this section, the state-based control of a modular process plant is presented. The state-based control is divided into two parts: (i) the control of services that follow a defined state-model and (ii) the orchestration of these services within the SCS. Each service of a modular process plant should follow the state model of [3]. This state model combines the well-known design of the state-machine of the first edition of the standard with the new hierarchical aspects and the enumeration of PackML [12]. The state model, which is used, is shown in Fig. 2.

Fig. 2 shows the state-model, which consists of 12 different states, which are structured in four hierarchical layers. The different layers are displayed by the different shades of the rectangles around groups of states in Fig. 2. The transitions are shown by arrows which connect two states or have a group of states as their source and a state as their target. The transitions

are divided into two sets, depending on the events which fire the transition. Such an event can be either a Service Command from the SCS (in this case, the transition is numbered and labelled according to the SCS command), or the transition firing is triggered by an internal event which stems from the dynamics of the actions of this service (in this case, the transition is simply labelled "SC" (state change)). Additionally, the event can also be caused by the firing of a transition or an active state of a different service of the module. The states are also divided into two different sets: acting states and waiting states. Each state which is followed by a transition of type SC is an acting state. All other states are waiting states. Within acting states programs with a defined end are executed. In waiting states, programs are executed in a loop-based manner to keep the plant within a defined situation, e.g. a mixing would continue within the state Held to prevent a curing of the fluid.

The hierarchical layers of the state model allow leaving any state of a group by the exiting transition, e.g. "4: Hold", "3: Stop" or "8: Abort". This allows implementing universal functionalities in the process services which are traditionally executed by the batch tool, such as to pause, stop or abort a batch phase or service.

In addition to the design of the services of a module, also the orchestration of the services is organized in a state-based manner. The main structure can be implemented in a classical sequence recipe which consists of states and transitions, similar to a Sequential Function Chart (SFC) according to [13]. In addition, other orchestration methods, such as other recipe types [10], are applicable. In this contribution, the authors focus on sequence recipes for the orchestration of process services. An example of a part of a sequence recipe is shown in Fig. 3. The example shows a single procedure that consists of several states and transitions. In the states of the recipe, commands are sent to the module's services, and at the transitions, the status' of the services are checked, see Fig. 4. Therefore, the output of the recipe's state-machine is the input for the service's state-model, and the output of the service's state-model is the input of the recipe's state-machine.

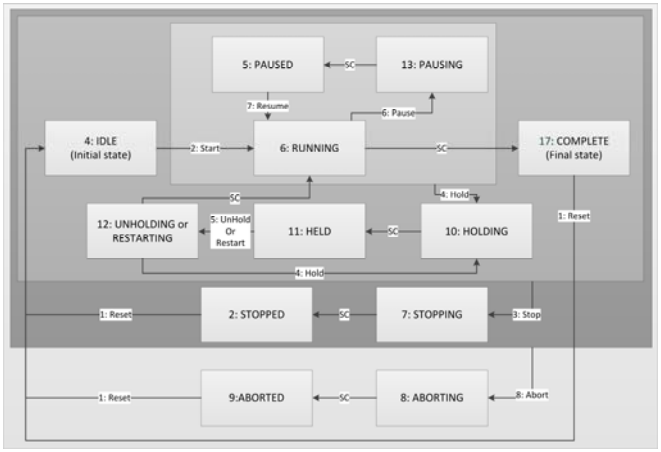

Fig. 2 State-model for process services following [3]

Fig. 3 shows only a single procedure of a recipe. A procedure describes a defined part of the recipe and contains the information about the needed resources for execution; in modular process automation, these resources are modules. As different procedures of a single recipe can be simultaneously active during the production process, also multiple states of the recipe can be active, which results in concurrency within the recipe's state machine. As concurrencies occur, the hardware of the SCS has to be able to cope with concurrencies, e.g. using different threads. Each procedure is allocated to a specific module as a resource which executes the invoked services of the procedure. Whenever concurrent active procedures are allocated to the same module, it has to be ensured that the same service is not addressed by these procedures. Therefore, a method is needed to verify the feasibility of the designed orchestration. Furthermore, the state machine in the recipe as well as the state model of the services are deterministic, as commands (in the service's state model) or states (in the recipe's state machine) always result in distinct behavior of the state machine or the state model, respectively. In the case of different service commands which occur at the same time, the command of higher priority is taken as the input of the service. Therefore, the service commands are ordered from high to low priority according to the state model of the services as follows: priority 1: abort; priority 2: stop; priority 3: hold; priority 4: start, pause, reset and unholding: As all service commands of the lowest priority 4 are only handled by the service in distinct service states, a distinct behavior of the service in total is ensured.

As the possibility of non-occurring events for the firing of transitions exist (e.g. in the case of a failed sensor), the controller of the module has to track the time of active service states and has to send a fault message to the SCS if the time exceeds a defined limit.

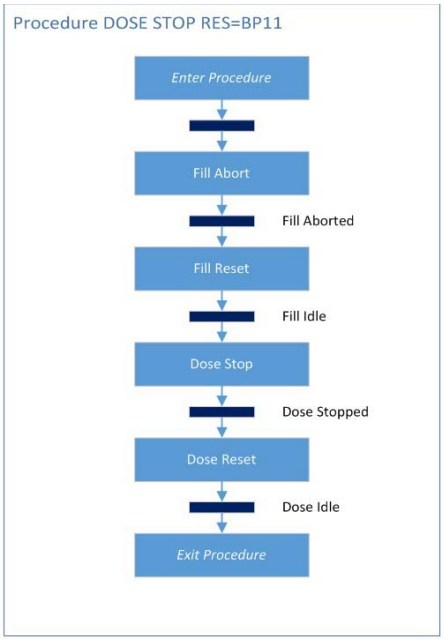

Fig. 3 Sequence recipe example [11]

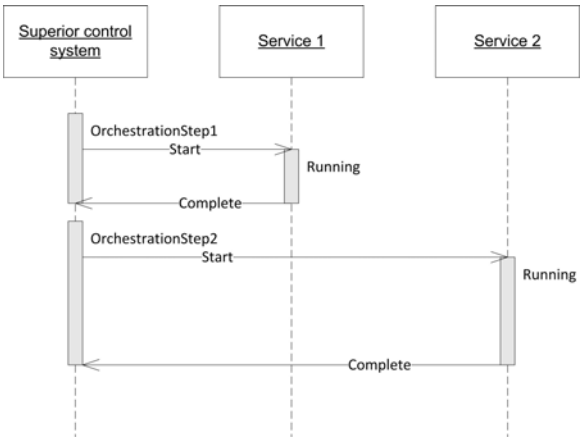

Fig. 4 Example of the communication between the orchestration unit and the services

To support the operator, all states of all active services of all modules are indicated within the HMI of the SCS, and the operator can switch into different operation modes in the SCS. These different operation modes have been implemented by different state machines of the services, following the concept of PackML [12]. So far three different operation modes have been identified as necessary:

1. Automatic: Within this mode, the services follow the instructions of the implemented orchestration within the SCS. A possible implementation on SCS level can be a sequence recipe.
2. Manual Central: Within this mode, the services and the field devices that have been selected for manual control by the module vendor can be controlled from the HMI of the SCS. SC-transitions cannot be forced to fire as the event for firing still stems from within the service/module.
3. Manual Local: Within this mode, the services and field devices of a single module can be controlled by means of a local panel at the module. This operation mode is needed for maintenance or commissioning of single modules.

To track the active operation mode, each service changes into a different instance of the defined state model whenever the operation mode of the service is changed. Thus, concurrencies within a service are excluded, as only one state model per service is active at a time. This concept differs from the concept of PackML in one aspect: The state models of each operation mode in modular process automation have to be of the same structure, but the executed programs within the corresponding states can differ. In PackML, on the contrary, the state models can differ additionally in their structure for each operation mode [12]. In the case of same structures of the state model in different operation modes, the change of the operation mode can be executed more easily between the same waiting states of each state model, as the module's actuators have the same status.

IV. Simulation Environment

As a part of the work of the authors, a simulation environment has been created to test the developed concepts. The simulation environment is divided into two different parts: In Matlab Simulink Stateflow, the services of the modules and the state machine of the orchestration unit have been implemented; whereas the visualization is a platform independent web application based on Foundation for Apps (foundation.zurb.com) and NodeJS (www.nodejs.org). The structure of the simulation environment is shown in Fig. 5 and described in detail in [11].

Within the part of Matlab Simulink Stateflow, the services have been implemented in statecharts. To implement the hierarchical layers of the different groups within the state model of the services, the states have been grouped according to Fig. 2, as shown in the lower right part of Fig. 6. The state machine of the recipe is structured in different procedures, which are represented by group states. The concurrencies between the procedures have been implemented by the decomposition mode AND within Matlab Simulink Stateflow. To apply this, the state which contains all procedures is set to decomposition mode AND. This results in a parallel simulation in different threads of the processor.

The communication between the Matlab part and the visualization has been implemented via an OPC UA server with a corresponding model which provides all necessary communication nodes. Those are the state variables representing the service states, which are written within Matlab, and the control variables for manual service control, which can be set within the visualization. As Matlab Simulink Stateflow does only support OPC DA but not OPC UA communication yet, a wrapper between OPC DA and OPC UA was necessary and has been implemented. The required communication nodes and all further information (e.g. regarding services, HMIs, Key Performance Indicators (KPIs)) are described within the MTP. Based on this information, the visualization and the OPC UA server namespace can be generated automatically.

Within the visualization, different information is provided to the operator (e.g. plant key performance indicators (KPIs), a plant overview, orchestration control and different layers of detail information). As this paper focuses on the service aspect, in the following a short introduction to the service visualization will be given. A screenshot of this visualization part is shown in the top left corner of Fig. 6, where a list of available services of the currently selected module is provided. The current state

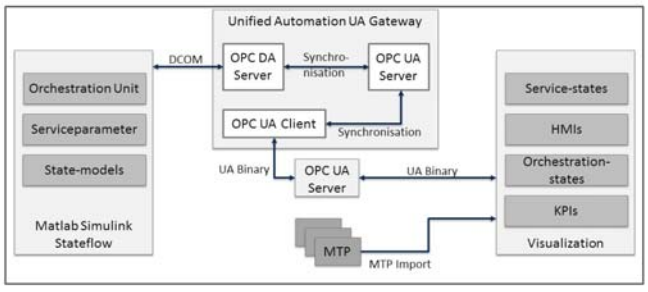

Fig. 5 Structure of the simulation environment [11]

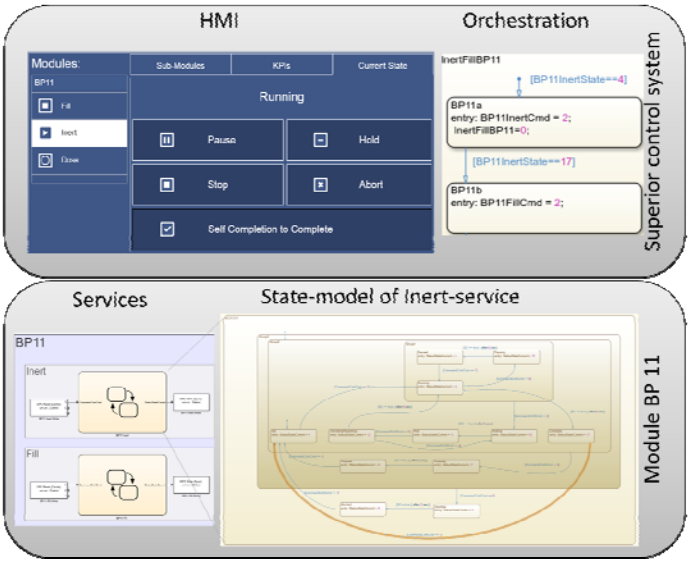

Fig. 6 Implementations within the simulation environment

tab provides further information about the currently selected service and its state. The Matlab simulation provides the corresponding values via the OPC UA server. If the operation mode is set to Automatic, the operator has a visual feedback limited to the next transitions which are reachable in the state model, but cannot control the service manually. In Manual Central operation mode, the operator can initiate possible service transitions on the screen. The resulting command is written to the Matlab simulation, which reacts accordingly and provides the corresponding state change feedback.

V. Case Study

A. Introduction to the NAMUR example case

The NAMUR/ZVEI working group 1.12 defined the NAMUR example case. The case comprises a chemical process with an endotherm reaction which is carried out in a modular process plant. Within the plant six modules of three different types are in operation: Dosing modules (named BP), a mixing and reacting module (CM05) and a distillation module (KW06). The BP modules have identical structure and only differ in their label. Each BP module offers three services: Inert, Fill and Dose. The CM05 module offers two services: Heat and Run, whereby the service Heat controls the valves which are the interfaces to an external heat exchanger. The KW06 modules offers three different services: Inert, Start-up and Shutdown. A more detailed definition of the functionality of each service is not relevant here.

B. Implementation within the simulation environment

This example case, described in the previous sub-section, has been implemented in the simulation environment Matlab. The services of all four BP-modules, the KW06-module and the CM05-module have each been implemented in the three different operation modes Automatic, Manual Central (ManCen) and Manual Local (ManLoc). Furthermore, the orchestration unit of the SCS and a physical model of the energy consumption of the process have been implemented in Matlab. The services have been implemented as shown in the lower right corner of Fig. , following the state-model of [3]. So far, the change between the operation modes has to be done manually, but an automatic handling of changes of the operation mode, which is typically a function of the process control system, will be implemented in the future. The orchestration unit is separated into four phases: RecipeIdle, StartUp, RunningMode and Shutdownmode, whereby the phases are represented by states. The transition from RecipeIdle to Startup fires if the Command Start is set in the visualization. After the StartUp, the orchestration changes into the RunningMode without any further command, because the implemented case is a continuous process which needs only to be started and runs afterwards until it is stopped. The orchestration phases are shown in Fig. 7. The orchestration phases itself have been implemented as group states which contain all needed procedures implemented as hierarchical states, as described before. In the StartUp phase, all BP-modules and the KW06-module are rendered inert first by the services inert in each module, which is first started (InertCmd = 2) and, after the state changed to complete (InertState = 17),

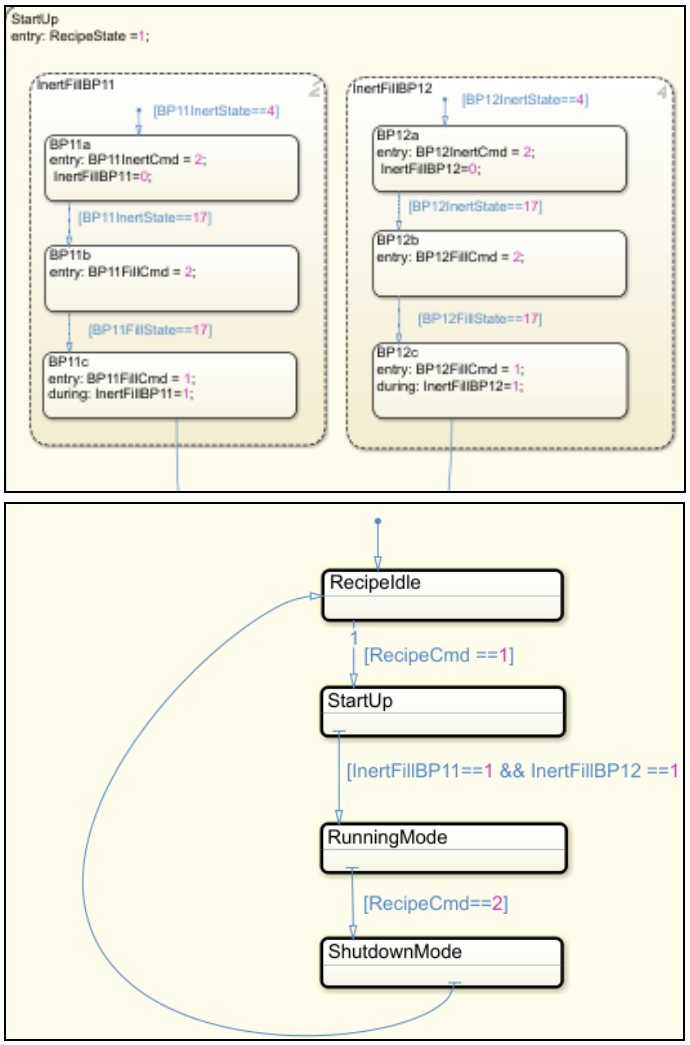

Fig. 7 Orchestration phases (lower part) and an execrpt of the procedures of the *StartUp* phase (upper part) in Matlab Simulink Stateflow

the service fill is started (FillCmd = 2). When the service fill has reached the state complete (FillState = 17), the inertization of a BP-module is finished. All other procedures are executed in the same way, but including different services, service commands and service transitions, as shown in the upper part of Fig. 6. The implementation shows the applicability of the approach, as all services can be controlled in a state-based manner via a state machine in the orchestration unit.

C. *Implementation in an industrial environment*

The concepts which were implemented in the simulation environment first, as described in sub-section V.B., have also been transferred to an industrial environment. Thereby the Matlab simulation was replaced by an industrial controller, which provides an own OPC UA server. On the controller, the state machine has been implemented. Within each state, corresponding actions are executed. In the example, a pump is used as a small dose module, which can be controlled via calibrate and dose services. Further, each service has parameters which can be specified. For example, the flow rate of the pump can be set (by the operator) and controlled (by a service of the module). Therefore, in the Running state, the calibration is used to calculate the necessary internal value for the pump speed. The OPC UA server of the controller provides for each service the state and control variables. The corresponding communication nodes are described within the MTP. By this, the information can be used again to generate the visualization automatically and the resulting visualization can directly control the dose module by means of the provided services.

VI. CONCLUSIONS AND FUTURE WORK

Within this contribution, an approach for the state-based control of services within modular process plants has been introduced. The contribution shows the feasibility of state-machines for the control of services and demonstrates a state-machine-based way for the orchestration of services. The presented concepts have been successfully tested in a simulation environment and in an industrial environment.

In future work, the presented concepts will be extended by a service relation concept, a handling of operation modes as well as a method for the verification of orchestrations. In addition, the simulation environment will be continuously improved by the integration of a physical model of the Namur example case plant. The industrial implementation will be improved by an implementation of an orchestration unit for the pump to operate the services also in the operation mode Automatic.

REFERENCES

[1] Bieringer, T., Bramsiepe, C., Brand, S., Brodhagen, A., Dreiser, C., Fleischer-Trebes, C., Kockmann, N., Lier, S., Schmalz, D., Schwede, C., Schweiger, A. and Stenger, F. (2016). Modular Plants - Flexible chemical production by modularization and standardization - status quo and future trends.

[2] Lier, S. and Grünewald, M. (2011). Net Present Value Analysis of Modular Chemical Production Plants, Chemical Engineering & Technology, vol. 34, pp. 809–816.

[3] IEC 61512-1 ed. 2 (unpublished). Batch Control - Part 1: Models and Terminology.

[4] ZVEI (2015). White Paper: Module-Based Production in the Process Industry – Effects on Automation in the "Industrie 4.0" Environment: Recommendations of the Modular Automation Working Group Following Namur Recommendation NE 148. German Electrical and Electronic Manufacturers' Association.

[5] VDI/VDE/NAMUR 2658-1 (2017). Automation engineering of modular systems in the process. (Greenprint)

[6] NE 148 (2013). Automation Requirements relating to Modularisation of Process Plants, Leverkusen

[7] Bloch, H., Fay, A. and Hoernicke, M. (2016). Analysis of service-oriented architecture approaches suitable for modular process automation. In 21th IEEE Conference on Emerging Technologies and Factory Automation (ETFA): September 6 - 9, 2016, Berlin.

[8] Bloch, H., Hoernicke, M., Hensel, S., Hahn, A., Fay, A., Urbas, L., Knohl, T. and Bernshausen, J. (2017). A Microservice-Based Architecture Approach for the Automation of Modular Process Plants. In: 22nd IEEE International Conference on Emerging Technology and Factory Automation. Limassol, Cyprus, 13-15 September.

[9] Urbas, L.; Krause, A.; Ziegler, J. (2012). Process control systems engineering, p. 41. Oldenbourg Industrieverlag, München.

[10] IEC 61512-1 (1997). Batch Control - Part 1: Models and Terminology.

[11] Bloch, H., Hensel, S., Hoernicke, M., Hahn, A., Fay, A., Urbas, L., Wassilew, S., Knohl, T., Bernshausen, J. and Haller, A. (2017b). Model-based Engineering of CPPS in the process industries. In 15th IEEE International Conference on Industrial Informatics (INDIN). Emden, Germany, 24-26 July.

[12] ANSI/ISA-TR 88.00.02-2015 (2015).: Machine and Unit States: An implementation example of ANSI/ISA-88.00.01.

[13] IEC 61131-3 (2013). Programmable controllers - Part 3: Programming languages.