

IEC 61499 Component Based Approach for Batch Control Systems

D. Dimitrova^{*/+}, S. Panjaitan⁺, I. Batchkova^{*}, G. Frey⁺

^{*}University of Chemical Technology and Metallurgy, 8. Kl. Ohridski Blvd. 1756 Sofia, Bulgaria
(e-mail: idilia@uctm.edu)

⁺University of Kaiserslautern, Department of Electrical and Computer Engineering,
Erwin-Schrödinger-Str. 12, 67663 Kaiserslautern, Germany
(e-mail: {dimitrova | panjaitan | frey}@eit.uni-kl.de)

Abstract: In process automation batch processes play a dominant role. With ISA SP88 and its IEC standard equivalent IEC 61512, there is a standard available covering the description of batch processes and plants over several hierarchical layers. For the instrumentation and automation components in all industrial systems there is a trend towards distributed solutions. The function block oriented IEC 61499 standard describes models to implement distributed control systems. In this contribution a new way to combine the concepts of SP88 for design with the models of IEC 61499 for implementation is proposed. To describe the control sequences Signal Interpreted Petri Nets (SIPN) are used to get a more formal model of the control than it is possible with the Procedure Function Chart (PFC) proposed in SP88. Based on this description basic functions for the control as well as the corresponding activation sequences are determined. The basic components are implemented by function blocks according to IEC 61499. The interconnection of the function blocks according to the required sequences is implemented using a scheduler concept. This concept allows re-configuration of the control without altering the function block diagram. Hence the proposed approach offers analyzable formal models, re-usable basic components, and easy re-configurability. The approach is illustrated using the Festo Mini Pulp Process (MPP).

1. INTRODUCTION

Hybrid processes are often found in chemical industry. Batch control as described in the ISA SP88 standard is an important concept in this area. It describes the continuous production of finite quantities of materials (batches) in chemical processes. As a result, the complexity of the process control is reduced.

As in other industries, in the process area flexibility of the production system is an increasingly important requirement. This implies several demands on the control systems such as reusability, interoperability, convertibility and easy customization. To fulfil all these requirements, a development process for flexible batch control is needed.

The main goal of this research is to investigate and to propose a feasible development process for batch control. The methodology combines the SP88 standard for batch process design with the IEC 61499 standard for distributed control. A concept for easy re-configuration based on task scheduling of basic functions in IEC 61499 is added to improve the flexibility of the resulting control system. Furthermore, Signal Interpreted Petri Nets (SIPN) are used to describe the execution schedules in a formal way for analysis of the system.

The paper is organized as follows: the next section briefly introduces IEC 61499 and SP88. Section 3 presents the proposed approach and Section 4 shows its application to the Festo Mini Pulp Process (MPP) as a case study. Conclusions and an outlook are given in the last section.

2. ISA SP88 AND IEC 61499 STANDARD

2.1. Batch Control based on SP88

ISA SP88 and its IEC standard equivalent IEC 61512 provide a guideline for designing batch control applications for manual as well as automatically controlled processes (ISA, 1995; IEC, 1997). The three control types in batch manufacturing are basic, procedural and coordination. These control types are applied to the control and equipment modules. To examine the relation between equipment and procedural control, SP88 defines a hierarchy of four types of recipes, i.e. general, site, master and control recipe (cf. Fig. 1).

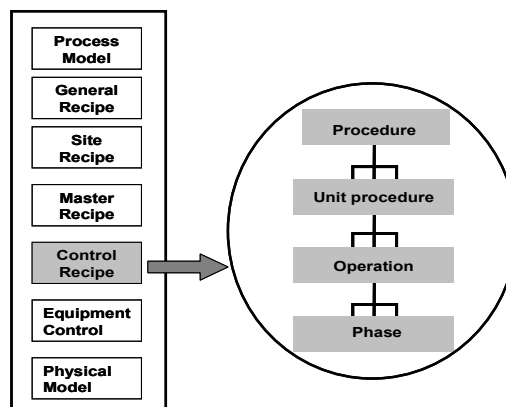


Fig. 1. ISA S88 Structure

This paper considers the control recipe and the procedural control. The control recipe contains all necessary information for the control of a batch process. Procedural Function Chart (PFC) is used to describe a control procedure hierarchically.

2.2. IEC 61499 Standard

IEC 61499 (IEC, 2005) defines a reference architecture for distributed automation systems and proposes different models including system, device, resource, application and function block (FB). A system contains interconnected devices that communicate with each other. A device contains at least one interface. Two kinds of interfaces are used for process-I/O and communication. The resource is a functional unit in a device that has independent control of its operation. An application consists of a network of FBs.

A function block as a functional software unit consists of head and body (cf. Fig. 2). Events flow on the head and data on the body. An Execution Control Chart (ECC) describes the internal behaviour of the basic FB instances. The ECC helps the programmer to decompose complex behaviour into smaller pieces called states. Each state is valid under a certain set of conditions. The states are associated with one or more algorithms and/or output events. The activation of a state implies the execution of the attached algorithms.

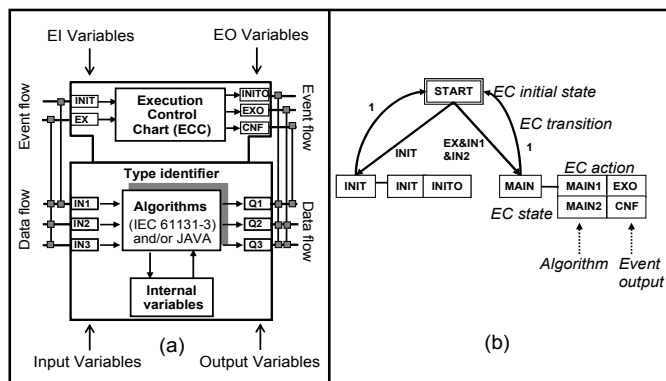


Fig. 2. (a) Basic FB and (b) ECC

3. COMPONENT-BASED APPROACH FOR BATCH CONTROL

Related works that combine the use of SP88 and IEC 61499 are presented in (Jukka, et al., 2007) and (Thramboulidis, et al. 2007). Their approaches adopt PFC according to SP88 to describe the batch procedure. The information on each operation is signal-based. After the design, a one-to-one mapping from PFC into an IEC 61499 FB network is performed. In that way, changes in the operation sequences require changes of the PFC as well as of the FB connections.

The approach presented here is related to the development of common functional components. Fig. 3 illustrates the proposed concept to combine ISA SP88 (high-level) and IEC 61499 (low-level). The goal is to provide highly reusable components concerning batch process and an easy way to reconfigure their executions. As a result, the control strategy can be flexibly managed.

By using this approach, once the generic functional components are built, their control property can be reused for a component with similar functionality. Hence, engineering time can be reduced. The proposed approach is divided into three activities: functional component development, control recipe modelling using SIPN, and mapping of the model to an IEC 61499 application.

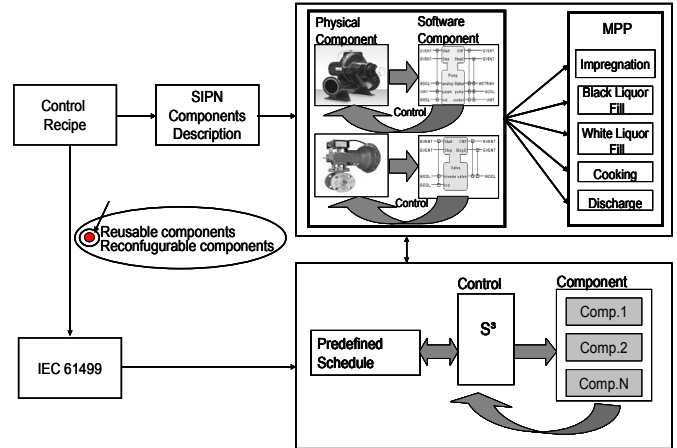


Fig. 3. Map of the suggested approach

3.1. Functional Components

Batch process control systems use sensors to measure the control variables and actuators to influence the process. Several hardware components usually used in batch control are pressure sensor, temperature sensor, level sensor, valve and pump. These devices are controlled by software controllers.

In this paper, a Festo MPP system is considered as a case study. The generic control components commonly used in this system are pump and valve. These components have a discrete feature (e.g. valve open/close, pump on/off) and continuous characteristic (e.g. proportional valve, proportional pump).

The traditional way to control them is by taking their input information, sending them to a (centralized) control algorithm and updating the output after a particular calculation is done. Hence, changes in a component require the change of this central algorithm. In the component approach, each hardware component has a corresponding software component (i.e. functional component). If several components are employed for an equipment or unit module, their software components are then compounded as a composite component. For instance, the inlet of a tank contains several valves. The functional component of valve hence can be instantiated several times. Customization is done according to valve number. Concisely, the functional components (i.e. model and source code) can be used and re-used in a control recipe as it is shown in Fig. 3. In implementation, the reuse is possible due to the instantiation and encapsulation capabilities provided by Function Block Development Kit (FBDK) as a standard tool for IEC 61499 implementations (Holobloc, 2007).













3.2. PFC and SIPN for Control Recipe

In (Jukka, et al., 2007) and (Thramboulidis, et al., 2007), PFC had been employed to describe the procedural control of each unit procedure. PFC is a well-known methodology used by industry for batch control. To allow for system verification, a formal analysis technique is needed. However, PFC does not provide a formal analysis yet. Signal Interpreted Petri Net (SIPN) is therefore considered in this paper to accommodate the need for formal techniques. Initial work on how to describe a control recipe using SIPN is reported in (Dimitrova, et al., 2007). In the paper at hand, PFC is used as procedure guideline to build the control while the SIPN is used for the formal analysis. The PFC can be implemented by any programming language used in IEC 61499 depending on the requirements and ability of the programmer.

SIPN has two basic types of nodes, i.e. places and transitions, connected through directed arcs. Places are associated with output signals. Transitions are labelled by Boolean expressions of the input signals, which serve as firing conditions. The dynamic behaviour of an SIPN model is given by the flow of tokens through the net. This flow is enabled to move from pre-place to post-place by the transitions firing. The extension of SIPN with time and hierarchical structures gives opportunity for fully description of control system functionality and requirements.

More details about the SIPN modelling concept, including the SIPN editor can be seen in (Frey, 2002). The last editor version includes model analysis and is developed by JPA² at the University of Kaiserslautern (Frey and Wagner, 2006).

Table 1. Mapping PFC into SIPN

No	PFC	SIPN	No	PFC	SIPN
1	 Start	 Initial place with token (no incoming arc)	4	 Phase	 Place
2	 Stop	 End place in net (no outgoing arc)	5	 Condition Transitions, Start and Stop for serial and parallel executions, Synchronization point and line	 Transition
3	 Procedure, Unit Procedure, Operation	 Hierarchical Place	6	 Direct Link, Synchronization line with material balance	 Arc

The mapping between SIPN and PFC is depicted in Table 1. *Start* notation in PFC is described by a place in the SIPN model that has initial marking. *Stop* notation is described by a final place. Hierarchical place in SIPN can be used to describe Procedure, unit procedure and operation in control recipe. The hierarchical place represents a composite entity of a procedure. A normal place describes a *Phase* while its inscription denotes the controlled component. The transition in SIPN can model conditional transition and synchronization among procedure levels in control recipe. The arc represents a link between procedural elements at the same level.

3.3. Mapping to IEC 61499

SIPN model, which describes the sequence of component execution, is mapped into IEC 61499 applications. The model is used as control scenario that will be implemented in FB-based system. The SIPN model with regard to the functional component is mapped into IEC 61499 by the following rules: Place's inscription denotes the related functional component implemented as one or more FBs either as basic (e.g. valve, pump, etc.) or composite entity. The token flow in the SIPN denotes the sequence of functional components' execution. Transition's inscription denotes condition (i.e. input variable) which is required to move from one component to the others. Time delay is drawn at the arc and implemented as an FB.

For changing the operation schedule, the structure of this model normally has to be changed on all levels (SIPN and FB network). An approach named Scheduler-Selector-Synchronizer (S³) is adopted to ease reconfiguration of the operation schedule to the point that the FB network does not have to be changed at all. A new FB *Scheduler* gets the schedule of operation (SOP) defined in the SIPN and computes it to be sent to FB *Selector* step by step. An example of schedule representation is, for example, SOP_{unit1} = <op1, op2, op3>. It means the schedule of unit1 is the sequence of op1, op2, and op3. FB Selector will run the regarded functional components according to the given step. The executed task will be resumed by FB *Synchronizer*. A confirmation will be sent to the FB *Scheduler* once the required task is completed. A new task is requested afterward. The details of S³ approach are presented in (Panjaitan and Frey, 2007). Its relations with SP88 and IEC 61499 are illustrated in Fig. 3.

The SIPN that describe Procedure, Unit Procedure and Operation are hierarchical elements and can be mapped into composite FBs. Their coordination can be implemented at the higher level, either in the higher composite FB or at the system application level. If the execution of unit procedures is in sequential order, the composite FBs for each unit procedure can be directly connected with each other according to their order. Otherwise, a higher-level S³ can be implemented.

4. CASE STUDY

The proposed approach is illustrated using the FESTO MPP (Festo website). Fig. 4 depicts the physical model of the MPP system. The processes are divided into five unit procedures: Impregnation, Black Liquor Fill, White Liquor Fill, Cooking and Discharge. These unit procedures are controlled by the functional components such as valves and pumps along with some analogue indicators. The reuse of software components depends on the functional requirements of each unit procedure. At the procedure level, these units will be run in series for the batch process: <Impregnation, Black Liquor Fill, White Liquor Fill, Cooking, Discharge>. After the batch is finished, a new batch can be processed.

The MPP is started by the operations of unit procedure *Impregnation*. The Impregnation unit consists of five phases. The phase is the lowest level in the procedural control recipe describing process of specific tasks or functions. Phase 1 is to open the route from impregnation liquor tank (T200) to Di-

gester tank (T300) and turning on the pump P200. Phase 2 is to close the outlets of T300 if LS+300 is true. After a delay time (Ti), Phase 3 is done to turn off the pump and close the previous opened route. Phase 4 is to depressurize the Digester by opening the exit valve for 2 seconds. Phase 5 is to close the exit valve.

The second unit procedure, *Black Liquor Fill* unit, is responsible to open the route from Digester T300 to the black liquor tank T400. The process will be finished if the level measurement LI400 is true. The third unit procedure is *White Liquor Fill*. It is concerned to control the valve V101. Previously the valve is opened. The operation is stopped when the level measurement rises up to 120 mm. For safety reasons, pump P100 must not be started if the level measurement LI100 is less than 120 mm. The fourth unit procedure is *Cooking*. In this unit, a batch of material is cooked concerning some desired characteristics such as temperature, pressure and time requirements. Sensors used in this process are denoted as TI300, PI300, LI100 and Tc. After cooking operations are completed, the digester is depressurized by opening the exit valves for 2 seconds. The last unit procedure in MPP for a batch process is *Discharge*. In this unit, the product is sloped out in the storage tank through the valve V105.

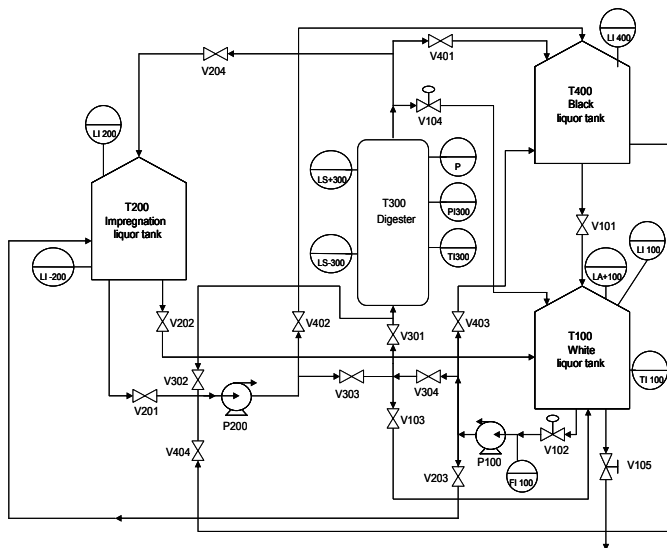


Fig. 4. P&ID of the Festo MPP case study

4.1. Functional Components in Impregnation unit

At the beginning, the elicitation of functional component requirements is performed. Basic functional components required for Impregnation unit are listed in Table 2. They are instantiations of valve and pump software controllers. The functionalities that will be executed in the Impregnation unit are listed in Table 3. These functionalities will be employed to control the impregnation unit. These functionalities are elements of three process groups in this unit listed in Table 4. Process *fill_T300_from_T200* aims to fill the digester tank (i.e. T300) by opening its outlets and route from T200 to T300. The filling process is stopped when the required level is reached. Process *pressurize_T300* aims to close the outlets

of T300 and pressurize it to maintain the air pressure inside the tank. The pressurization is performed for 1800 seconds. Finally, process *depressurize_T300* will depressurize tank T300. These three processes will be concatenated for the impregnation unit.

Table 2. Instantiation of basic components

Components	Instantiation
Valve	V104, V201, V204, V301, V303, V401
Pump	P200

Table 3. Functional components for the Impregnation

Components	Description
route_T200_T300()	Open or close the route from T200 to T300 (i.e. valves V201, V301, V303) and start or stop Pump P200.
T300_outlets()	Open or close the outlets of T300, i.e. V204, V104, V401.
fill_T300_to_limit	Filling the T300 until the level sensor LS+300 is reached
Wait(t)	Waiting time based on the given times (t).

Table 4. Process group in the Impregnation

Process	Embedded sequence
fill_T300_from_T200	{T300_outlets(open), route_T200_T300(open), fill_T300_to_limit}
pressurize_T300	{T300_outlets(close), wait(1800s), route_T200_T300(close)}
depressurize_T300	{T300_outlets(open), wait(2s), T300_outlets(close)}

4.2. SIPN model of Impregnation Unit

For the Functional components, since the different procedure parts can access the same basic functions (e.g., valve), a verification step is needed. The verification is performed to check that (especially in non-sequential operation) there are no confliction output settings. For this purpose, the SIPN model can be used. Formal analysis using SIPN proposed in this paper has three goals: (1) find conflicting outputs especially in parallel executions, (2) find undesired output combinations (e.g. referring to a combination of opened valves that is forbidden in the current setting), (3) verification of specified flow.

Fig. 5 shows the hierarchical SIPN model for the Impregnation unit in the MPP process. Hierarchical places are indicated by dotted lines. The Petri net shows how different functionalities are combined to reach the desired process (top-level net) and what elementary processes these functionalities are composed of (low-level net). For the place *pressurize_T300* the underlying SIPN is also depicted. It composes several steps of operation. In the given example, output conflict is not applicable since in this unit everything runs sequentially. Undesired output combinations from P&ID applicable to the processes can be verified by using the reachability graph or by model-checking based on the SIPN model.

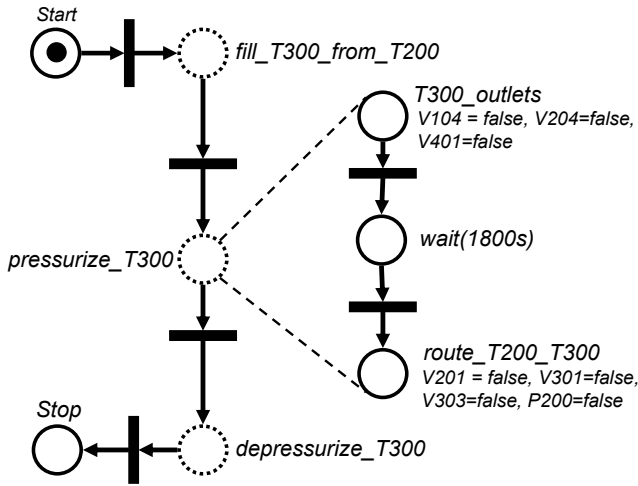


Fig. 5. SIPN model for the Impregnation unit

4.3. IEC 61499 Implementation for Impregnation

Fig. 6(a) shows a composite FB for *route T200_T300*. The FB composes three valves and one pump. The reuse of FB valve is three times (i.e., V_303, V301, and V201). Another composite FB named T300_Outlets composes three valves as well (i.e., V104, V204, V401).

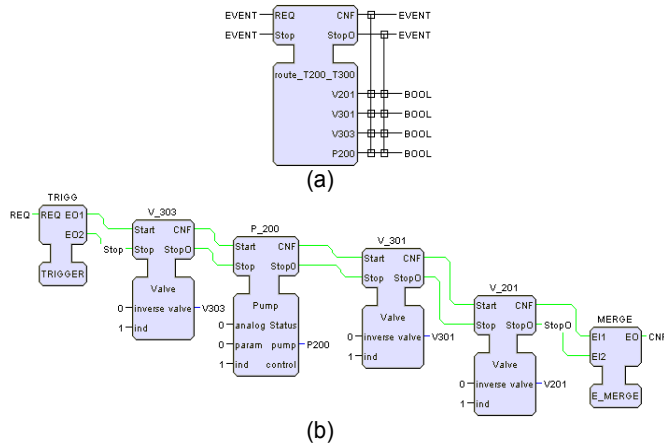


Fig. 6. Composite FB: (a) Interface and (b) Body

The implementation of the Impregnation unit in a composite FB of IEC 61499 is shown in Fig. 7. S³ architecture is applied for the corresponding FBs Scheduler, Selector, and Synchronizer. These FBs are used to control the component based on the given component schedule (i.e., SOP). In each SOP, functional components are arranged in an ordered list separated by commas and enclosed by corner brackets. The SOP of impregnation unit for normal operation concatenates the sequence of *fill_T300_from_T200*, *pressurize_T300* and *depressurize_T300* (see Table 4):

$$SOP_{normal} = \langle fill_T300_from_T200, pressurize_T300, depressurize_T300 \rangle$$

Or, by replacing process groups by functional components:

$$SOP_{normal} = \langle T300_outlets(open), route_T200_T300(open), fill_T300_to_limit, T300_outlets(close), wait(1800s), route_T200_T300(close), T300_outlets(open), wait(2s), T300_outlets(close) \rangle$$

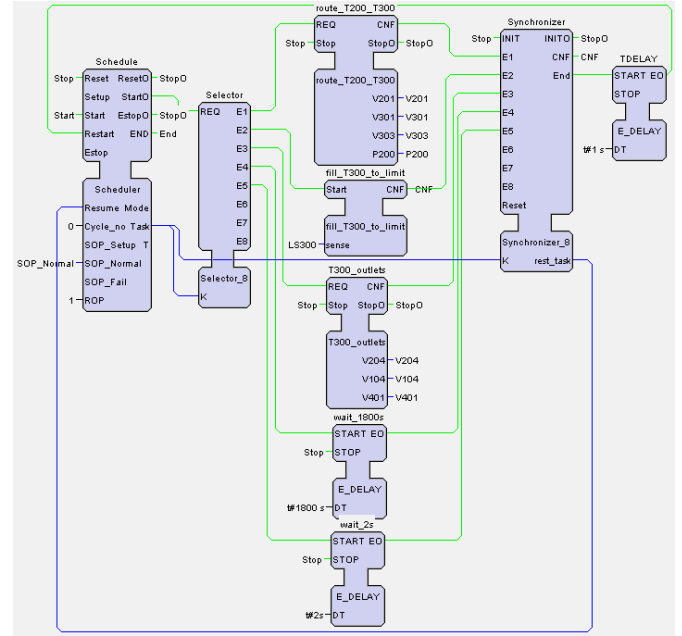


Fig. 7. FBs' network of Impregnation unit

A conditional relation between an SOP to the other can also be described. $SOP_{impregnation} \xrightarrow{t} SOP_{BlackLiquorFill}$ represents that $SOP_{BlackLiquorFill}$ can be operated if $SOP_{impregnation}$ has already been done and condition t is reached.

Furthermore, all unit procedures in the MPP are managed in the procedure level in a series order as the most case in batch control. The series order is quite simple and reduces the complexity in managing the unit procedure. Fig. 8 illustrates the control of five unit procedures using S³ architecture. Each unit is represented by an FB, i.e. impregnation, BlackLiquorFill, WhiteLiquorFill, Cooking, and Discharge. FBs Scheduler_MPP, Selector_MPP, and Synchronizer_MPP are used to control those units using S³ approach. Their SOP for normal operation based on the name of the controlled FBs is as follows:

$$SOP_{normal_MPP} = \langle impregnation, BlakLiquorFill, WhiteLiquorFill, Cooking, Discharge \rangle$$

If the order among unit procedure is not serial, then the required sequence can simply be given as a new SOP normal process of MPP. Likewise when there is looping to other unit in the middle of procedure control. S³ architecture can also be used by configuring the corresponding SOP. For instance, SOP for MPP procedure can be like $\langle Impregnation, BlackLiquorFill, WhiteLiquorFill, Impregnation, Cooking, WhiteliquorFill, Discharge \rangle$.

In Fig. 8, S³ architecture is applied in two units, i.e. Impregnation and Cooking. Note: For FB Scheduler, the representation of SOP on FB programming level is given by

integer task values determined by a binary encoding. Task values for Impregnation are: $route_T200_T300 = 1 = 2^0$, $fill_T300_to_limit = 2 = 2^1$, $T300_outlets = 4 = 2^2$, $wait_1800s = 8 = 2^3$, $wait_2s = 16 = 2^4$. Based on these values, the SOP yields, $SOP_{normal} = \langle 4, 1, 2, 4, 8, 1, 4, 16, 4 \rangle$

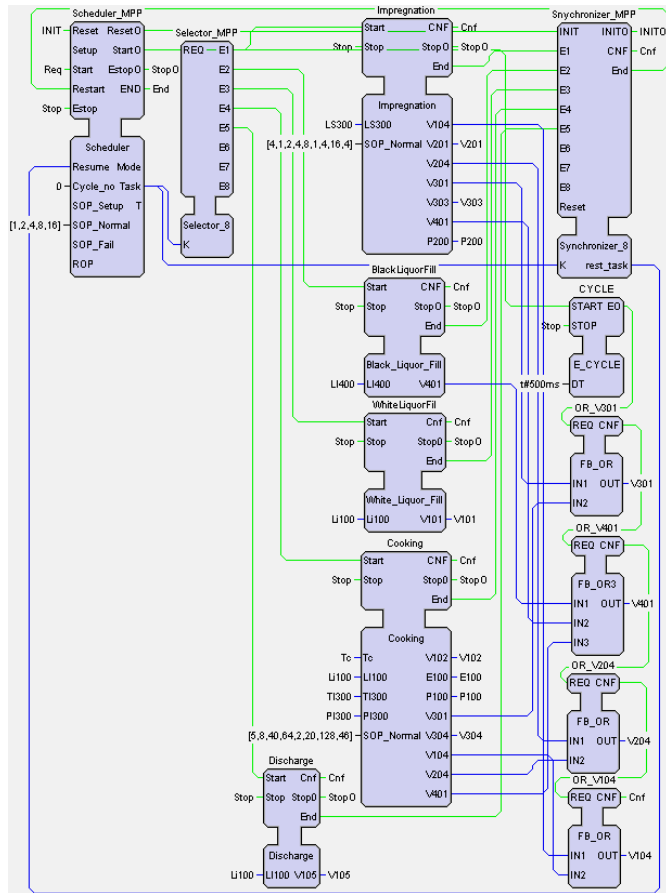


Fig. 8. FBS' network of MPP procedure control.

Other units contain a simple controller along with its constraint. There could be an output connected to more than one unit procedure. For that purpose several additional FBs representing logical OR are added. Direct connection can cause a conflict because different values come to an output. The OR function in the controller is used to build the connection. However it is not intended to solve potential conflicts. These should be analyzed prior to implementation using the formal SIPN model. This analysis is especially important when several processes should run on a system concurrently.

5. CONCLUSIONS AND OUTLOOK

The proposed approach for batch control development process has been elucidated in this paper. It traces the migration path to IEC 61499 control application. The approach consists of three main parts. First, functional component development (basic control) is to improve reusability of control modules. The development of software components for the related hardware components is influenced by some common properties in Batch Control to improve flexibility. It is achieved

by using generic functional re-usable control components that can be reused to control different equipments and processes with similar functionality. In the given example (i.e. MPP system) the total reuse of functional component valve is 15 times either proportional or discrete valve. Second, SIPN model has been used to describe the procedural model in control recipe with regard to ISA S88. In that way, formal analysis can be performed. Third, mapping from SIPN model into the FBs' network is done by using S³ architecture. The model is interpreted as a Schedule of operation (SOP) that will be managed by S³ components. As a result, the approach increases the design flexibility and paves an efficient way to reconfigure the operation schedule.

Further work will be directed to a more complex case study for batch processes and the definition of other generic functional component which are usually implemented in this area.

REFERENCES

Dimitrova, D., G. Frey and I. Batchkova. (2007). Sequential control at the supervisory level of batch plant using signal interpreted Petri nets. In: *Proc. of the International Conference "Automatics and Informatics'07"*, Sofia-Bulgaria, October, pp.V-17 ÷ V-20.

Festo website, Festo Didactic Mini-Pulp Process (MPP) System, <http://www.festo-didactic.com>.

Frey, G. (2002). *Design and formal analysis of Petri net based logic control algorithms*, Dissertation, University of Kaiserslautern, Shaker-Verlag, Aachen.

Frey, G. and F. Wagner (2006). A toolbox for the development of logic controllers using Petri nets. In: *Proc. of the 8th International Workshop on Discrete Event Systems (WODES)*, Ann Arbor, Michigan, USA, pp. 473-474.

Holobloc (2007). Official website for Function Block Development Kit (FBDK) tool, www.holobloc.com.

IEC (1997). *Batch control, part 1: Model and terminology*, The International Electrotechnical Commission (IEC), Final draft, 1997.

IEC (2005). *International Standard Function Block – Part 1: Architecture*, IEC Press.

ISA (1995). *Batch Control Part 1: Model and terminology*, The International Society for Measurement and Control, ISA Press, ISA – S88.01-1995.

Jukka P., J. H. Christensen, S. A. Sierla, and K. O. Koskinen (2007). A migration path to IEC 61499 for the batch process industry. In: *Proc. of the 5th IEEE International Conference on Industrial Informatics (INDIN'07)*, Vienna, Austria, Vol. 2, pp. 811-816.

Panjaitan S. and G. Frey. (2007). Development process for distributed automation system combining UML and IEC 61499, *Int. Journal of Manufacturing Research*, Vol. 2, No. 1, pp. 1-20, Inderscience Publisher.

Thramboulidis K., S. Sierla, N. Papakonstantinou and K. Koskinen (2007). An IEC 61499 based approach for distributed batch process control, In: *Proc. of the 5th IEEE International Conference on Industrial Informatics (INDIN'07)*, Vienna, Austria, Vol. 1, pp.177-182.