Interview Guide V2 EN

- Welcome

Hello and thank you for taking the time to do this interview with me. [redacted]. Would you perhaps like to briefly introduce yourself and explain your connection to BPMN or to process engineering, manufacturing engineering or process modeling? I would like to ask you not to state your name in the process, but only the following information:
- Job title and description of employer
- Basis of expertise on the research topic
- Education or professional background
- Work experience (e.g. more than 10 years)

- Introduction

Our research focuses on developing a methodology to represent continuous processes in BPMN and make them executable in a workflow engine. For this task, we have been working on BPMN extensions for continuous processes. Why continuous processes? Because discrete processes have already been addressed in other research and do not present the same difficulties in being represented correctly using BPMN. BPMN is already a widely used standard in business process management and has found its way into manufacturing. Discrete manufacturing processes can already be modeled using BPMN 2.0.
Basically, we want to introduce a methodology to represent such processes in such a way that they can be understood by anyone in a company, from engineers to managers. This could be achieved by using this notation. Another benefit is also that there are already a number of workflow engines - applications that allow these process models to be executed based on the logic implemented for each symbol. We are working with a web-based application that is extensible and has implemented several communication interfaces. Therefore, another advantage is interoperability in this context compared to other proprietary, rigid software applications. We want to find out if this technique can also be used to implement digital images. Since digital images are used to represent a physical system or process in digital form - usually using data or mathematical models - we had to find a way to represent the flow of continuous processes as they are known from the process industry. For this reason, we focused on modeling control loops.
The process models should be easy to understand by people with different backgrounds using BPMN. The interviews are conducted to find out how process and control engineering and techniques from business process modeling can be combined and how initial results are perceived by experts like you. We also want to find out if there are any weaknesses identified by experts and how we can address them.

- Specific introduction

- "Digital Twin":
There are different methods to simulate things from the real world, for example real machines. Partly, however, one notices that it would take more parameters than with normal simulation methods to completely represent a machine in the same way it behaves in reality. A digital twin tries to get as close as possible to the real behavior of a machine or other objects. This should lead to the fact that if something is triggered with a real machine, the digital twin shows the same or a similar behavior.

- Continuous Processes
I would like to explain continuous processes with examples.

If you take beer brewing here, there are 2 possibilities. The discrete - i.e. non-continuous variant - would be if you put the ingredients into a closed vessel, 10 liters of water, etc. and simply let the brewing process proceed step by step. In the end, a limited amount of beer will come out.
The other continuous variant would be when you don't have a completely closed vessel, but interconnected vessels where ingredients are added again and again and beer is taken out again and again. This goes on all the time, so you can't trace which liter of water goes with which liter of beer. A partial process takes place in the first vessel, while at the same time the last process step takes place in the last vessel before the beer is finished.

- Closed loop systems:
A closed loop is that logic in the form of hardware or software that makes continuous brewing possible.
When you have a process like continuous brewing, you have to look at how to avoid bad beer while the process is running. You want to keep the quality at a certain point. With incremental brewing, you only have the 10 liters where something can go wrong and then you do better with the next 10 liters.
But what if you run the brewing process all the time and beer is being produced constantly? Then you have to check the process and see that you get the right quality of the beer. This means testing or measuring values that describe the quality, checking how these values differ from optimal values, and reacting accordingly - if something is wrong with the sugar or alcohol content, the mixing ratio must be changed.
This means that in a closed control loop, certain values are checked while the process is running, these are compared with optimum values, and depending on the deviation, the system reacts accordingly.

- Continuous Processes

From a computer science perspective, continuous processes consist of a constantly repeating sequence of state queries and regulations (S&R). In each case, S&R are traditional pieces of code that refer to sensors or actuators.

In order to consistently formally describe, model, and subsequently execute such continuous processes, we have identified the following characteristics.

1) Would you rate the following characteristics as important or unimportant? Please give a reasoning for your response:

Characteristics	**Important (yes)**	**Unimportant (no)**	**Why?**
1. Different S&R Combinations are independent and may run in parallel			
2. Regulations always follow state queries			
3. The duration of each S&R combination is limited			
4. If state queries deliver certain results, the system is shut down			
5. Prior to a shutdown the system needs to be transitioned into a consistent state			
6. The resulting system must be understandable			

for people			

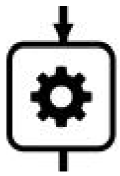
2) Can you name any graphical features that you find important for modeling continuous processes? Do they perhaps result in features that we have forgotten?

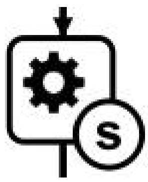
3) In your opinion, what are the challenges in modeling continuous processes?

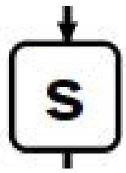
- BPMN Extension Model

Introduction to extensions

I will show you processes modeled with BPMN 2.0 and with our extensions. The extensions are intended to provide predefined modeling conventions for routines commonly used in process and control engineering, and to help visualize the differences between parallel paths in the process models. The processes are modeled in the [redacted] ([redacted]). For comprehension of the [redacted] models, three additional symbols need to be explained.







Service Call (HTTP Request)

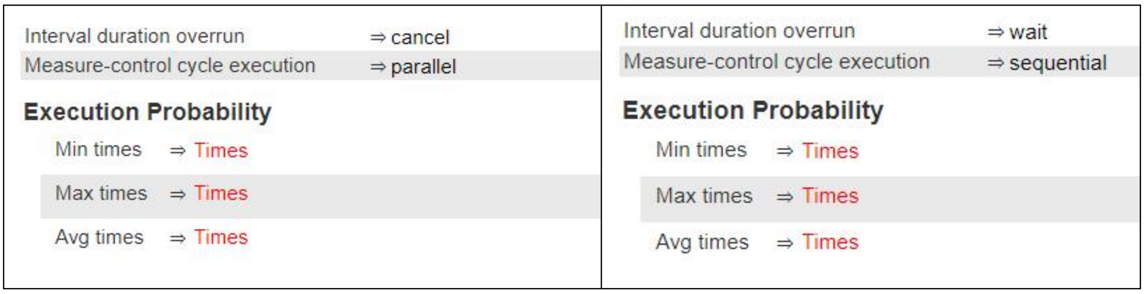
Service Call with Script (HTTP Request)

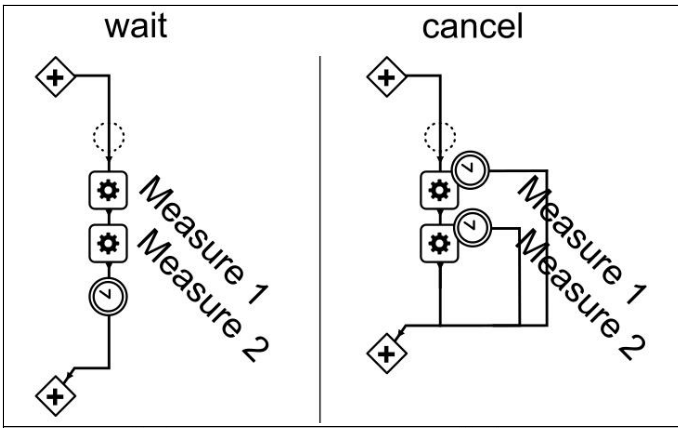
Script

Extensions

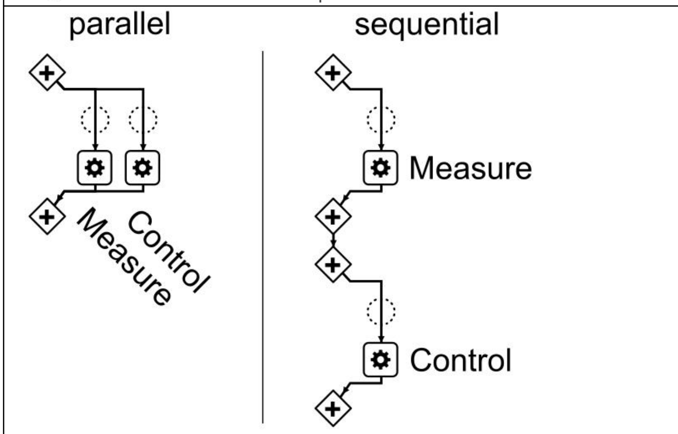
- Closed-Loop Subsystem Gateway

The gateway is a combination of an inclusive and an event-based gateway. It contains branches or edges that are triggered for the state polling and regulation phases of the cycle, as well as branches that are executed when abort events are received. The events and tasks in each edge are independent of each other. Thus, we fulfill the first of the above features - that individual progressions are independent of each other and they are executed in parallel. The gateway also allows defining the interval duration of each cycle as well as exceedance conditions (wait - cancel) and execution order for state queries and regulations (S&R) or measurement and control tasks.





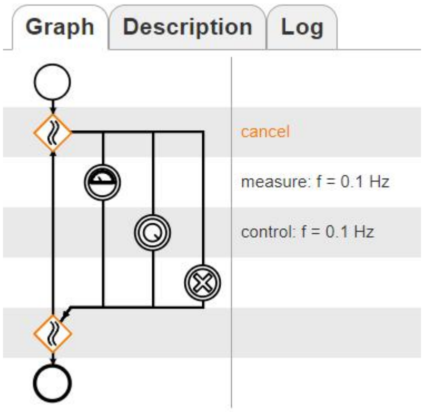
If **wait** is selected, the next iteration starts when all branches are finished and the specified interval duration is reached.
When **cancel** is selected, the interval duration defines exactly the time in which each branch is to be terminated. If the tasks in a branch finish faster, the branch will wait. If not all tasks are finished yet, they will be canceled.



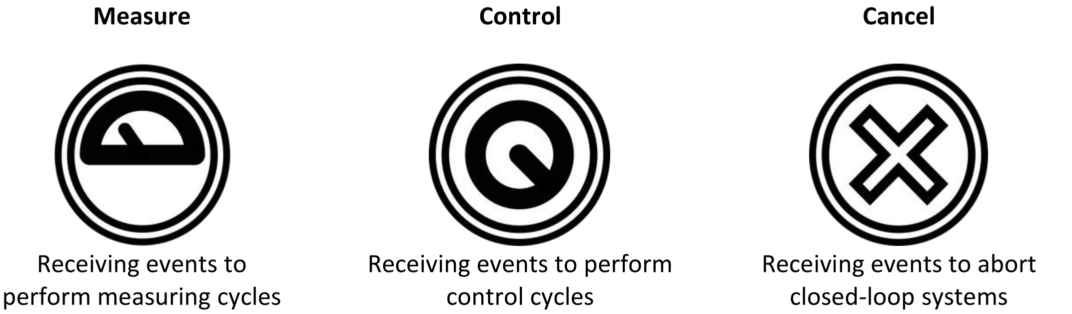
With **parallel**, the tasks are executed in parallel according to measure and control events.
With **sequential**, the tasks after control events are executed only after all tasks after measure events have been completed.

- Intermediate Catching Events

In a closed-loop subsystem, specific events are expected to fall into one of the following three categories - events for state queries/measurements, events for rules or regulations, and events for the interruption of the closed-loop subsystem. There is at least one edge for each event category that originates from the gateway. The edges indicate which tasks are running side by side.
As soon as these events occur, the tasks that are arranged in the edges after them are also executed.
Here is a picture of a closed-loop subsystem in which only events of the three categories are modeled without any tasks following them.

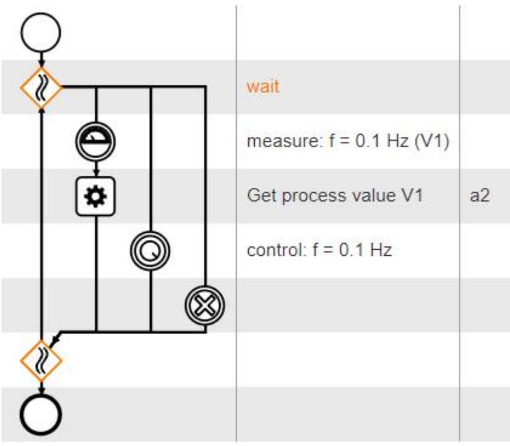


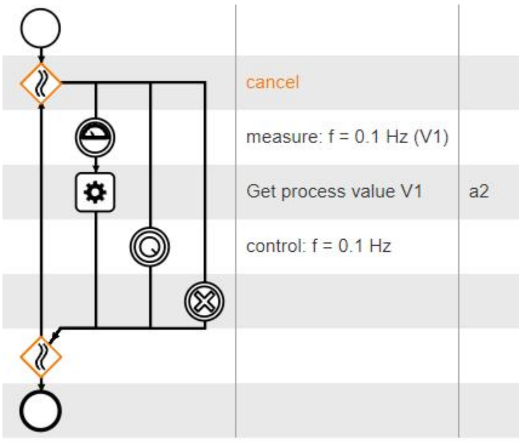
The three event categories, which we defined, are as follows:

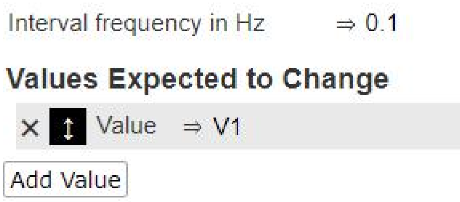


These symbols indicate the purpose of the following tasks. These tasks are only executed when the events are triggered. This means that the measurement event indicates that the subsequent symbols only indicate measurement processes or state queries. The same applies to regulation (control) and abort events. For state queries and regulations (measurement and control events) we can define a cycle time. This allows us to define the duration of adjustments in the system. Depending on whether the closed-loop subsystem follows a parallel or sequential or a wait or cancel approach, the execution runs differently. These conditions can be used to define the extent to which adjustments are made to the system.

Here you see a closed loop subsystem with a task for a measurement. In this case, the event for the measurement is triggered every 10 seconds. After that the value V1 is fetched or measured. "wait" means here that a new cycle starts only when the measurement is done i.e. the process in this edge is finished. With "cancel" the new cycle is started automatically after 10 s.



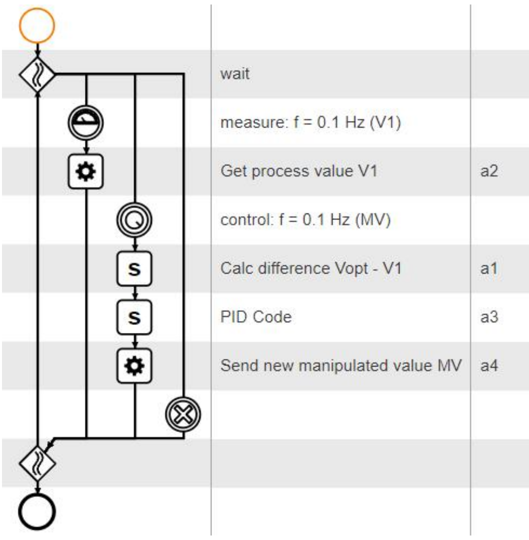


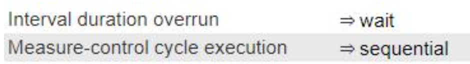


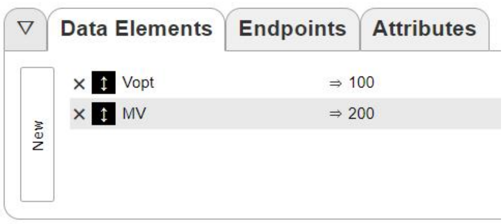
In Measure, besides the frequency of the measurement event, it is possible to define the value that changes during the process.

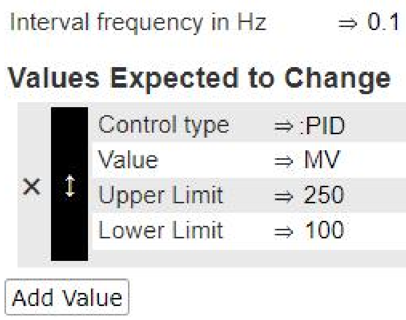
Control events can also be used to define which controller model is used - PID, PI, PD, - these controllers are represented in their mathematical form. The tasks for them are basically calculations represented in fixed sub-processes. After these calculations, the user can add tasks for further data processing. This can also be done after measurement tasks, which can also be called data acquisition tasks.

Here you can see a process model with a value that is measured and a subsequent regulation.







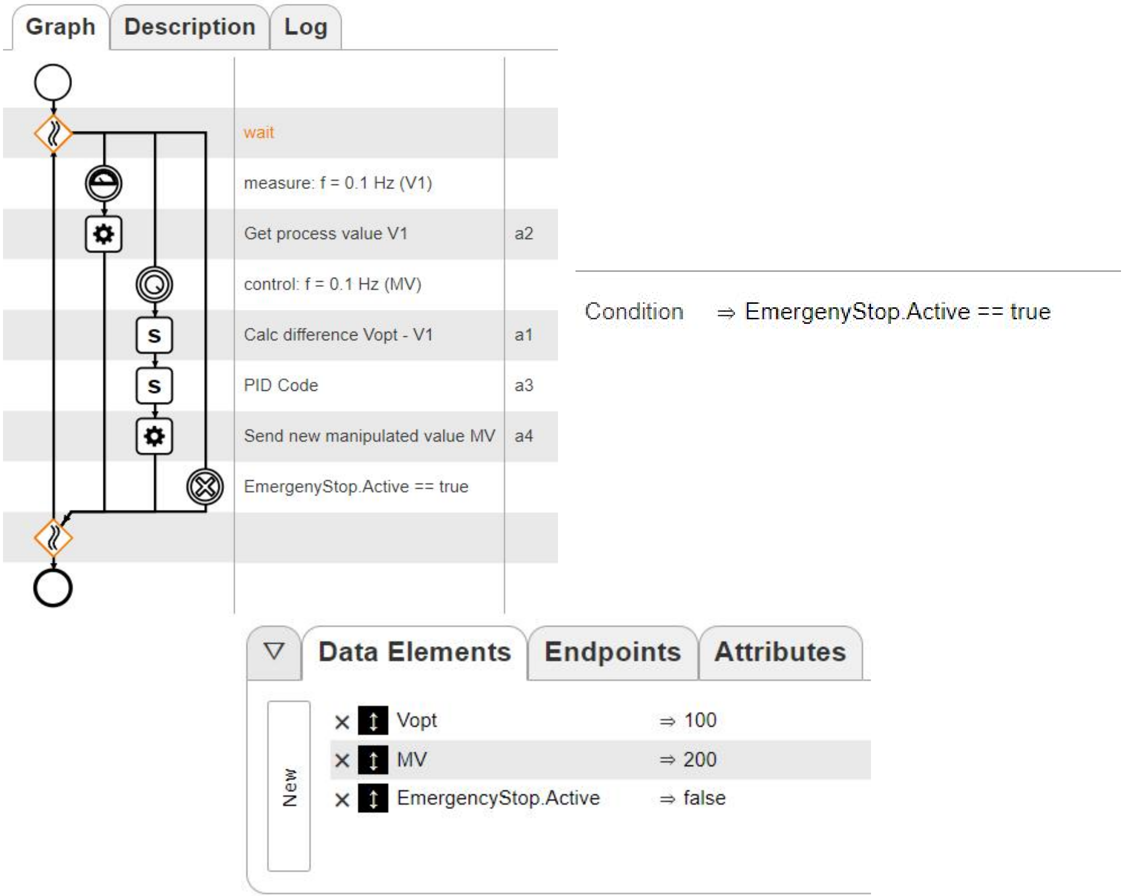


"Wait" means again that for the next cycle to start the system waits for the termination of all tasks - also for the regulation tasks. "Sequential" means that the tasks are executed one after the other, i.e. the state is first measured or queried and the regulation is carried out with this measured value. The current value V1 is subtracted from the optimum value Vopt and the new control value MV is calculated with this difference. This is then sent with a service request to the corresponding actuator - i.e. an element that actively influences the process. If you wish, you can combine the difference calculation, the control calculation PID code and the sending of the command to the system into one subprocess. In the case of Control, the type of control (here PID) as well as the new control value and its limits can also be entered.

If "parallel" were used here, the last value of V1 would be taken, for which there is no time guarantee.

State queries and regulations should be triggered at regular frequency - termination events, on the other hand, are triggered only by their termination conditions, which the user can define. An example of an abort event would be when something triggers the abort of a cycle, such as a watchdog function to monitor the maximum cycle time, or it could be an emergency stop.

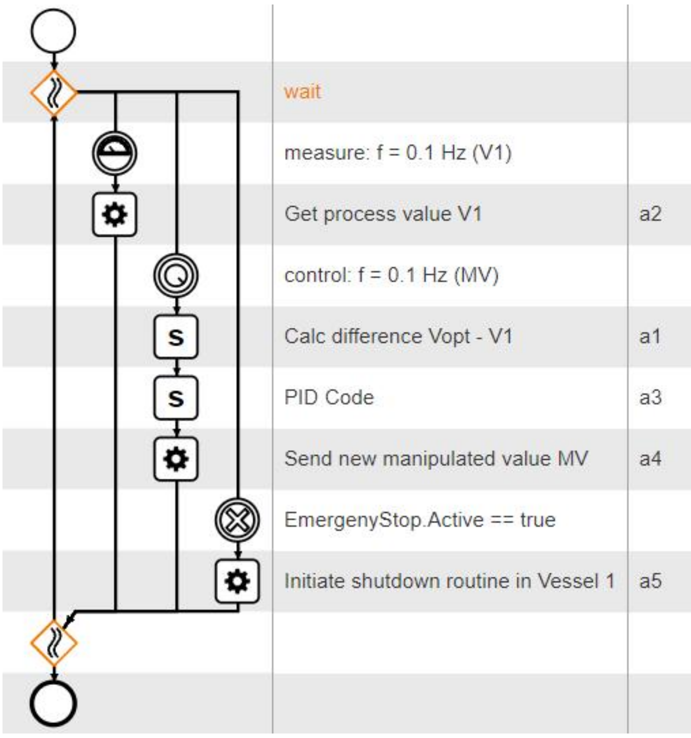
Here you can see a process with a value to be measured, a control and a termination condition.



As soon as the termination condition "EmergencyStop.Active" becomes true, repetitive tasks are terminated. Termination conditions are re-evaluated on each cycle.

After the event is triggered, tasks for the cleanup routine can be processed before the cycle ends or the process is completely terminated.

Here you can see a process where cleanup tasks have been defined.



The extensions presented are intended to help in the modeling of continuous processes by providing templates for the creation of process models. And on the other hand, through the representation as a closed-loop subsystem with its own symbols for state query, regulation and termination events, help to understand such processes more easily. In addition, for a clearer representation of the entire process, subprocesses can also be used for subdivision. Thus, we also fulfill the last feature - comprehensibility of models of continuous processes.

Examples for process models

I will now show you process examples modeled with the extensions presented in our paper. I would like you to take a look at the models and tell me what you can read out of them and whether the models fulfill the necessary information content for modeling the underlying control processes. Beforehand, I will explain to you what kind of process is to be modeled for each example. Please give open feedback on the models in BPMN.

4) Evaluate the following model: simple feedback temperature control for a heat exchanger based on the example from the MathWorks library[1].

The temperature of a liquid in a stirred tank is controlled using a heat exchanger. The heat flow introduced through the heat exchanger is controlled by a valve that controls the steam flow. The disturbing environmental influence to be considered is the fluctuating temperature of the liquid supplied. The tank is assumed to be insulated.

[1] See https://de.mathworks.com/help/control/ug/temperature-control-in-a-heat-exchanger.html, accessed on 05/05/2021

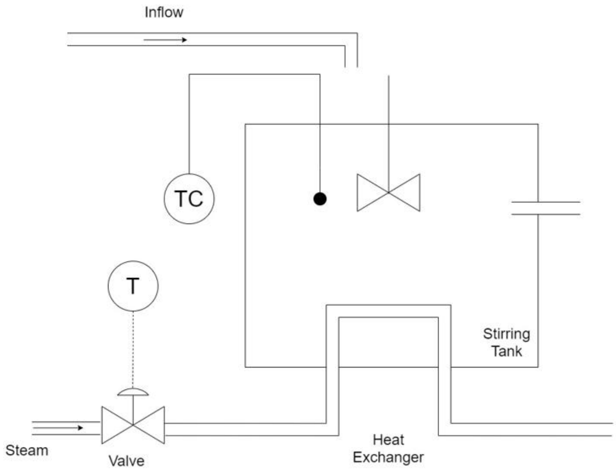


Bild 1: Process flow chart[2]

▽ **Data Elements** | **Endpoints** | **Attributes**

New

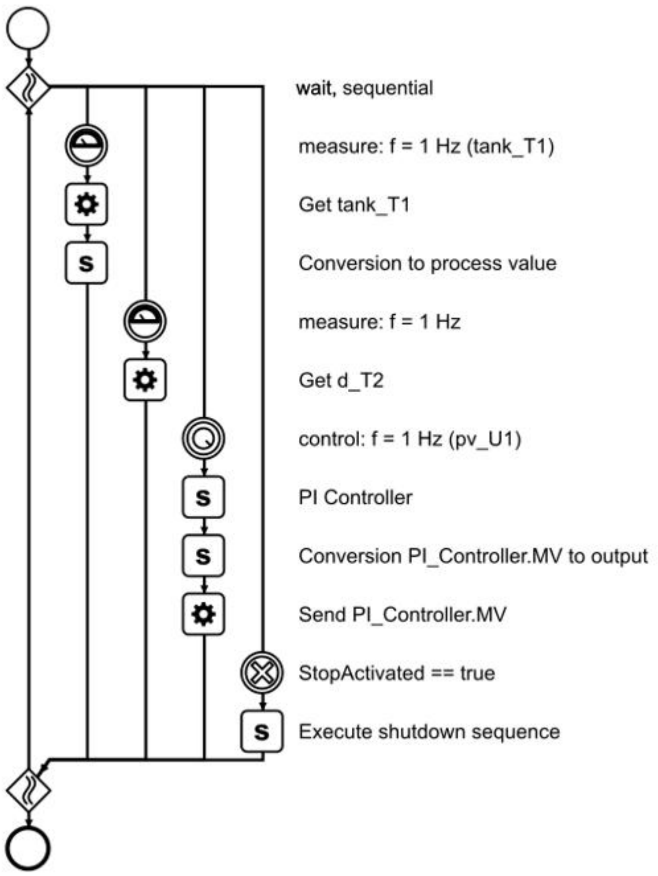
tank_T1	⇒ 0
theta	⇒ 14.7
tau	⇒ 21.3
tauc	⇒ 24.5582
PI_Controller.ER	⇒ 0
PI_Controller.PV	⇒ 0
PI_Controller.SP	⇒ 210
PI_Controller.GAIN	⇒ 1.2341
PI_Controller.TC	⇒ 24.5582
PI_Controller.PV_Out	⇒ 0
PI_Controller.IntER	⇒ 0
PI_Controller.prevER	⇒ 0
d_T2	⇒ 0
PI_Controller.MV	⇒ 0
MV_U1	⇒ 0

[2] See https://de.mathworks.com/help/control/ug/temperature-control-in-a-heat-exchanger.html, accessed on 05/05/2021

Data Elements | Endpoints | Attributes

timeout	⇒ http://gruppe.wst.univie.ac.at/~mangler/services/timeout.php
get_tank_T1	⇒ http-get://127.0.0.1:8080/process1/parameters/tank/t1
post_U1	⇒ http-post://127.0.0.1:8080/process1/parameters/valve/v1
get_d_T2	⇒ http-get://127.0.0.1:8080/process1/parameters/disturbance/t2

Interval duration overrun	⇒ wait
Measure-control cycle execution	⇒ sequential



ID	⇒ a3
Label	⇒ Conversion to process value
Script	⇒ `PI_Controller_PV = tank_T1`

ID ⇒ a1
Label ⇒ PI Controller
Script ⇒

```
PI_Controller.prevER = PI_Controller.ER;
PI_Controller.ER = PI_Controller.PV - PI_Controller.SP;PI_Controller.IntER = PI_Controller.IntER +
PI_Controller.DerER = (PI_Controller.ER - PI_Controller.prevER) * interval_frequency;
PI_Controller.MV = (PI_Controller.GAIN * PI_Controller.ER) + (PI_Controller.GAIN/tauc * PI_Control
```

ID	⇒ a7
Label	⇒ Conversion PI_Controller.MV to output
Script	⇒ `MV_U1 = PI_Controller.MV`

I ask you to rate the model according to the following criteria on a scale from 1 (very bad) to 5 (very good):

Criteria	Question	1 (very bad)	2 (bad)	3 (neither)	4 (good)	5 (very good)
Comprehensibility	What happens?					
Clarity	Can I grasp the entire system at a glance?					
Simplicity	Could the model be presented even more simply?					
Logic	Is it clear what happens in parallel and what happens sequentially?					
Extensibility	Could something be added to the model that would improve the information content?					

5) Rate the following model:

The model is based on the description of the heating process, taken from training documents of the company Siemens[3].

This is also a temperature control for a stirred reactor, the control is realized in this example with a PID controller, a manual control as well as a pulse generator. The heating is not done by a heat exchanger but by a heating element. Furthermore, interlock conditions are defined. The descriptions from the training documents for process modeling with SIMATIC PCS7 were used as a basis for the process model[4]. Our process model is modeled with an automatic control system that breaks out of the closed-loop system when switched to manual control. We assume that the system has already started and is controlled automatically. Furthermore, the process is described for one reactor and not for 2 reactors as in the documentation. PV_In stands accordingly for Pcs7AnIn.PV_In for the temperature control.

[3] See https://www.automation.siemens.com/sce-static/learning-training-documents/pcs7/v9-0/p01-06-control-loop-v9-tud-0719-de.pdf, accessed on 05/05/2021

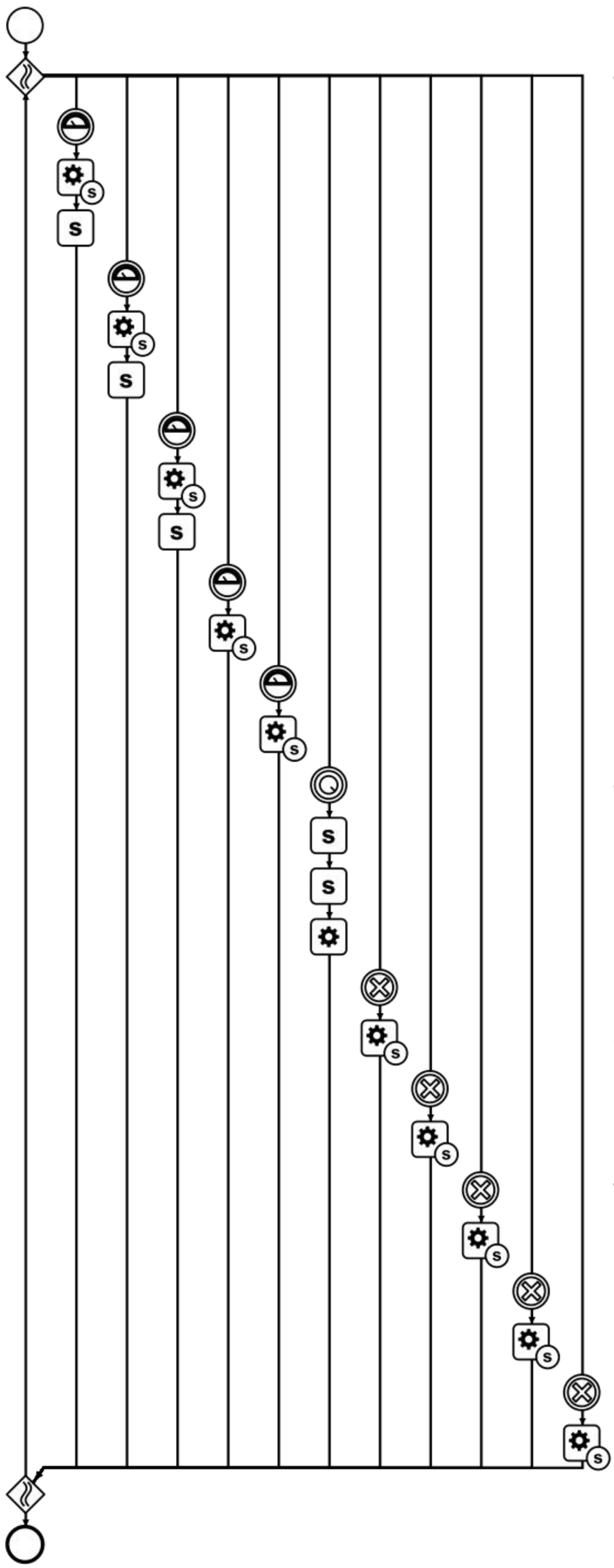
[4] See https://www.automation.siemens.com/sce-static/learning-training-documents/pcs7/v9-0/p01-06-control-loop-v9-tud-0719-de.pdf, accessed on 05/05/2021

▽ **Data Elements** | **Endpoints** | **Attributes**

New

T_Reactor_In	⇒ 0
T_Reactor_Max	⇒ 60.0
Level_Reactor_In	⇒ 0
Level_Reactor_Min	⇒ 200.0
PID_Controller.GAIN	⇒ 1.0
PID_Controller.TI	⇒ 100.0
PID_Controller.TD	⇒ 0.0
PID_Controller.DiffGAIN	⇒ 1.0
PID_Controller.ER	⇒ 0.0
PID_Controller.PV	⇒ 0.0
PID_Controller.SP	⇒ 50.0
PV_Out	⇒ 0.0
PV_In	⇒ 0.0
OperationMode	⇒ Automatic
MainSwitch	⇒ On
PID_Controller.prevER	⇒ 0
PID_Controller.IntER	⇒ 0
PID_Controller.DerER	⇒ 0
PID_Controller.MV	⇒ 0

Interval duration overrun	⇒ cancel
Measure-control cycle execution	⇒ sequential


cancel, sequential
measure: f = 0.1 Hz (T_Reactor_In)
Get T_Reactor_In
Data conversion
measure: f = 0.1 Hz (Level_Reactor_In)
Get Level_Reactor_In
Data conversion
measure: f = 0.1 Hz (OperationMode)
Get OperationMode
Data conversion
measure: f = 0.1 Hz (EmergencyStop)
Get EmergencyStop
measure: f = 0.1 Hz (MainSwitch)
Get MainSwitch
control: f = 0.1 Hz (PV_In)
PID Controller
PWM
Send PV_Out
MainSwitch == Off
Send message to operator
EmergencyStop == true
Send message to operator
T_Reactor > T_Reactor_Max
Switch to safe shutdown routine
Level_Reactor < Level_Reactor_Min
Switch to safe shutdown routine
OperationMode == Manual
Call manual routine

ID ⇒ a7

Label ⇒ PID Controller

Script ⇒
```
PID_Controller.prevER = PID_Controller.ER;
PID_Controller.ER = PID_Controller.PV - PID_Controller.SP;
PID_Controller.IntER = PID_Controller.IntER + (PID_Controller.ER * 1/interval_frequency);
PID_Controller.DerER = (PID_Controller.ER - PID_Controller.prevER) * interval_frequency;
PID_Controller.MV = PID_Controller.GAIN * (PID_Controller.ER + (PID_Controller.IntER / PID_Controll
```

I ask you to rate the model according to the following criteria on a scale from 1 (very bad) to 5 (very good):

Criteria	Question	1 (very bad)	2 (bad)	3 (neither)	4 (good)	5 (very good)
Comprehensibility	What happens?					
Clarity	Can I grasp the entire system at a glance?					
Simplicity	Could the model be presented even more simply?					
Logic	Is it clear what happens in parallel and what happens sequentially?					
Extensibility	Could something be added to the model that would improve the information content?					

6) Based on these extensions, would you be willing to implement this modeling method in your everyday work when developing a model of a continuous process?

7) How well do you think these extensions describe a control system for these examples?

Features	1 (very bad)	2 (bad)	3 (neither)	4 (good)	5 (very good)
Model 1 (Question 3)					
Model 2 (Question 4)					

Please explain your answer.

8) What is missing for a detailed process description?

9) If you have experience in control engineering - what could you recommend as further improvements for the extensions, in order to make them more attractive for engineers?

10) Finally, I would like to go into a little more detail about the models with our extensions and ask you to rate them again - on a scale of 1 to 5, with 5 being the best value.

Question	1 (very bad)	2 (bad)	3 (neither)	4 (good)	5 (very good)
How easy is it to understand in the models shown that the individual processes run in parallel and independently of each other?					
How easy is it to define when an adjustment/regulation is made to the system?					
How easy is it to define the max. duration of an adjustment?					
How easy is it to define, under which conditions all repetitive tasks shall end?					
How easy is it to define that cleanup tasks have to be done once afterwards?					
How easy is it to describe complex operations in the context of continuous processes with these extensions?					

Thank you very much for taking the time to do this!
I appreciate feedback on the interview!

What do you think about the questions? Were they easy to understand? Were they too long? Too closed? Was the length of the interview uncomfortable?