Aspect-oriented Data and Safety Modeling for Cyber-Physical Systems in Process Automation

Dirk Kuschnerus
Institute of Electronic Circuits
Ruhr-Universität Bochum
Bochum, Germany
Email: dirk.kuschnerus@rub.de

Attila Bilgic
KROHNE Messtechnik GmbH
Duisburg, Germany
Email: a.bilgic@krohne.com

Thomas Musch
Institute of Electronic Circuits
Ruhr-Universität Bochum
Bochum, Germany
Email: thomas.musch@est.rub.de

***Abstract*—Cyber-physical systems (CPS) integrate computation with physical processes, enabling the dynamic adaption of systems based on economic and environmental conditions. The adoption of CPS in industrial process automation is impeded by legacy systems with severe functional safety constraints and the need for highly configurable devices. To transfer the benefits of CPS to process automation, the inherent conflict between CPS safety and configurability must be explicitly considered during system design and operation. This paper proposes aspect-oriented modeling of safety and data for CPS in process automation as a baseline for formal consistency monitoring.**

I. Introduction

Cyber-physical Systems (CPS) integrate computation and physical processes [1]. A remaining key research challenge for CPS is the definition of suitable modeling approaches covering abstraction and functional safety [2], [3]. Devices used in process automation CPS are usually mass market products which must be adapted to the specific process environment of the customer. Suppliers implement this requirement by adding configurability to their products, tailoring them to the customer's requirements during production and on-site. This drastically increases device complexity, an effect which is amplified by the emergent requirements for highly dynamic systems arising from initiatives like the German "Industry 4.0", where factory automation shall enable the dynamic adaption of the production to customer's desires or real-time demand and economical factors along the life cycle of a product [4].
Unlike in factory automation, systems in process automation are typically large scale plants with complex static control loops, where despite its benefits CPS adoption is limited by functional safety restrictions imposed by standards such as IEC 61508 [5] due to risks for humans and environment. Safety-critical applications require extended verification of the plant's safety functions, which is impeded when the implementing CPS components are configurable, creating a dependence between safety-critical behavior and configuration data. This conflict raises the need for formal specification and verification techniques to enable safety verification of devices considering specific configurations. In [6], we introduced a domain model for hierarchical modeling of CPS in process automation. In this paper, we propose a modeling approach advancing our domain model, using the aspect-oriented paradigm to explicitly model the safety and data concerns separated from the CPS domain model. Our goal is to enable the reasoning about safety in presence of complex configurability, setting a baseline for the application of dynamic CPS in process automation while preserving the required functional safety of the legacy installations.
The remainder of this paper is organized as follows: Section II examines related work on CPS modeling, Sections III and IV describe the interconnected modeling of the safety and data aspects. Both aspects are jointly examined in a case study in Section V, which also covers inconsistencies between aspect and domain models. Section VI gives a conclusion and outlines further research directions.

II. Related Work

To obtain an overview of methods for modeling CPS in process automation, we conducted a systematic mapping following the method introduced in [7]. The mapping identified and categorized a total of 448 relevant publications, 15 of which target the domain of process automation. This low number is also confirmed by [8], where only one publication from the domain of industrial automation is listed.
Following our mapping, we further examined the relevant papers from all domains in a systematic literature review as suggested in [9], focusing on the aspects of data and functional safety. From the 15 process automation papers only [10], [11] and [12] address functional safety. A combination of safety and data modeling was only found in [12], where the reachability of unsafe hybrid parametrized automata states is determined but further safety and data concepts such as hierarchical modeling and data dependencies are not covered. To additionally cover cross-domain and generic approaches, we broadened our investigation to relevant approaches from all domains and identified 5 further papers covering data and safety aspects in CPS ([13], [14], [15], [16], [17]). [14] describes architectural views for heterogeneous CPS models and consistency considerations between these views as well as behavioral semantics for system verification. The approach uses automata states to model and verify the safety of the CPS. [16] develops a formal framework and graphical notation for the development of hybrid systems using graphs and hybrid automata. Neither [13], [14], [15] nor [16] include the modeling of safety concepts that are incorporated in the design of the whole CPS on various abstraction levels or the in-depth coverage of configuration and the effects of configuration data on system behavior and safety. [17] does not cover physical and deployment views of CPS which we consider important for safety and data consistency. In addition, none of the publications mentioned before use aspect-oriented modeling

for the cross-cutting concerns of safety and data.
Concluding our literature review, to the best of our knowledge our approach differentiates from the state of the art by covering the explicit aspect-oriented modeling of functional safety and data for CPS in process automation. We are further not aware of generic or cross-domain approaches providing a comprehensive modeling coverage of safety and data concepts which we consider crucial for CPS in process automation.

III. Aspect-oriented Safety Modeling

In aspect-oriented development, cross-cutting concerns affecting major parts of the development artifacts are represented as detached aspects to achieve separation between functionality and cross-cutting concerns [18]. During the definition of our domain model, we identified the influence of functional safety on its *structural*, *behavioral*, *physical*, *deployment* and *communication* viewpoints on the *plant*, *CPS*, *subsystem* and *component* layers of abstraction. In addition to their distributed influence, safety concepts typically are major development artifacts which regarding to their specification and certification efforts are desired to be reused in multiple projects. We therefore follow the aspect-oriented paradigm by integrating these concerns into safety aspect models abstracted from a specific device development and connecting them to our domain model using formal weaving functions.
Fig. 1 shows the scope of our safety aspect. Aspect models

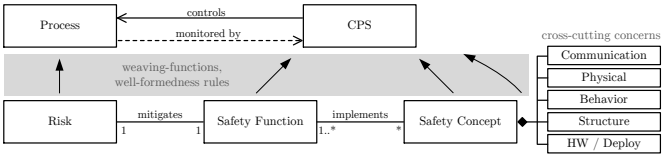

Fig. 1. Scope of the safety aspect

define cross-cutting safety concerns from all viewpoints and abstraction layers. Multiple concerns are integrated to safety concepts implementing safety functions, which are defined in IEC 61508 [5] as functions controlling a process or system to mitigate the risk of dangerous failure causing harm to humans and environment. The connection points between aspect models and the domain model are formally defined by weaving functions and structurally constrained by well-formedness rules (*wfrs*). The following paragraphs give an overview of the aspect beginning with the cross-cutting concerns.
The influence of functional safety on the logical CPS structure is primarily specified by the criticality and safety integrity level (*SIL*) of structural model elements, needed as baseline for further approaches such as partitioning and deployment. The function *crit* defines whether an element is safety-related (*sr*), non-safety related (*nsr*) or has a mixed criticality:

$$\begin{aligned} \textbf{\textit{crit}} :& Entity \cup DeployContext \cup Channel \cup Protocol \\ & \rightarrow \{sr, nsr, mixed\} \end{aligned} \tag{1}$$

The function *sil* defines a SIL for each element used to determine its valid dependencies regarding safety. To connect this safety information to our domain model, we define the weaving-function *addAttribute*:

$$\begin{aligned} \textbf{\textit{addAttribute}} =& \{f_{aA_1}, ..., f_{aA_n}\} \\ f_{aA_i} =& E' \subseteq Entities \rightarrow type(att_i \in Attributes) \end{aligned} \tag{2}$$

specifying a function f_{aA_i} for each attribute A_i that shall be added to the domain model. f_{aA_i} assigns the type of A_i to each of the targeted Entities E'. f_{aA_i} is by convention named after the added attribute A_i. The application of every weaving function is constrained by associated structural wfrs.
IEC 61508 differentiates between systematic failures and random hardware failures e.g. due to hardware aging. The latter are included in safety analysis such as Failure Modes Effects and Diagnostic Analysis (FMEDA) to determine the risk of dangerous undetected failures of the safety function. We add hardware failure information to the hardware elements in the domain model using the weaving function $addAttribute$, covering the probability of a hardware error, the fractions of safe, unsafe, detected and undetected errors and aggregated values for system integrators.
The dependable transfer of safety-related data must be ensured by safe communication channels in safety-critical CPS. The applicability of a channel for transporting data of a specific SIL is influenced by the bit error probability of the underlying hardware link as well as qualitative and quantitative communication measures modeled by the safety aspect both in the *deployment* and *communication* viewpoints. Safety-critical CPS typically adapt their behavior according

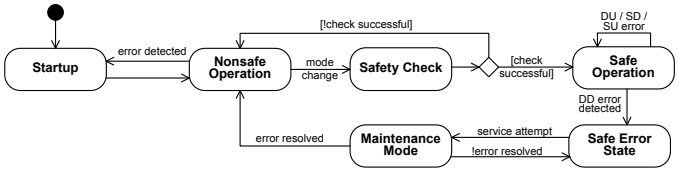

Fig. 2. Safety modes introduced by the safety aspect

to their internal safety mode. To model the overlaying safety modes of the CPS under development and the modification of the domain model behavior during the modes, our aspect uses a safety automaton connected to the behavioral semantics of the domain model via weaving functions. Behavioral changes triggered by the safety mode are the augmentation of the behavioral semantics, e.g. the addition of an additional safety check, and their restriction, e.g. due to a simulation function which may only be started in nonsafe operation. Both augmentation and restriction can be applied to all behavioral models used in [6]. Fig. 2 shows the generic safety automaton, describing the internal safety mode of the CPS from startup to nonsafe and safe operation. The CPS stays in safe operation in case of dangerous undetected and safe errors, switching to a safe state on detection of critical errors. The augmentation of behavioral semantics is used in the motivated verification of process safety functions considering their configurable realization. On the subsystem layer, safety functions are modeled by adding control modes to the hybrid automaton of the process step. The process safety time, i.e. the time between the occurrence of a failure and a resulting hazardous event defines the detection and reaction time available to the safety functions for mitigating the risk of the event and is modeled inside the safety automaton of the specific CPS. The realization of safety functions is modeled on the component layer by modifying the system's activity diagrams. Both concepts are detailed in section V-D.
As shown in Fig. 1, our safety aspect additionally defines reusable safety concepts from sets of cross-cutting concerns. These templates can e.g. define diagnostic functions to detect the hardware failures introduced above, facilitating the tracing

between failure analysis and mitigation as shown in the 2-channel safety concept in Fig. 11.

IV. Aspect-oriented Data Modeling

The device development for industrial CPS targets mass markets where it is unfeasible to develop individual products for every customer. As a result, complementing the measurement data the CPS gathers and processes, industrial CPS contain a large number of configuration data used for the customization of base device variants to the customer's application. Both data categories influence all viewpoints over the entire system hierarchy. A configuration data model is often used detached from a specific device, e.g. during manufacturing and order processes or customer service and should be separated from other development artifacts. We therefore model CPS data as aspect to enable data development, deployment and analysis to be conducted independently from the domain model.

Fig. 3 shows the fundamental data aspect model for CPS in process automation, categorizing data items into device variables and static data. Device variables represent values connected to the physical process which are periodically refreshed and distributed throughout the CPS, whereas static data items describe typically persistent attributes of the CPS itself. Every data item is uniquely identified by its *localId* and scope, which defines the area where device variables are distributed and static data is synchronized and available to all modules. This can be e.g. a partition or a node. Device variables are produced by one distinct software module (*producer*), can be set to a synthetic test value (*synthetic*) and connected to an external CPS interface (*outputChannel*). Each device variable has an update interval and can be kept local by stopping the bus distribution. The data aspect defines two types of device variables: a float value with timestamp and a double value. Static data is protected by an access level specifying valid editors as well as the item's settings for persistent storage and replication between data stores. Static data is defined for several data types and contains actual, default and SIL values and associated value ranges.

The data aspect also defines dependencies between data items. Configuration details of device variables such as limits and default values are given by a referenced static data item (*configuredBy)*. Both device variables and static data can be derived from other data items. Derived data items are locally calculated from primary data items and are not persistently stored. In addition, static data items can interfere with each other during the update of their values. An *updateDependency* between multiple data items denotes that the CPS can only accept an updated value of a data item if the dependent items are also updated. The *checkDependency* is an internal dependency that triggers a validation of connected data items if the value of one data item is altered.

While the safety and data aspects are specified concurrently to the domain model, there is interdependence between the aspect models. Device variables and static data both have a criticality and SIL defined via the weaving function *addAttribute* from the safety aspect. In addition, the SIL value in static data items defines a value that is set when the safe operation mode specified by the safety aspect is entered.

The aspect model is connected to the domain model using

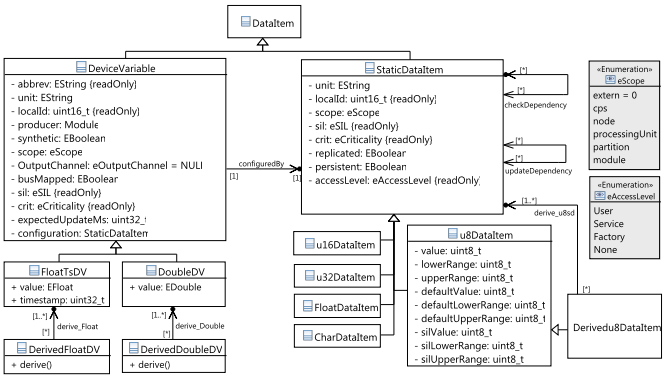

Fig. 3. Basic data model defined by the data aspect

the concepts of weaving functions and restrictions specified by wfrs as described in Section III. In the structural, communication and deployment views, attributes can be altered and entities such as software modules can be deactivated by the data aspect. As the data aspect models the CPS internal representation of process values, it adds a connection between process values and device variables thus connecting physical and computational parts of the domain model. In the deployment and communication views, the storage and distribution data flow of device variables and static data is modeled using the weaving function *addData*. This function also specifies hardware configuration by connecting static data items to hardware nodes. As described in Section III, the aspect models influence the behavior of the CPS. The influence of configuration data to the behavioral semantics is modeled using the weaving functions *addBehavior* and *removeBehavior* as shown in Fig. 9.

V. Case Study

A. Example Process - Distillation in MDI Production

We use the simplified chemical process of methylene diphenyl diisocyanate (MDI) production [19] as case study for exemplary application and evaluation of our modeling approach. MDI as a base product of polyurethanes is one of the most produced isocyanates. It can be generated by condensation of aniline and formaldehyde to methylenedianiline (MDA) using hydrochloric acid (HCl) as catalyst followed by phosgenation of the MDA. Using the highly dangerous phosgene, the process is well-suited to study our domain model in a safety-critical environment. As shown in Fig. 4, phosgene and HCl must be separated from the crude MDI after the phosgenation in a distinct subprocess "phosgene separation", on which we focus in our case study.

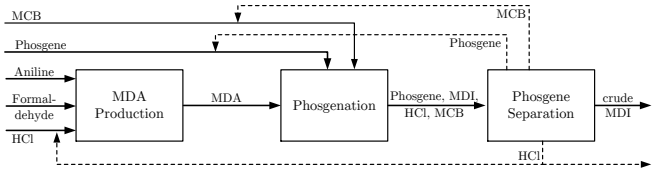

Fig. 4. Schematic Overview of MDI Production

The separation of phosgene, HCl, MDI and the solvent chlorobenzene (MCB) can be realized by distillation, as schematically shown in Fig. 5. The feed mixture *F* enters the column in liquid state and flows towards the base of the column which is continuously heated by circulation of the base product (MCB and MDI) through a steam reboiler. This leads to vaporization of phosgene, HCl and parts of the MCB, forming a vapor flow *V* towards the head of the column. This head product exits the column and enters a condenser, which cools down the vapor to liquefy MCB and phosgene while extracting the gaseous HCl. MCB and phosgene are then stored in an output tank and directed towards other subprocesses. In a non-ideal distillation column, the vapor emerging at the column bottom contains both the lighter and heavier compound. To reduce this mixing and to obtain higher purity of base and head product, column floors and a partial reflux *L* from the output tank to the column are installed. When the steam meets with the reflux at a certain floor, it partly condenses whereat mostly the heavier compound is liquefied. The condensing energy contrarily causes the vaporization of the lighter compound in the reflux. Moving towards the column head, this effect is multiplied by introducing additional floors to reach a desired level of purity. Typical distillation columns are controlled using five control

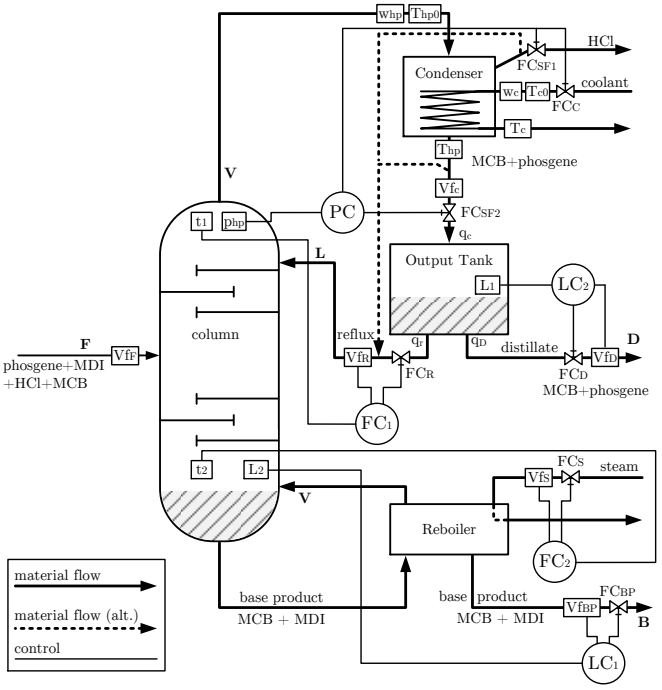

Fig. 5. Schematic Overview of Phosgene Separation by Distillation

loops. To obtain stable column operation, the levels at the column base (LC_1) and the output tank (LC_2) as well as the column pressure (PC) are controlled. The product quality is defined by the desired separation between the compounds in both base and head product (distillate). Control schemes for distillation columns are commonly named after the variables used to control the product quality. In our case study, we apply the LV control scheme, which controls the head product composition by the reflux (FC_1 controlling *L*) and the base product composition by the flow of steam to the reboiler (FC_2 controlling *V*). To reduce the size of our case study, we evaluate our aspect-oriented modeling approach focusing on condenser and output tank of the distillation. In the following section, we introduce both process steps and derive a state space representation to which we apply our approach.

B. State Space Representation

Control algorithms in process automation are designed based on system models either deduced from physical characteristics of the controlled process or aggregated from measurements. The variables and detail of the model must be adapted to the actual control task. In our case study, the safety-relevant criterion of the condenser is the complete liquefaction of phosgene, warranted if the condenser cools the head product lower than the boiling point of phosgene at 7.44 degrees Celsius. The controlled variable in the condenser is the mass flow w_c of the coolant. We abstract the behavior of the condenser with that of a heat exchanger:

$$\begin{aligned} \dot{T_{hp}} &= \frac{w_{hp}}{m_{hp}} T_{hp0} - \frac{w_{hp}}{m_{hp}} T_{hp}(t) - \frac{kAT_{hp}(t)}{m_{hp}c_{hp}} + \frac{kAT_c(t)}{m_{hp}c_{hp}} \\ \dot{T_c} &= \frac{w_c}{m_c} T_{c0} - \frac{w_c}{m_c} T_c(t) + \frac{kAT_{hp}(t)}{m_c c_c} - \frac{kAT_c(t)}{m_c c_c} \end{aligned} \tag{3}$$

where $\dot{T_{hp}}$ and $\dot{T_c}$ are the time-derivatives of the head product and coolant temperatures after condensing, k is a constant of the heat exchanger describing its ability to transfer energy, A is the area of the heat exchanger, m_{hp} and m_c are the masses, c_{hp} and c_c the heat capacities, w_{hp} and w_c the mass flows and T_{hp0} and T_{c0} the temperatures at the entry point of the condenser of the head product and coolant, respectively.
The change in head product temperature depends on the heat transfer between product and coolant as well as on their initial temperatures. For controlling the head product temperature by manipulating the mass flow of the coolant, we rewrite the second equation for $T_c(t)$ and insert it into the first to obtain a non-linear equation for the dependency of $\dot{T_{hp}}$ and w_c. To facilitate control and verification activities we generate a linear state space representation of the form

$$\begin{aligned} \Delta\dot{x_t} &= A\Delta x(t) + B\Delta u(t) + E\Delta d(t) \\ \Delta y(t) &= C\Delta x(t) + D\Delta u(t) \end{aligned} \tag{4}$$

where $\Delta x(t) = \begin{pmatrix} \Delta T_{hp}(t) \\ \Delta T_c(t) \end{pmatrix}$ is the state, $\Delta u(t) = \Delta w_c(t)$ the input, $\Delta d(t) = \begin{pmatrix} \Delta w_{hp}(t) \\ \Delta T_{hp0}(t) \end{pmatrix}$ are the disturbance variables and $\Delta y(t)$ is the output of the condenser. Δ symbolizes small deviations from the original variables in the cause of linearization. The matrices A, B, C, D and E are obtained by Taylor linearization of the equations in (3) at a specific operation point in which T_{hp} and T_c are stable, i.e. $\dot{T_{hp}} = 0$ and $\dot{T_c} = 0$:

$$\begin{aligned} A &= \begin{pmatrix} 0 & -\frac{w_{hpR}c_c w_{cR}}{kA} - w_{hpR} - \frac{c_c w_{cR}}{c_{hp}} - \frac{kA}{c_{hp}} \\ \frac{kA}{m_c c_c} & -\frac{w_{cR}}{m_c} - \frac{kA}{m_{hp}chp} \end{pmatrix} \\ B &= \begin{pmatrix} -\frac{w_{hpR}c_c T_{cR}}{kA} + \frac{w_{hpR}c_c T_{c0}}{kA} - \frac{c_c T_{cR}}{c_{hp}} + \frac{c_c T_{c0}}{c_{hp}} \\ T_{c0} - T_{cR} \end{pmatrix} \\ C &= \begin{pmatrix} 1 & 0 \end{pmatrix} \\ D &= 0 \\ E &= \begin{pmatrix} T_{hp0R} - \frac{c_c w_{cR} T_{cR}}{kA} - T_{cR} + \frac{c_c w_{cR} T_{c0}}{kA} & \frac{w_{hpR}}{m_{hp}} \\ 0 & 0 \end{pmatrix} \end{aligned} \tag{5}$$

Variables with the index *R* are fixed in the operation point. While more complex models for condensing processes exist (see e.g. [20]), we use this abstracted form to keep the size of the verification problem manageable for our case study.
The output tank is the second process step covered by our case study. The critical process variable concerning functional safety is the level of the distillate $h(t)$ inside the tank which is controlled by LC_2 via the valve FC_D. The level is influenced by the input q_c from the condenser, the reflux q_r to the column and the output flow q_D:

$$\begin{aligned} q_c &= Vf_c * d_c(t) \\ q_r &= k_r * d_r(t)\sqrt{2gh(t)} \\ q_D &= k_D * u(t)\sqrt{2gh(t)} \end{aligned} \tag{6}$$

The flows are defined by the opening of the corresponding valve ($d_c(t), d_r(t), u(t) \in [0..1]$) and the constants k_r, Vf_c, k_D defining the maximum flow at the entry and exit points of the flow. For the level control, $d_c(t)$ and $d_r(t)$ are disturbances and the input $u(t)$ is located at FC_D. q_r and q_D are dependent on the current level h(t) according to Torricelli's law. The time-derivative $\dot{h}$ is given by the non-linear equation

$$\dot{h} = \frac{q_c - q_r - q_D}{A_T} \tag{7}$$

which can be linearized to a linear state space model in the form of (4) where $\Delta x(t) = \Delta y(t) = \Delta h(t)$ and

$$\begin{aligned} A &= \left(\tfrac{1}{A_T}\sqrt{2g}\tfrac{1}{2\sqrt{h_R}}(k_r d_{rR} + k_D u_R)\right) \\ B &= \left(-\tfrac{1}{A_T}k_D\sqrt{2g}\sqrt{h_R}\right) \\ C &= 1 \\ D &= 0 \\ E &= \begin{pmatrix}\frac{Vf_c}{A_T} & -\frac{1}{A_T}k_{rR}\sqrt{2g}\sqrt{h_R}\end{pmatrix} \\ \sqrt{h_R} &= \frac{-Vf_c d_{cR}}{(-k_r d_{rR} - k_D u_R)\sqrt{2g}} \end{aligned} \tag{8}$$

C. Modeling the CPS

Focusing on the condenser and output tank of the phosgene separation, we first model the control systems *PC* and LC_2 from Fig. 5 excluding data and functional safety. Both span over the subsystem and component layers of abstraction. The connection between the continuous state space models of condenser and tank introduced in section V-B and the behavior of their control systems is established on the subsystem layer using hybrid automata as defined in [21].
The states V of a hybrid automaton
$H = \langle X, V, E, init, inv, flow, jump, event, \Sigma \rangle$ represent control modes defining the continuous flow of real number variables X, while the transitions E specify discrete mode switches. The labeling functions *init, inv, flow* and *jump* assign predicates to each control mode v_i, where *init* assigns initial values to the variables of v_i, *inv* defines invariants for the variables, *flow* specifies the continuous change of variable values within a control mode and *jump* defines conditions for control switches. The function *event* assigns a triggering event from the set Σ of events to each control switch. In our domain model, the automata of a subprocess can exchange signals and change their control modes by the discrete event of receiving a signal and by the violation of a state invariant.
Figure 6 shows the hybrid automaton of the condenser, whose *control* state defines the control algorithm of the condenser in

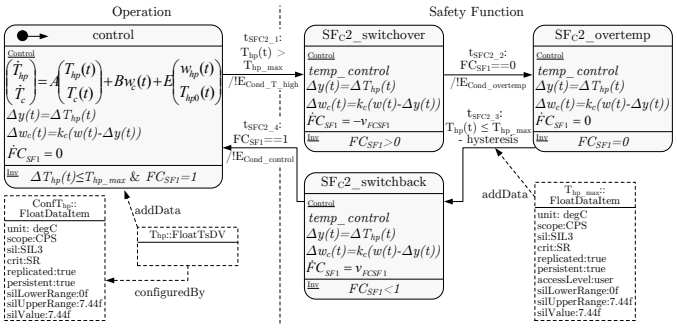

Fig. 6. Behavior of the Condenser Control System on Subsystem Level

safe operation mode, where the mass flow of the coolant w_c is controlled to keep the temperature T_{hp} below T_{hp_max}. The control system in our case study uses a proportional controller $\Delta w_c(t) = k_c(w(t) - \Delta y(t))$ which multiplies the comparison of reference variable $w(t)$ and output $\Delta y(t)$ by a factor k_c. The invariants of the control state define that during regular control, the HCl output valve FC_{FS1} is constantly open and the temperature ΔT_{hp} is always below the threshold T_{hp_max} of 7.44 degrees Celsius. Similar to the condenser, the behavior of the output tank on subsystem level is defined as hybrid automaton shown in Fig. 7. During regular operation, the automaton controls the level of the output tank via the valve FC_D, as defined in (8).
In our domain model, the component layer as lowest layer of abstraction specifies the implementation of the control algorithms defined on the subsystem layer. Fig. 8 shows the deployment structure of the condenser and output tank. The condenser consists of five nodes which are containers for groups of hardware units connected by communication links and typically contained in a separate housing. Three of the nodes are vortex sensors which count the vortexes in the Kármán vortex street behind a bluff body in the pipe to measure the flow speed. The structural model of a vortex node in our domain model consists of a sensor (e.g. piezo elements or a physical switch) for vortex counting, pressure and temperature sensors for mass flow calculation, a vortex algorithm software calculating the mass flow from the vortex frequency f as well as software components for inter-node data management and communication. *Condenser_Vortex1* and *Condenser_Vortex2* measure the mass flow w_{hp} and the temperature T_{hp0}, while *Condenser_Vortex3* measures the incoming mass flow w_c and temperature T_{c0} of the coolant. The node *Condenser_Temp1* contains two temperature sensors measuring the temperature T_{hp} of the distillate. In addition to the three sensor nodes, the converter includes a node with two microcontroller units (*MCUs*) running the software components for the temperature control algorithm and the actual valve control. The converter controls the valve nodes *FCC*, *FCSF1* and *FCSF2*.
The output tank in our case study consists of three nodes: two radar level sensor nodes *Tank_Radar1* and *Tank_Radar2* as well as the *Tank_converter* node which runs the level control algorithm and controls the valve FC_D. The nodes of the converter and the tank are connected using a bus system while the connections between valves and control nodes are realized via point to point links.
On the component level the behavior of each control mode from the hybrid automaton on the subsystem level is modeled

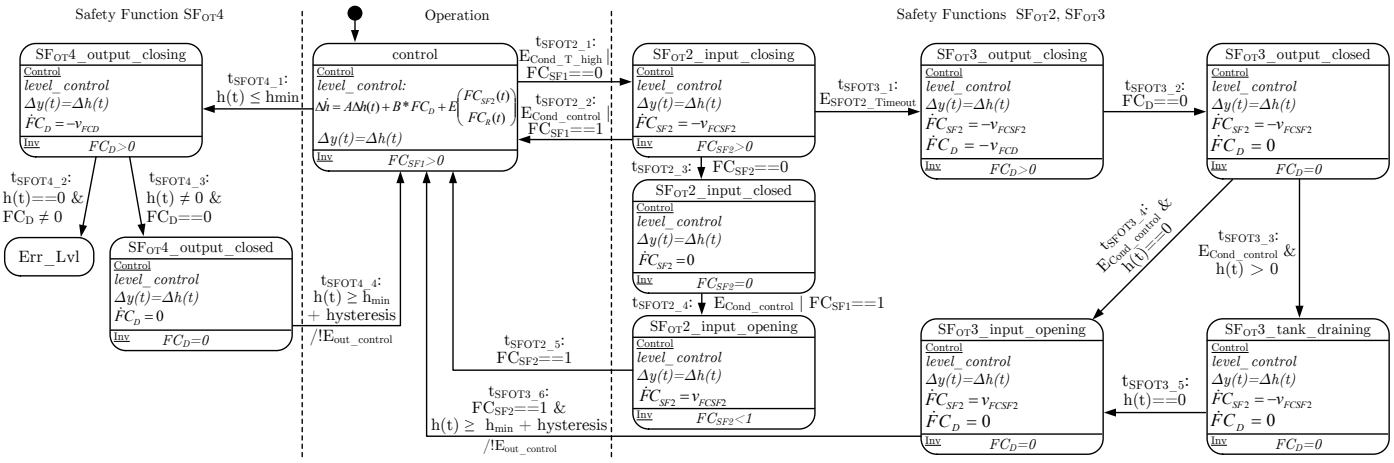

Fig. 7. Behavior of the Output Tank Control System on Subsystem Level

as activity diagram defining the control and data flows between the control system entities in this specific mode. Each of the activities carried out by an entity can additionally be detailed by a distinct activity diagram. Fig. 9 partly shows the activity diagram for the control mode of the condenser. The grey actions mark the standard behavior of the control system while the dotted flows and white activities are weaved from the safety aspect as described in section V-D. The mass flow w_{hp1} is calculated from the vortex frequency f_{hp1} and the temperature T_{hp0_1} and transferred via the bus using the action *transferSR* denoting safety-critical communication. Together with the current temperature value T_{hp}, the temperature control calculates a control input $controlFC_C$ which is adjusted in the *ValveControl* module of the valve FC_C.

D. Modeling Safety Aspects

Complementing the system specification, we use safety aspect models to define the safety functions of condenser and tank on the subsystem layer of abstraction as well as their realization on the component layer. To demonstrate the specification of hardware failures and their tracing to detection methods, we apply the safety concept of 2-channel redundancy by structural and behavioral weaving to the condenser domain model.

The safety functions of condenser and output tank are closely coupled. SF_C1 in the condenser ensures that the HCl which is used in other processes or even sold to contractors is not contaminated with phosgene. To prevent this, PC continuously controls pressure and temperature T_{hp} by manipulating the incoming mass flow w_c of the coolant. A demand on the safety function SF_C2 occurs when T_{hp} rises above 7.44 degrees Celsius due to insufficient cooling. In this case, the safety function assumes that phosgene has not been completely liquefied and is flowing as gas towards the valve FC_{SF1}. SF_C2 must close the valve before phosgene exits. The resulting effect is a mixture of HCL and phosgene leaving the condenser towards the output tank.

The output tank has four distinct safety functions. $SF_{OT}1$ continuously controls the tank level via the valve FC_D. $SF_{OT}2$ reacts on the flow of HCl towards the output tank evoked by the triggering of SF_C2 by closing the valve FC_{SF2} before HCl can enter the tank, directing the complete condenser output back to the distillation column as reflux. $SF_{OT}3$ closes the input and output valves FC_{SF2} and FC_D in case $SF_{OT}2$ is not successful in preventing the entry of HCl into the output tank. $SF_{OT}3$ is responsible for emptying the tank via the reflux to ensure that no HCl is put out to following subprocesses. A demand on $SF_{OT}4$ is triggered when the tank level falls below a minimum threshold. $SF_{OT}4$ is responsible for ensuring that the output valve is never opened when the tank is completely empty to prevent damage to following subprocesses caused by empty pipes. Table I lists the safety functions covered by our case study.

TABLE I. SAFETY FUNCTIONS OF CONDENSER AND OUTPUT TANK

	Process Step	Mode	Control	Description
SF_C1	condenser	continuous	PC	temperature control
SF_C2	condenser	on demand	PC	prevent phosgene output
$SF_{OT}1$	tank	continuous	LC_2	level control
$SF_{OT}2$	tank	on demand	LC_2, PC	prevent HCl output
$SF_{OT}3$	tank	on demand	LC_2	prevent HCl output
$SF_{OT}4$	tank	on demand	LC_2	prevent empty pipe

As introduced in section III, our domain model extends the behavioral specification of the system models using the weaving functions addBehavior and removeBehavior. Fig. 6 shows the behavioral extension of the condenser automaton by the behavior of the safety functions from Table I. The safety function SF_C1 is continuously executed during safe operation in the control state of the automaton without altering the behavior. SF_C2 is executed on demand, i.e. when a dangerous event occurs that may lead to process risks which have to be mitigated. We use the weaving function *addBehavior* to add three additional states to the automaton that are executed by SF_C2. The demand for SF_C2 is modeled by the transition t_{SFC2_1} which transfers the automaton to the state *switchover* when the temperature $T_{hp}(t)$ rises above T_{hp_max} and broadcasts the signal $E_{Cond\_T\_high}$. In this state, SF_C2 closes the valve FC_{SF1} at maximum speed v_{FCSF1} which is denoted by the flow equation $\dot{FC}_{SF1} = -v_{FCSF1}$. When the valve is completely closed, SF_C2 enters the state *overtemp* in which the valve FC_{SF1} remains closed ($\dot{FC}_{SF1} = 0$) so that gaseous phosgene cannot be released and the total output of the condenser flows towards the tank. When the temperature $T_{hp}(t)$ falls below T_{hp_max} including a hysteresis preventing

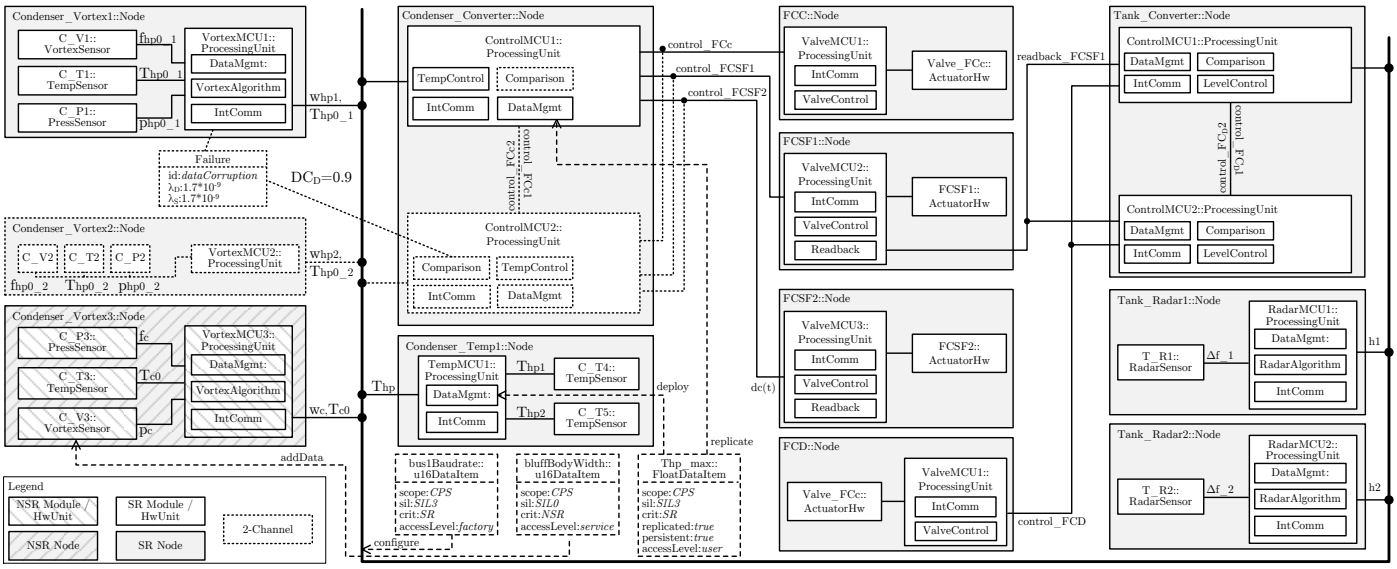

Fig. 8. Deployment Diagram of the Condenser Control System

the rapid switching of control modes, transition t_{SFC2_3} fires and the valve FC_{SF1} is opened again in the state *switchback* to finally return to the behavior of the control state when the valve is completely opened.

The same mechanism is used to attach the safety functions of the output tank to the hybrid automaton in Fig. 7. $SF_{OT}1$ is continuously processed in the control state of the hybrid automaton. The safety functions $SF_{OT}2$ and $SF_{OT}3$ are chained. $SF_{OT}2$ is entered when the valve FC_{SF1} is completely closed or the signal $E_{Cond\_T\_high}$ is received which announces the triggering of SF_C1. In both cases, the tank control closes the input valve ($\dot{FC}_{SF2} = -v_{FCSF2}$) to prevent the inflow of HCl. The safety time for $SF_{OT}2$ is defined by the time the HCl needs to flow from the condenser to the valve FC_{SF2}. If FC_{SF2} closes before the safety time exceeds, $SF_{OT}2$ returns to the control state via t_{SFOT2_3}. If the closing of the valve requires more time (denoted by the signal $E_{SFOT2\_Timeout}$), HCl enters the output tank and the transition t_{SFOT3_1} activates safety function $SF_{OT}3$ as second part of the chain. $SF_{OT}3$ closes the output valve FC_D by setting the flow $\dot{FC}_D = -v_{FCD}$ in the state *output_closing*. Once FC_D is completely closed, t_{SFOT3_2} fires and the tank level decreases due to the reflux to the column. The safety function waits for the signal $E_{Cond\_control}$ which announces regular condenser operation with complete separation of HCl and phosgene. If at this point in time the tank still contains HCl and phosgene, $SF_{OT}3$ enters the state *tank_draining* and waits for $h(t)$ to decrease to zero. If $h(t)$ equals zero, the safety function reopens FC_{SF2} in the state *input_opening*, filling the output tank with phosgene, and returns to normal operation via t_{SFOT3_6} when the level rises above a minimum threshold $h_{min} + hysteresis$.

The safety function $SF_{OT}4$ ensures that the output valve FC_D is closed when the tank level $h(t)$ decreases to zero to prevent damage to following subprocesses. The demand for the safety function is a decrease of $h(t)$ below a configurable threshold h_{min} which causes firing of the transition t_{SFOT4_1}. In the state *output_closing*, $SF_{OT}4$ closes the valve FC_D. If FC_D is completely closed, the safety function waits in the state *output_closed* for the tank level to rise over the minimum threshold with hysteresis and switches back to normal operation via t_{SFOT4_4}. If however the tank level decreases to zero before FC_D is completely closed, a safety-critical error *Err_Lvl* occurs representing the failure of $SF_{OT}4$ in preserving a safe system state for the process step. The safety functions of condenser and output tank have to react by transferring their process step into a safe state during the interval between the occurrence and the consequence of dangerous events referred to as *process safety time*. Fig. 10 shows the safety automaton of condenser and output tank. As described in section III, in addition to the modes of operation the safety automaton defines the process safety times and safety-critical errors for each safety function. For SF_C2, the automaton defines the safety time $tsafe_{SFC2}$ whose violation leads to the critical error Err_FC_{SF1}, and the chained safety times $tsafe_{SFOT2}$ and $tsafe_{SFOT3}$ leading to the error state Err_FC_D. Section V-F argues that the occurrence of these safety-critical errors depends on the configuration on both subsystem and component layer and uses the automaton in Fig. 10 to define a reachability problem supporting the prove of correct system configurations regarding functional safety.

To support reuse and reduce modeling complexity, the safety aspect defines safety concepts which can be used to implement safety functions. These safety concepts are weaved into both structural and behavioral models on the component layer. Fig. 11 shows the structural aspect model of the safety concept "2-channel" which introduces a system of two sensors, two processing units and two actuators. The sensors send a dynamic data item to both processing units where they are compared by software modules. The comparison modules on each processing unit then calculate a derived data item and exchange it to verify the calculation on each MCU. Each of the processing units controls an actuator connected to the same physical process. This safety concept introduces redundancy to safety-related control tasks.

Fig. 8 shows the weaving of the 2-channel safety concept to the condenser control structure. Both the node *Condenser_Vortex2* and the processing units *VortexMCU2* and *ControlMCU2* and

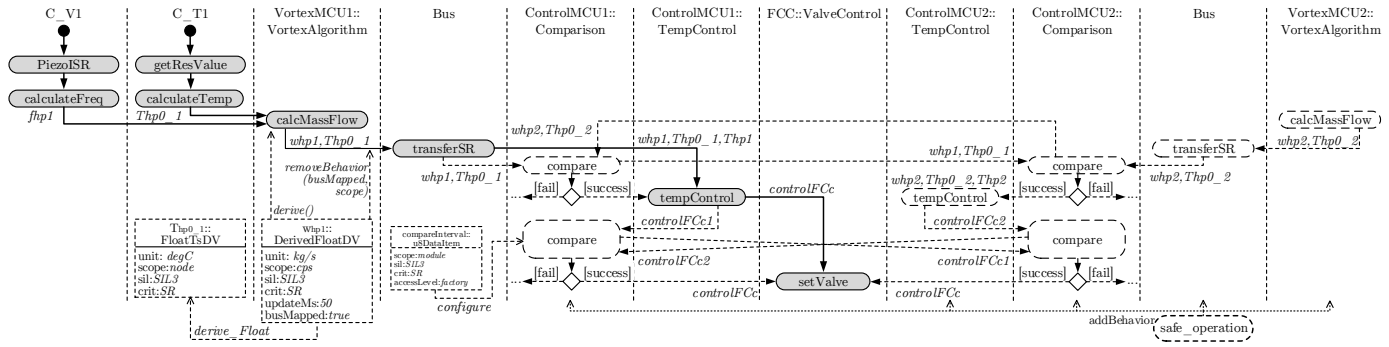

Fig. 9. Weaving of 2-Channel Concept into Condenser Behavior

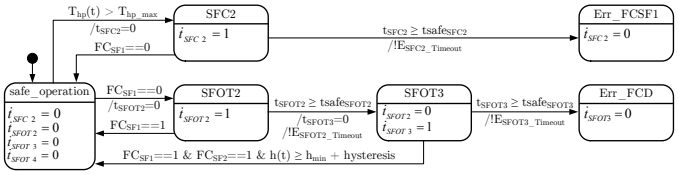

Fig. 10. Safety Automaton of Condenser and Output Tank

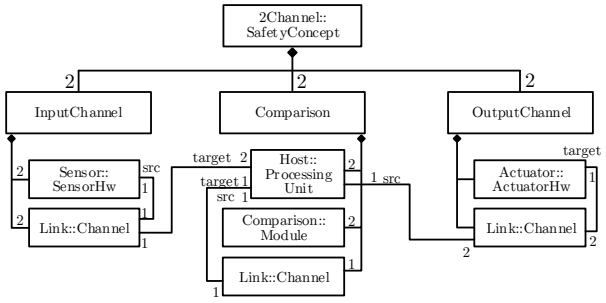

Fig. 11. Structural Aspect Model for Safety Concept "2-Channel"

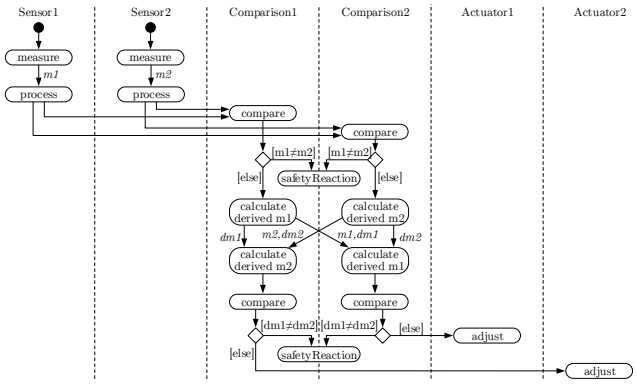

Fig. 12. Behavioral Aspect Model for Safety Concept "2-Channel"

the corresponding bus channels are part of the 2-channel safety concept added to the model via the weaving function *addEntity*. Well formedness rules ensure the correct binding of aspect and domain models using the types of the elements in the aspect model as reference.

The hardware failure information introduced in section III can be used on the component level to document the identified failure modes from a safety analysis and trace their implemented countermeasures to support system certification. In Fig. 8 this is illustrated using the failure *dataCorruption* which occurs at the memory of VortexMCU1 and leads to a dangerous failure with the rate $\lambda_D = 1.7 * 10^{-9}$. The 2-channel safety concept acts as diagnostic measure against the dataCorruption error with a diagnostic coverage DC_D of 90% regarding dangerous failures.

Complementing the structure, a behavioral description of safety concepts is part of the aspect models as shown for the 2-channel concept in Fig. 12. The activity diagram defines the cross-verification data exchange between the concept entities. An exemplary weaving of the safety concept is shown in the activity diagram of the condenser in Fig. 9 using the weaving function *addBehavior* to add entities, activities, control and data flows to the original behavior with correspondence to the safety concept ensured by well-formedness rules. The aspect-oriented approach improves the accessibility of common safety patterns like the 2-channel approach to domain experts without explicit safety knowledge and encourages extended reuse from implementation to documentation and certification artifacts when reusing complete safety concepts.

E. Modeling Data Aspects

We use the data aspect to add configurability to condenser and tank on subsystem and component level and demonstrate how a specific configuration can be defined and bound to the domain models. On the subsystem level, the aspect focuses on the parameters of process steps and their safety functions, providing a process owner's point of view. The major part of the data represent customization and data processing of the CPS parts realizing the process control on the component level. Similar to the safety aspect, weaving functions connect device variables and static data items with the behavioral and structural models on both layers, where the weaving on component level corresponds to the customization of a mass market device to a specific application.

The hybrid automata on the subsystem layer use device variables in their flow equations and static data items in transition guards and invariants. The weaving function *addData* connects the device variables and static data items from the data aspect to the variables and constants of the hybrid automata. Fig. 6 shows the connection of the device variable T_{hp} and its configuration data item $ConfT_{hp}$ as well as the maximum value T_{hp_max} to the condenser automaton. This reduced data set shows that inline modeling of all

configuration data values would magnify the complexity of behavioral models beyond the point of efficient handling. Using our aspect-oriented approach, we promote to hold complete data models externally, e.g. in a separate database and trace the connections to the variables in the automata via the aspect weaving.

On the component layer, the origin of the device variables introduced in the hybrid automata is defined by connecting the device variables to producing software modules in the structural and deployment viewpoint using the weaving function *producedBy*. Dependent on their scope, the device variables are then distributed to all data management modules in the scope of the device variable. Fig. 8 illustrates the source modules of device variables by the labels of their outgoing links. Hardware-related characteristics used by software modules are bound to the respective hardware unit or link via the *addData* weaving function, configuration of software modules is done using the function *configure*. In Fig. 8, the construction variable bluff body width of the vortex sensor C_V3 is defined using the static data item *bluffBodyWidth* and the baudrate of the bus system connecting the nodes is set using the data item *bus1Baudrate*. The storage and replication of static data items is defined in the structural models of the component by the weaving functions *replicate* and *deploy*, as shown in Fig. 8 for the example of the maximum temperature at the condenser T_{hp_max}, which is deployed at *TempMCU1* of the condenser and replicated to *ControlMCU1*.

Configuration data influences the behavior of the control system on the component layer by manipulating the control flows of the overview and detailed activity diagrams. The scope and distribution settings of device variables can add sending behavior via the *addBehavior* weaving function as shown in Fig. 9 for w_{hp1}. The common case of behavioral manipulation is the binding of activities and control paths to device variables as shown for the control state of the tank subsystem in Fig. 13.

F. Model-based Verification

During our case study, we focused on inconsistencies arising from the interdependence between the influences of data and safety aspects on the domain model. The configuration of a domain model via the data aspect can lead to an unsafe system due to structural and behavioral configuration errors. Structural inconsistencies are static misconfigurations which prevent correct system execution, arising from the weaving of the data aspect into domain models with varying complexity. They can be mitigated using the well-formedness rules connected to the weaving functions of our aspect models. For example in the hardware and deployment viewpoint, the safety aspect adds wfrs that constrain the deployment of elements to safety considerations:

$$\boldsymbol{Wfr_{sil\_deploy}} : \forall dep_i \in Deployee, dc_i \in DeployContext : (dep_i, dc_i) \in deploy \rightarrow sil(dep_i) \leq sil(dc_i) \quad (9)$$

wfr_{sil_deploy} ensures that elements dep_i may only be deployed on hardware or partitions dc_i that have a SIL high enough to provide the safety demanded by dep_i. Additional wfrs restrict the control of safety-related actuator and sensor hardware. In our case study, the deployment of device variables to producing modules in Fig. 8 must be checked using wfrs that guarantee that the producing software modules of sr device variables are sr themselves, have a sil equal or higher than those of the device variable and are deployed on an appropriate hardware unit.

Apart from these errors within the scope of a single viewpoint, more complex scenarios arise when interactions of multiple system parts defined on multiple views must be considered. The device variable w_{hp1} e.g. is produced at the vortex sensor *Condenser_Vortex1*, and transferred to both Control MCUs of the condenser where it is compared before further processing (see Fig. 8 and 9). If one of the compared w_{hp} values is configured with a narrow scope, it is not transferred and thus not usable by the comparison module. In addition, static inconsistencies can arise due to influence of both data and safety aspects on the same domain model elements. During our case study, this conflict occurred in combination with check and update dependencies of data items. Check dependencies ensure that the configuration of a data item d_1 is only accepted by the system if the attached data items $d_2..d_n$ perform checks including range compliance on their values. If the safety aspect however removes parts of an activity diagram by the weaving function *removeBehavior* to which the dependent data item $d_{i>1}$ is attached, data item d_1 cannot satisfy its check dependencies and thus cannot be configured. In industrial scale CPS with multiple processes and control loops, even the basic single-view cases discussed above become critical and the set of possible inconsistencies are not manually controllable. Our case study therefore confirms the need for algorithmic support to concisely monitor the structural consistency of safety and data aspects with respect to the domain model during system development and operation.

Inconsistencies arising due to the influence of data items on the system behavior are more difficult to detect. As illustrated in Fig. 10, safety functions of a control system must mitigate risks within the process safety time defined in the hybrid automata on the subsystem layer from the process owner's point of view. In contrast to this, device vendors deliver systems which are highly configurable via data items manipulating structure and behavior on the component layer. As motivated before, this raises the problem of verifying that a system configuration given by an instanced data model does not prevent the safety functions from transferring the system into a safe state in case of a dangerous event within the safety time. A longer query interval of the valve value FC_{SF1} due to energy restrictions e.g. leads to longer activation and thus shorter available reaction time of the safety function $SF_{OT}2$.

To detect and prevent these error scenarios, we propose to extend the activity diagrams on the component layer by worst case execution times (WCET) determined for each action and accumulated for activities in the overview activity diagrams. Since every state in the hybrid automaton of a control system is represented by a distinct overview activity diagram at the component layer, the accumulated execution times of a diagram iteration can be interpreted as the WCET of a transition exiting the corresponding state in the hybrid automaton of the control system. Through the manipulation of control flows in activities, the influence of static data items and device variables on the WCET becomes traceable and can be automatically evaluated from the activity diagrams, specifying the correlation between data and execution time.

Fig. 13 shows excerpts from the tank and safety automaton and the overview activity diagram of the tank *control* state. Transition t_{SFOT2_1} marks the activation of safety function

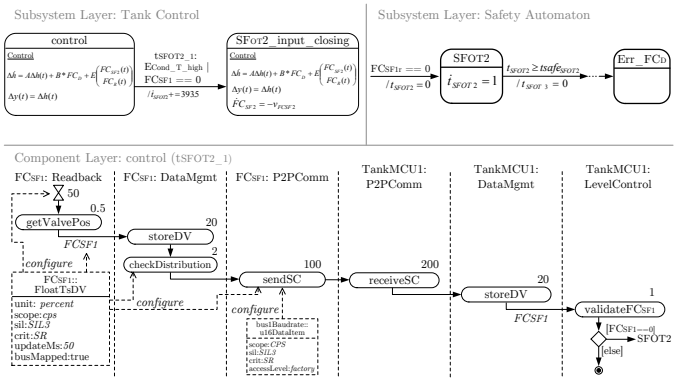

Fig. 13. Dynamic Verification

$SF_{OT}2$ by detection of the closing of valve FC_{SF1}. The example activity diagram shows that the configuration of the device variable FC_{SF1} heavily influences the execution time of the transition: *updateMs* defines the update interval of 50ms, *busMapped* defines the transfer function used and the static data item *bus1Baudrate* influences the transfer time of the device variable. In the example, the data-including WCET of the transition t_{SFOT2_1} is 393.5ms. This delay is added to the counter t_{SFOT2} in the safety automaton which models the occurrence of a safety-critical error in case of a safety timeout. Note that the timer in the safety automation starts at the condition $FC_{SF1r} == 0$ which references the real value of the valve independent of control system delays.

Using this approach, the ability of a control system to perform its safety functions within the safety time under a given configuration can be expressed as reachability problem, where the error states of the safety automaton may not be reached by the configuration under test. When moving from our simple example to more complex models, e.g. the 2-channel safety scheme weaved into the condenser behavior in Fig. 9, the need for automated verification becomes obvious. Such a verification approach must prove that the error states of the system, e.g. Err_FC_D in Fig. 13, can never be reached given the influences of the data configuration on the execution times of the hybrid automata. Existing approaches use over-approximation of the continuous flows inside the automata states to manage termination and execution time and space. We currently investigate the application of the tool *SpaceEx* which was used by Frehse et. al. in [22] for control system verification to our data-driven approach.

VI. Conclusion and Future Work

In this paper, we extend our domain model for hierarchical modeling of CPS in industrial automation introduced in [6] by defining comprehensive aspect models for the cross-cutting concerns of functional safety and data. Both aspects are key concerns in process automation systems. We use the concepts of weaving functions and well-formedness rules to give a concise definition of the valid connections between aspect and domain models, providing a baseline for isolated formal reasoning about data and safety. We apply our aspect models to a case study on the development of an industrial CPS in MDI production, focusing on the conflicts between functional safety and configurability. We identify possible consistency conflicts due to the interdependence between safety and data aspects and the domain model. Based on our case study, we propose approaches for static and dynamic verification of the safety of given configurations and deduce the need for algorithmic support for monitoring and configuring large-scale industrial safety-critical CPS.

Future work includes the formal definition of configuration and data conflicts in industrial CPS and research on algorithms enabling offline and online monitoring of configurations conflicting with the functional safety of the CPS.

References

[1] E. A. Lee, "Cyber physical systems: Design challenges," University of California, Berkeley, USA, Tech. Rep. UCB/EECS-2008-8, Jan 2008.

[2] P. Derler, E. Lee *et al.*, "Modeling cyber-physical systems," *Proc. IEEE*, vol. 100, no. 1, pp. 13–28, 2012.

[3] R. Rajkumar, I. Lee *et al.*, "Cyber-physical systems: The next computing revolution," in *Proc. DAC 2010*, Jun 2010, pp. 731–736.

[4] A. Colombo, T. Bangemann *et al.*, *Industrial Cloud-Based Cyber-Physical Systems: The IMC-AESOP Approach*. Springer, 2014.

[5] *Functional safety of electrical/electronic/programmable electronic safety-related systems*, IEC Std. 61 508, 2010.

[6] D. Kuschnerus, A. Bilgic *et al.*, "A hierarchical domain model for safety-critical cyber-physical systems in process automation," in *Proc. INDIN 2015*, Cambridge, UK, Jul. 2015.

[7] K. Petersen, R. Feldt *et al.*, "Systematic mapping studies in software engineering," in *Proc. EASE'08*, Bari, Italy, Jun 2008, pp. 68–77.

[8] S. Khaitan and J. McCalley, "Design techniques and applications of cyberphysical systems: A survey," *IEEE Systems Journal*.

[9] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ., Keele, UK, Tech. Rep. EBSE-2007-01, Jul 2007.

[10] B. Genge and C. Siaterlis, "Physical process resilience-aware network design for SCADA systems," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 142–157, 2014.

[11] N. Saeedloei and G. Gupta, "A logic-based modeling and verification of CPS," *SIGBED Rev.*, vol. 8, no. 2, pp. 31–34, 2011.

[12] C. Schwarz, "Modelling a real-time control system using parameterized linear hybrid automata," in *Informatik 2011*, Bonn, Germany, Oct 2011, pp. 328–328.

[13] L. Besnard, A. Bouakaz *et al.*, "Timed behavioural modelling and affine scheduling of embedded software architectures in the AADL using polychrony," *Sci. Comp. Prog.*, 2014.

[14] A. Rajhans, A. Bhave *et al.*, "Supporting heterogeneity in cyber-physical systems architectures," *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3178 – 3193, 2014.

[15] F. Saunders, J. Rife *et al.*, "Information flow diagram analysis of a model cyber-physical system: Conflict detection and resolution for airport surface traffic," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 28, no. 12, pp. 26–35, 2013.

[16] T. Stauner, "Systematic development of hybrid systems," Ph.D. dissertation, Technische Universitaet Muenchen, 2001.

[17] M. Zheng, J. Sun *et al.*, "Towards a model checker for NesC and wireless sensor networks," in *LNCS*. Berlin Heidelberg: Springer-Verlag, 2011, vol. 6991.

[18] R. France, I. Ray *et al.*, "Aspect-oriented approach to early design modelling," *Softw., IEE Proc.*, vol. 151, no. 4, pp. 173–185, Aug 2004.

[19] H. Pirkl, J. Bolton *et al.*, "Process for the preparation of highly pure 2,4'-methylenediphenyldiisocyanate," Patent EP1 561 746A2, Jan, 2005.

[20] R. Shah, A. Alleyne *et al.*, "Dynamic modeling and control of single and multi-evaporator subcritical vapor compression systems," Air Conditioning and Refrigeration Center. College of Engineering. University of Illinois, Tech. Rep. ACRC-TR-216, Aug. 2003.

[21] T. Henzinger, "The theory of hybrid automata," in *Ver. Digital Hybrid Systems*. Berlin Heidelberg: Springer-Verlag, 2000, vol. 170.

[22] G. Frehse, A. Hamann *et al.*, "Formal analysis of timing effects on closed-loop properties of control software," in *Real-Time Systems Symposium (RTSS), 2014 IEEE*, Dec 2014, pp. 53–62.