Early Validation of Control Software for Automation Plants on the Example of a Seawater Desalination Plant

Veronika Brandstetter[1], Andreas Froese[2], Bastian Tenbergen[2], Andreas Vogelsang[3], Jan Christoph Wehrstedt[1], Thorsten Weyer[2]

[1] Siemens AG, Corporate Technology, Germany
{veronika.brandstetter|janchristoph.wehrstedt}@siemens.com
[2] paluno – The Ruhr Institute for Software Technology, Univ. of Duisburg-Essen, Germany
{andreas.froese|bastian.tenbergen|thorsten.weyer}@paluno.uni-due.de
[3] Technische Universität München, Germany
vogelsan@in.tum.de

Abstract. In automation plants, the software that controls the behavior of a plant must adhere to strict process requirements arising from the technical processes and from the physical plant design. In current practice, the validation of the control software starts late in the engineering process – in many cases not before the plant is almost completely constructed, leading to increased efforts for correcting revealed defects. Based on an industrial example, we propose an approach that allows early validation of automation software against the plant processes and assumptions about the physical plant design through simulation.

Keywords: Automation Technology, Process Plants, Desalination Plant, Context Modeling, Simulation, Validation, Executable Requirements, Models.

1 Introduction

The development of plants (e.g. processing facilities in the chemical industry, production facilities in factories, or baggage routing facilities at airports) is a complex planning and engineering task. Typically, such plants are designed individually and the entire construction process from the first idea until commissioning takes years, involving many different disciplines like process engineering, physical plant design, mechanics, electronics, and software engineering [1].

The *automation software* of the plant controls its various technical devices (e.g. valves, containers, or conveyor belts). The overall goal of the automation software is to control the involved processes safely, reliably, with sufficient quality, and with the lowest demand on resources, such as time, energy, and material input [2]. Validating whether the automation software satisfies the process requirements of the entire plant is typically conducted at a late stage in the plant development when technical processes and the physical plant design are already defined. However, it is widely acknowledged, that the later the control software of the plant is validated, the higher the effort for correcting revealed defects is, leading to budget overruns and project delays.

This research was funded in part by the BMBF under grants 01IS12005A, 01IS12005C, and 01IS12005R.

In this paper, we propose an approach that fosters early validation of automation control software against the plant process based on specified requirements of the control software and assumptions about the physical plant design. Our approach aims at identifying defects in the requirements for the automation software (in the sense of incorrectly or incompletely specified requirements). The key idea of the approach is to use simulation during early stages of plant development to assess the impact of the specified requirements for the automation software on the plant process. The approach primarily aims at validating requirements for control software of plants. Yet, we expect that our approach can be used widely to validate application software for technical systems with complex technical and physical incarnations, where assumptions about the physical design can be made early in the development process.

2 Running Example

We illustrate our approach by means of a seawater desalination plant[1]. Desalination plants are used to remove salts from seawater to produce drinking water. In our simplified running example, seawater is collected through a subsurface intake system of four beach wells drilled into the seashore. From there the salt water is pumped through pipelines to the seawater tank, where it is stored and pretreated with chemicals before desalination can take place. An overview of a typical plant architecture and the technical architecture of one such beach wells is shown in Fig. 1. In the automation industry, technical architectures are documented using piping and instrumentation diagrams (P&ID). The left hand side of Fig. 1 shows the P&ID for one beach well, which is equipped with a pump and a discharge valve through which water is advanced to the seawater tank, a bypass control valve to adjust the flow rate into the tank, and a set of sensors. These beach well components are controlled by the beach well software to ensure that beach well actuators are controlled according to strict process requirements. In the remainder of this paper, we will focus on the beach well software to satisfy the process requirements shown in Table 1.

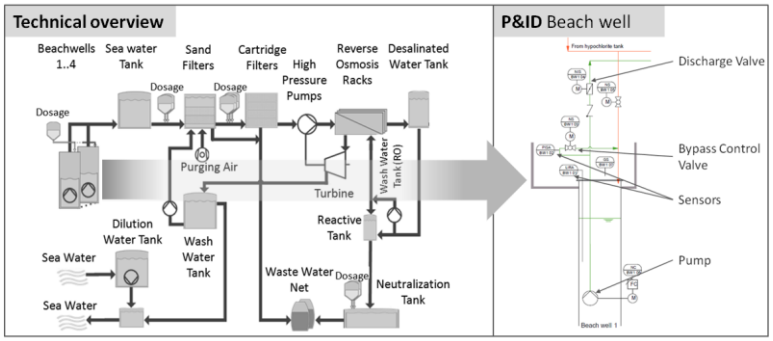

Fig. 1. Technical architecture of a typical seawater desalination plant and one beach well

[1] This running example is based on a free instructional DVD, see http://goo.gl/ppsNNo

Table 1. Process requirements for a beach well

ID	Process Requirement
Req 1	The pump must only run if filling level of the beach well tank is sufficient
Req 2	The pump load shall be minimized using the bypass control valve
Req 3	Discharge valve must be closed before pump starts
Req 4	Discharge valve must be open after pump has started
Req 5	Discharge valve must be closed after pump has stopped

3 Solution Approach

We seek to foster the early validation of automation software by means of simulation during early stages of the plant development process, i.e. when the plant is not yet finished, but when assumptions about the technical architecture of the plant can be made. The key idea of our approach (sketched in Fig. 2) is to document assumptions about the plant architecture explicitly by means of context models (see Section 3.1).

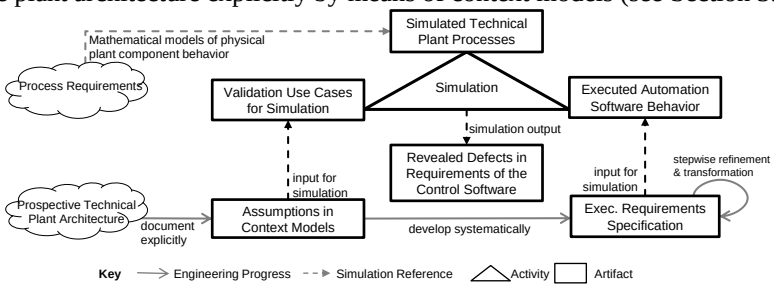

Fig. 2. Overview over the approach

Context models often serve as a reference artifact in quality assurance approaches (see, e.g. [8]) and allow for systematically developing validation use cases (see Section 3.2) with execution semantics, which is necessary for the simulation of automation control software in particular (see, e.g., [9]). Based on the validation use cases, we derive an executable specification, configure the simulation tool, and execute the formal specification and check it against the validation use cases (see Section 3.3). The output of the simulation reveals incorrect or incomplete behavioral requirements of the automation software.

3.1 Assumptions about the Prospective Plant

We employ a number of diagrammatic representations, which we call operational context models, to document assumptions about the beach well's technical environment. These are introduced in the following.

Structural Operational Context Model (SOCM). The SOCM documents structural characteristics of the beach well's software, its physical components, and the components interacting with the beach well. This also entails dependencies between human users and automation software of other plant components, as shown in Fig. 3.

As can be seen, the SOCM depicts the interactions between the beach well components from Fig. 1 and documents the information exchange as well as relevant interfaces. Furthermore, interaction with human users is depicted as well as context influences (e.g., the beach wells pump water to a seawater tank).

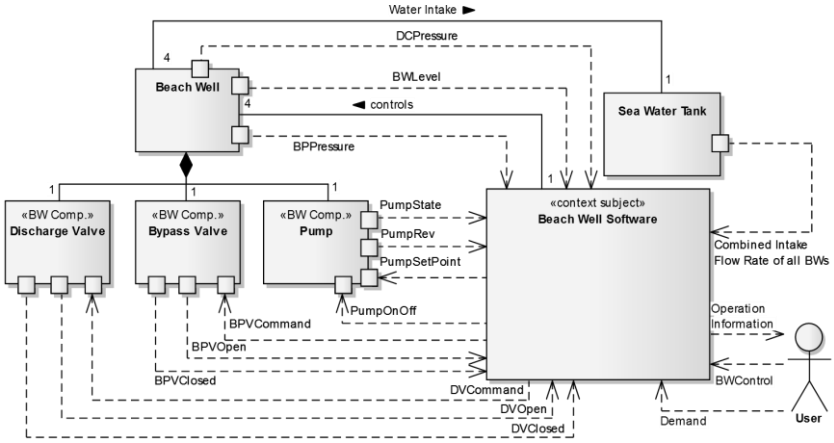

Fig. 3. Structural operational context model of the beach well control software

Functional Operational Context Model (FOCM). The FOCM documents the externally visible functions of the automation software and the physical plant. By abstracting from structural dependencies, the focus is on the functional dependency between the automation software and the physical plant components. This allows adopting a service-oriented view on the functionality by depicting only functions that influence each other. Fig. 4 depicts the FOCM of the beach well software and component functions. The software offers three externally visible functions by which the beach well component functions are controlled: “start beach well”, “stop beach well”, and “balance load”. The signals from the SOCM in Fig. 3 were supplied with concrete values.

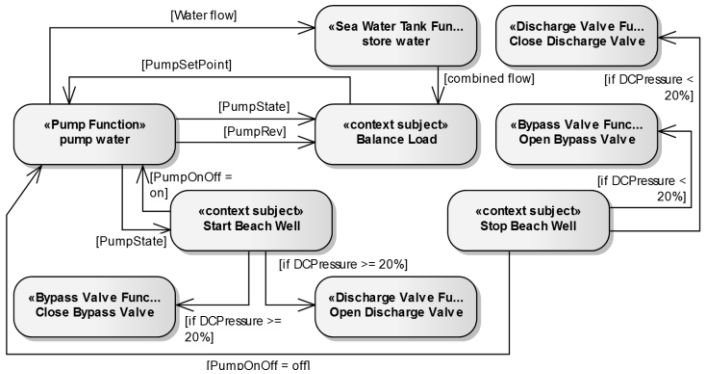

Fig. 4. Functional operational context model of the beach well control software

Behavioral Operational Context Model (BOCM). The BOCM documents the externally observable states of the physical plant components. Internal states of the plant components are abstracted and only states relevant to the automation software are documented. Transitions between states are triggered depending on the values of the signals from the FOCM. Fig. 5 shows an example of the BOCM of the beach well. As can be seen, the externally observable states of the beach well components from the SOCM are depicted as concurrent substates of the entire beach well. The guards on the transitions are conditions specified in the FOCM with regard to the beach well function "start beach well". Both bypass valve and discharge valve can be open or closed. The pump can either be off or on assume some intermediate states.

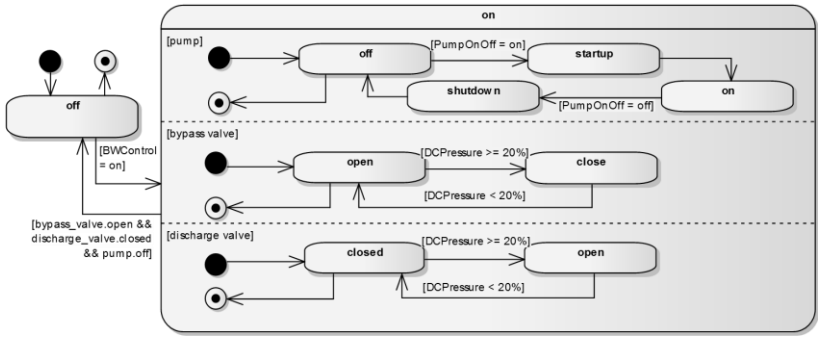

Fig. 5. Behavioral operational context model of the beach well and its components

3.2 Validation Use Cases

Based on the assumptions about the beach well's context (see Section 3.1), validation use cases are developed for every automation software function from the FOCM. Table 2 shows the validation use case for the FOCM function "Start Beach Well" using the Reqs 1, 3, and 4 from Table 1 as pre- and post-conditions regarding the startup procedure for the beach wells.

Table 2. Validation use case "Start Beach Well"

Title	Start Beach Well	
Description	A beach well is manually started by the user of the desalination plant	
Trigger	**BWControl == "The user initiates the start of the beach well".**	
Precondition	**bypass_valve.open == true && discharge_valve.closed == true**	
Postcondition	**pump.on == true && discharge_valve.open == true && bypass_valve.closed == true**	
Step	**Action**	**Actor**
1	The User initiates the start of a beach well	User
2	The Beach Well Software checks whether the beach well is in standby	Beach Well Software
3	The Beach Well Software closes the discharge valve	Beach Well Software
4	The Discharge Valve sends feedback that the valve is closed	Discharge Valve
5	The Beach Well Software starts the Pump with minimal revolutions	Beach Well Software
6	The Pump sends feedback that the Pump is started	Pump
7	The Beach Well Software closes the Bypass valve	Beach Well Software
8	The Beach Well Software opens the Discharge Valve	Beach Well Software
9	The Discharge Valve sends feedback that the valve is opened	Discharge Valve
10	The Beach Well Software reports that the beach well is on	Beach Well Software

Typically, use cases comprise a set of sequential steps called a scenario, which document interactions between the automation software and the physical plant components that lead to the desired post-conditions [4]. Since the process requirements demand a certain sequence of valve actuation and minimal tank filling level, the beach well software must perform several steps as documented in the scenario.

Simulation requires the scenario to be executable. Hence, we formalize the scenario using Message Sequence Charts (MSCs, [5]), as shown in Fig. 6. The MSC defines a sequence of messages based on the interface information and the plant signals from the SOCM (Fig. 3). These messages trigger transitions between states from the BOCM (Fig. 5). This formalized scenario serves as reference for the simulation of the beach well software and beach well technical process (see Section 3.3).

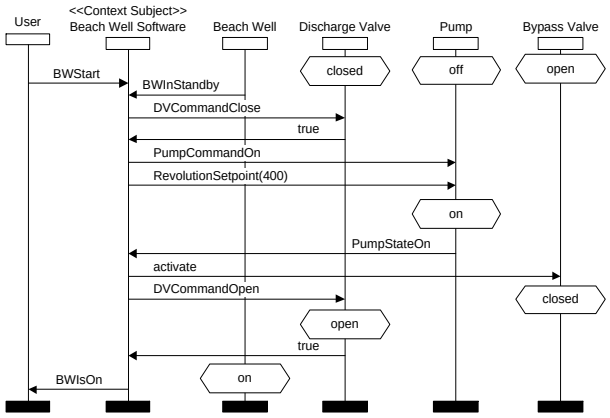

Fig. 6. Formalized scenario from the validation use case "Start Beach Well" in Table 2

3.3 Executable Requirements Specifications

After the validation use case was documented and the scenario was formalized, a simulation tool can be configured (e.g. in Aspen, MATLAB-SIMULINK, or Modelica). For this purpose, the simulated plant process (which gave rise to the process requirements from Table 1) must be modeled and coupled with the formalized scenario in a simulation process, which executes the validation use case.

Modeling the Simulated Plant Process. The plant process is described using algebraic equations representing the physical behavior of the beach well components. These equations can be taken from component libraries, in which the relevant configuration parameters (e.g. height of tank, length of pipes) are stored. Fig. 7 shows the steps involved in modeling the plant process of the beach well. The upper section of Fig. 7 depicts the relevant excerpt of the plant architecture taken from the SOCM (Fig. 3). The middle section of Fig. 7 shows the differential equations for flow, filling level, beach well tank pressure, and the seawater tank pressure. The process models of pump and discharge valve comprise two equations for flow and pressure. Since the four components are associated with three connecting pipes, six additional equations are necessary to describe the flow balance in the physical pipes and the pressure po-

tential at the connection points. Once the relevant physical plant components and the equations representing their dynamic behavior are identified, the simulation tool can be configured (bottom section of Fig. 7)[2].

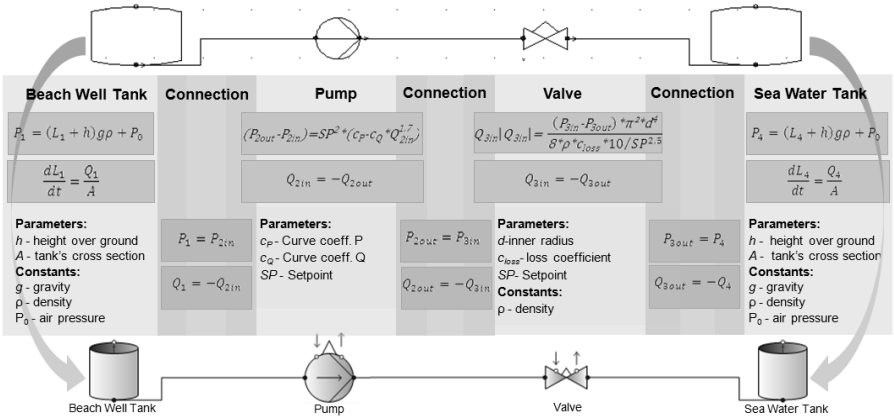

Fig. 7. Modeling the Simulated Beach Well Processes.

Coupling with the Executable Automation Software Behavior. The functional interplay between the beach well software and the plant process must be represented in the simulation tool. To this end, we structure the behavior of the automation software by functions [6], which subsume the process requirements from Table 1 and formalizes them by a behavioral model using the executable specification from Fig. 6.

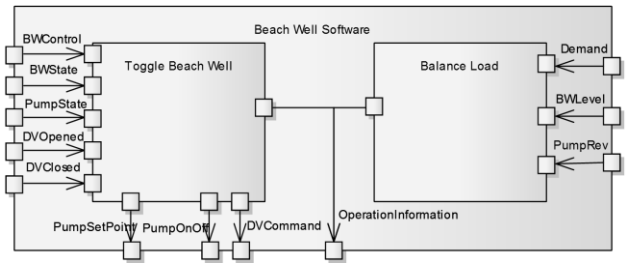

Fig. 8. SysML block definition diagram showing two beach well software functions

Fig. 8 shows two functions of the beach well software: "Toggle Beach Well" and "Balance Load". Each function handles a subset of the input and output signals of the automation software specified in the SOCM (Fig. 3). The behavior of these functions must be specified by an executable behavior description (e.g. a state machine or a code snippet), which allows executing the scenario from Fig. 6, as shown in Fig. 9.

Simulation Process. Based on the formalized validation use cases (see Section 3.2), the equation system documenting the plant processes (Fig. 7) and the automation

[2] In this example, we have used the inhouse tool CoSMOS by Siemens, which can be used to configure and simulate automation plants in an object-oriented fashion, similar to Matlab.

software behavior (Fig. 8), simulation can be conducted by executing the beach well processes and emulating a physical process that takes place therein. The scenario from the validation use cases is used as input for the simulation. If the simulation tool can execute the scenario and the externally observable states from the BOCM match the final states of the physical plant components at the end of the simulation run, the requirements specification is valid with regard to that use case.

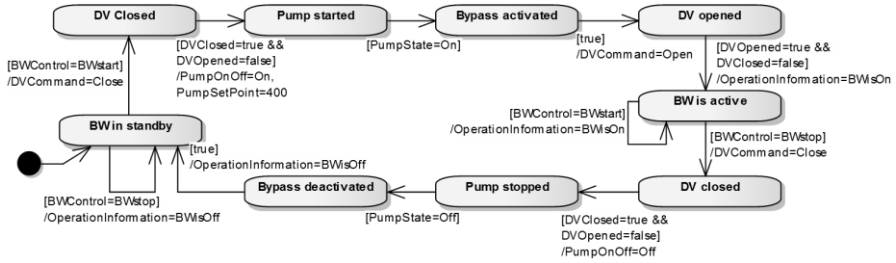

Fig. 9. Behavior of the "Toggle Beach Well" function described by a state machine

4 Conclusion

In this paper, we presented an approach, which enables the developers to validate the control software of automation plants early in the engineering process by first modeling assumptions about the design of the plant and subsequently deriving an executable specification which can be simulated. If no defects are revealed in the simulation it can be concluded that the behavior of the control software is valid with respect to the plant process and the assumptions about the plant design (assuming the behavior is implemented correctly). Albeit this approach was illustrated using an example of the automation industry, we believe it is applicable to any type of software system that controls real-world processes (e.g., business processes for information systems).

References

1. Löwen, U., Bertsch, R., Böhm, B., et a.: Systematization of the Engineering in Automation Plants. In: Automatisierungstechnische Praxis 4, pp. 54-61 (2005)
2. Wagner, T., Wehrstedt, J., Löwen, U., et al: Application and Evaluation in the Automation Domain. In: Model-based Engineering of Embedded Systems, Springer, (2012)
3. Davis, A.: Software Requirements: Objects, Functions, and States. Prentice-Hall (1993)
4. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Professional (2001)
5. ITU-T Z.120: Formal description techniques – Message Sequence Chart, 02/2011.
6. Vogelsang, A., Eder, S., Hackenberg, G., et al.: Supporting concurrent de-velopment of requirements and architecture: A model-based approach. MODELSWARD 2014.
7. Broy, M.: Multifunctional software systems: Structured modeling and specification of functional requirements. In: Science of Comp. Prog. 75(12), pp. 1193-1214 (2010)
8. Jackson, M.: Problem frames. Addison-Wesley. Harlow (2006)
9. Li, D., Li, X., Liu, J., Liu, Z.: Validation of requirement models by automatic prototyping. Innovations in Systems and Softw. Eng. 4, pp. 241–248 (2008)