Softw Syst Model (2018) 17:551–572
https://doi.org/10.1007/s10270-016-0551-z

SPECIAL SECTION PAPER

Toward an execution system for self-healing workflows in cyber-physical systems

Ronny Seiger[1] · **Steffen Huber**[2] · **Thomas Schlegel**[3]

Received: 15 October 2015 / Revised: 30 March 2016 / Accepted: 12 July 2016 / Published online: 3 August 2016

Abstract Cyber-physical systems (CPS) represent a new class of information system that also takes real-world data and effects into account. Software-controlled sensors, actuators and smart objects enable a close coupling of the cyber and physical worlds. Introducing processes into CPS to automate repetitive tasks promises advantages regarding resource utilization and flexibility of control systems for smart spaces. However, process execution systems face new challenges when being adapted for process execution in CPS: the automated processing of sensor events and data, the dynamic invocation of services, the integration of human interaction, and the synchronization of the cyber and physical worlds. Current workflow engines fulfill these requirements only to a certain degree. In this work, we present PROtEUS—an integrated system for process execution in CPS. PROtEUS integrates components for event processing, data routing, dynamic service selection and human interaction on the modeling and execution level. It is the basis for executing self-healing model-based workflows in CPS. We demonstrate the applicability of PROtEUS within two case studies from the Smart Home domain and discuss its feasibility for introducing workflows into cyber-physical systems.

Keywords Process execution · Cyber-physical systems · Workflow system · Internet of things · System architecture · Middleware · Event processing

1 Introduction

Cyber-physical systems (CPS) represent an emerging type of distributed system that integrates a multitude of sensors, actuators and software applications into large networks of interconnected components and things (Internet of things) [3]. A closed feedback loop (*MAPE-K*) exists between local sensing (*Monitor*) and processing on embedded systems (*Analyze*), computing on local or remote cloud-based servers (*Plan*), and controlling local actuators and applications (*Execute*), which are able to have an impact on the physical world [26]. That way, the virtual (cyber) and the real world (physical) are interwoven to a new degree [28]. The trend toward ever more intelligent environments (*Smart Spaces*) shows the increasing importance of CPS throughout all areas of life [10].

Processes have been widely used for describing the flow of activities and work in order to facilitate the automation of repetitive tasks. They can be regarded as high-level programs consisting of method calls and additional logic defining the flow of data and activations between the components of a system. The introduction of processes into CPS carries a great potential for automating tasks and creating intelligent environments of interconnected ubiquitous devices saving resources and assisting users with their everyday activities. However, cyber-physical systems also add new requirements to process-aware information systems that current workflow engines are not able completely fulfill. Among others,

Communicated by Dr. Selmin Nurcan.

✉ Ronny Seiger
ronny.seiger@tu-dresden.de

Steffen Huber
steffen.huber@tu-dresden.de

Thomas Schlegel
thomas.schlegel@hs-karlsruhe.de

[1] Software Technology Group, Technische Universität Dresden, Dresden, Germany

[2] Software Engineering of Ubiquitous Systems Group, Technische Universität Dresden, Dresden, Germany

[3] Institute of Ubiquitous Mobility Systems, Karlsruhe University of Applied Sciences, Karlsruhe, Germany

an increased level of automation in CPS requires: (1) the automated reaction to of sensor events from the physical world; (2) processing of data; (3) the dynamic resource allocation of devices with varying availability and capabilities; (4) the integration of users through various modern end-user devices and modalities; and (5) the synchronization of the cyber and physical world in case of inconsistent process execution states. Current process execution systems have been designed with a strong focus on Web-based workflows involving Web services, high-level human activities and organizational processes [43]. From another perspective, processes can be considered as a more formal abstraction of hardware-related programs regarding only a closed set of homogeneous resources [13]. These approaches leave a gap between hardware-related process execution and high-level interactive workflows involving humans, which is the reason for the only partial fulfillment of the CPS requirements (1)–(5) by state-of-the-art process management systems.

In this paper, we present an integrated process execution system for cyber-physical systems (PROtEUS) intended to bridge the aforementioned gap. Our research focuses on the designing a comprehensive execution system that integrates established components and modeling concepts to cope with the specific requirements for process execution predominant in CPS (1)–(5). PROtEUS introduces processes into cyber-physical environments (*CPS Workflows*) by incorporating a model-based execution engine, a complex event processing engine, a service platform and a dynamic services caller, as well as bidirectional communication channels for interacting with users. The combination of these components on the modeling and execution level yields an increased level of process-based automation in CPS with respect to the processing of events, orchestrating data flow, finding and invoking services dynamically, and ubiquitous human access. Based on these concepts, we propose a mechanism to implement self-healing workflows for CPS using the MAPE-K feedback loop [18] executed by PROtEUS to synchronize the process execution states of the virtual and physical worlds and therefore ensure *Cyber-physical Consistency*.

The paper is structured as follows: Section 2 presents two exemplary scenario processes in the Smart Home domain. Section 3 describes the goals and challenges in process execution for CPS derived from related research and use cases that we address with this work. Section 4 introduces the basic modeling concepts and system architecture for an integrated process execution system and its proof-of-concept implementation in the form of PROtEUS. Section 5 proposes a synchronization mechanism for virtual and real-world processes based on PROtEUS and the MAPE-K feedback loop. Section 6 shows the practical application of PROtEUS for process execution in the scenario processes. Section 7 discusses our approach and results in detail. Section 8 presents related research with respect to the requirements for process execution in CPS. Section 9 concludes the paper and shows starting points for future work.

This paper is an extended version of the contribution [46] published as part of the proceedings of the BPMDS 2015 working conference. The challenges for process execution in CPS are more focused on the specific properties of CPS and used as a guideline for structuring each section of the paper. In addition, the descriptions of the basic concepts are extended to also include the underlying concepts on the modeling level. The paper is extended by a second application scenario and case study to illustrate the newly introduced requirement of synchronizing the cyber world and physical world. Applying the basic concepts of PROtEUS, we present an approach for detecting inconsistencies between the virtual process state and the actual physical process execution, and for restoring cyber-physical consistency. The case study section is extended by a more detailed demonstration of using PROtEUS for process execution. The discussion section is extended with a more detailed evaluation of how PROtEUS meets the requirements for process execution in CPS. The related work section is restructured and contains a more thorough discussion and comparison with related approaches.

2 Scenarios

A typical application domain for CPS is the domestic environment [51]. *Smart Homes* are equipped with a multitude of sensors for measuring physical properties or more complex data, e. g., with the help of cameras, microphones or infrared sensors. Actuators controlling household applications enable the manipulation of the physical world by means of software. These applications can either be running on local computers and embedded systems (e. g., WiFi routers, Smart TVs and service robots) or they can be executed in the form of cloud services. Stationary and mobile devices provide access to CPS components for interaction and control. Ambient assisted living (AAL) employs this Smart Home technology to support elderly people in living a more self-determined life. In the following, we present two typical processes as extensions of the use cases presented in [47] in the AAL domain. Using processes in the AAL and Smart Home domains may help with the automatic provision of assistance for people as well as an increase in comfort and saving of resources. To illustrate these different purposes, we present a process for automated health support (assistance) and for home automation (energy saving)—both also increasing the user's comfort.

2.1 Scenario 1: Health monitoring

Alice is a 73-year-old woman living alone in her AAL-enabled apartment. She is wearing a fitness tracker, which

senses movement, heart rate, blood pressure and sleeping states. The fitness tracker's data can be accessed and processed by the Smart Home. When Alice gets up from the kitchen table, she suddenly faints due to postural hypotension (sudden drop in blood pressure). The Smart Home detects that Alice's health might be in a critical condition and asks Alice if something is wrong. After a defined time without response, an emergency call is placed automatically. Upon the paramedics' arrival at the apartment, the door is unlocked by the medics to provide Alice with medical assistance.

2.2 Scenario 2: Home automation

Saving resources is one of the main goals of using Smart Home technology. By switching on the light and turning on the heating automatically only in the presence of its inhabitants, the Smart Home is able contribute to increasing the users' comfort and also to reducing energy consumption. Parts of Alice's morning routine can be supported by a process-based home automation system: Alice has set her alarm clock to waking up around 7 a.m.. Her fitness tracker analyzes her sleeping phases and sends an event to the Smart Home control system when it is around 7 a.m. and she is not in REM sleep anymore. This triggers the light to be slowly switched on in her bedroom and the alarm clock to play her favorite tune to wake up. After that, the heating is turned on in her bathroom. That way, everything is prepared for Alice to get up and proceed with her morning routine.

3 Goals and challenges

From related work [10,28,38,55,65] and from the Smart Home scenarios in Sect. 2, we identified and focus on several requirements for a process execution environment operating within CPS. These challenges are predominant specifically for the automation of processes in cyber-physical domains. However, this list of challenges is not comprehensive as we only aim at supporting people in loosely coupled, very dynamic cyber-physical environments, e. g., Smart Homes.

C1 event abstraction and processing: CPS consist of loosely coupled devices that usually communicate on an event-driven basis. A multitude of sensors, actuators and other software entities produces constant streams of event data from within the CPS [55]. In order to reflect this property of event-based communication in processes, events have to be abstracted on different levels of granularity and introduced as first-class entities in process definitions. The modeling and processing of events ranging from single event sources up to a large number of heterogeneous events have to be supported to cope with the complexity of modeling the reactions to a combination of events from a multitude of sources.

The integration of events into process control flow facilitates the connection between the *physical* and the *cyber* world. In our application scenarios, a health warning event triggers a process that eventually leads to an automatic emergency call, a door unlocking process is triggered when the paramedics' arrival is detected, and a wake-up event is triggered in the morning. The main challenge for high-level event detection is the processing of a multitude of data from physical and virtual sensors. This comprises homogenizing low-level sensor readings and defining patterns within these data streams that lead to higher-order events, which are contained in the high-level process description [55].

C2 automated data flow: The main goal of introducing processes into CPS is to increase the automation of repetitive tasks. Besides automating the control flow among services, actuators and other entities involved in process execution (e. g., humans), the flow of data also has to be automated to reach an increased level of autonomy in CPS. This requires more sophisticated means for formalizing the flow of simple and complex typed data inside processes in order to automatically process data in high-level process steps and pass data between them (e. g., to evaluate Alice's response) [5].

C3 dynamic service invocation: Heterogeneous components (e. g., actuators, services and applications) are used as process resources for the execution of tasks within CPS. In the scenarios, in-house services are called to switch on the light, turn on the heating and ask for Alice's well-being on a tablet device. After the timeout related to Alice's answer, an external Web service placing an emergency call is invoked and the door unlocking subprocess controls the local door lock actuator. The main challenge for integrating these CPS components and services as process resources originates from the dynamic nature of CPS, i. .e, the varying availability of components at runtime. At design time (i. e., during process modeling), the required CPS components or services that can execute a specific task may be unknown [24]. Even at runtime (i. e., during process execution), the availability of process resources and their capabilities may vary depending on various context factors [10]. Therefore, an abstraction from concrete component instances to a type and capability-based resource allocation is needed to separate the concerns of modeling task behavior and dynamic allocation of process resources.

C4 ubiquitous human access: Sensor readings may be error-prone, resulting in the detection of false or ambiguous high-level events. Due to the close coupling of events and process control flow, we also need to account for falsely triggered processes. User interactions are needed for performing manual tasks, reacting to errors and providing data in case of uncertainty. In the scenario, user interaction is requested asking for Alice's feedback after the detection of a health warning event. Regarding the current trends of interactive

devices becoming more and more ubiquitous and integrated into daily lives as envisioned by Mark Weiser [61], ubiquitous process control and context-adaptive selection of appropriate modalities for human interaction with and within processes as well as their process model integration are important requirements [10,38].

C5 synchronization of worlds and self-healing: Processes in CPS have the capability to influence real-world properties by calling actuator functionality. However, the success of an actuator call is usually evaluated based on the response of the actuator's software/server. Despite malfunctions (e. g., a broken light bulb) or inaccuracies concerning physical properties (e. g., rotation angle of the heating valve), the software may report the successful execution of the operation, which leads to inconsistent states of the virtual process (*success*) and the real-world process (*failure*). The process execution environment has to be able to detect these inconsistencies and perform compensation actions to synchronize both worlds [65].

The execution of processes in cyber-physical systems poses new requirements for process-aware information systems (PAIS) as they have to be able to handle novel properties introduced by CPS [10,28,38]. The aim of this work is to develop a process execution system integrating various existing components and technologies for achieving the goals mentioned above thereby introducing the concept of processes into CPS. The main challenges are posed by the combination of the concepts and properties of low-level sensors and actuators as key elements of CPS with formalized high-level workflows. On the other hand, users also need to be considered as part of the automated processes involving sensors and actuators. PROtEUS is designed to bridge this gap between hardware-related automatic processes and human-centered workflows, which is prevalent in current process/workflow systems. An additional goal is the synchronization of the virtual state of the process execution with the physical effects of real-world process activities. The aspect of achieving real-time process execution is an important factor for CPS [28] but out of scope of this work. The list of challenges is not exhaustive for CPS, but it represents common requirements for PAIS operating in cyber-physical environments.

4 Basic system architecture and modeling concepts

The process execution system for cyber-physical systems *PROtEUS* consists of several components designed to meet the challenges CPS pose for automated process execution as identified in Sect. 3. It is designed in alignment with the reference architecture for workflow management systems proposed by the WfMC [22]. The core of PROtEUS consists of a component-based meta-model describing the structure of process models and a process engine responsible for instantiating these models and executing actual process instances [47]. Access to the engine's process control functionality is achieved by the process manager. A complex event processing (CEP) engine is able to process large amounts of external sensor data and trigger high-level events within process instances. PROtEUS also offers a service platform for deploying and calling services during the execution of processes as well as an external service invoker. A semantic access layer component provides external services for finding and invoking process resources dynamically based on their semantic descriptions [24]. Human interactions are enabled by a human task handler sending interaction requests to interactive clients. The bidirectional communication with the process execution system from remote clients for monitoring, interaction and control purposes is enabled by a WebSocket server. The overall systems architecture is depicted in Fig. 1. The interfaces between the cyber world and the physical world are represented by the software-controlled sensors and actuators of the CPS. The following sections explain the individual components and their interactions in more detail.

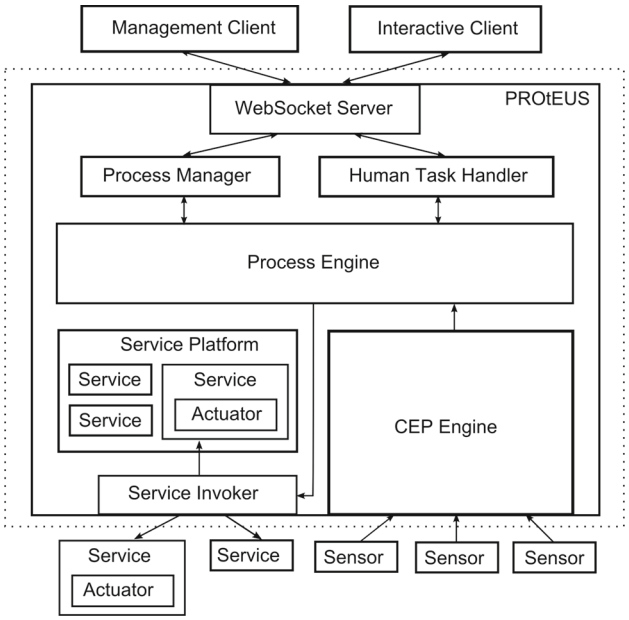

Fig. 1 Overview of the process execution system's (PROtEUS) architecture

4.1 Core process engine and process manager

The meta-model is designed with regard to the properties of CPS [47]. In general, it follows a component-oriented view on process elements. Processes and process steps can be composed hierarchically, and ports describe their input and

output with regard to control and typed data flow. Transitions between these ports allow for modeling the concrete data and control flow between process activities. The following sections explain the CPS specific extensions and elements of this meta-model and their semiformal execution semantics as UML communication diagrams with respect the challenges in more detail.

The core of the process execution environment is a Petri net-based *Process Engine*. We decided to use a self-developed process engine in order to extend the modeling concepts and achieve a better integration of all components necessary for executing CPS processes with respect to the challenges. Petri nets provide the necessary formalisms to model and verify the control flow and behavior of workflows within distributed systems [57]. According to Baheti and Gill [3], Petri nets are suitable for verifying at least the software-related parts of CPS to prove safety features, which is why we decided to also base the implementation of the process engine on Petri nets. The verification of the physical aspects of processes in CPS goes beyond the classical Petri net approaches and needs further research.

The engine instantiates a process model and walks through the process description based on the YAWL [58]-compatible process meta-model described in [47]. Through subtype polymorphism, the process engine is able to call a specific execution method according to the type process step to be executed. There exist special types for control flow logic (e. g., for splits, joins and loops), data flow logic (e. g., for mapping, replication and (de-)composition) and the invocation of various types of services. Through specializations of an atomic or a composite process step, the meta-model can easily be extended to support a wide range of further process elements. Figure 9 shows the graphical representation of a process according to Scenario 1 presented in Sect. 2, which was created using the PROtEUS process model editor [47].

The *Process Manager* component is responsible for managing process models and process instances. The manager enables upload, parametrization and deployment of process models in the execution system as well the control of process instances. Furthermore, the manager provides access to monitoring information about processes and the states of process instances, which can be queried by clients.

4.2 Event abstraction and processing

An important and central part of PROtEUS is the complex event processing (CEP) engine. As CPS are characterized by being highly event-driven (cf. Sect. 3), the process execution environment has to be able to process data from sensors and other sources of events. A CEP engine allows for processing of large amounts of data streams and recognizing patterns within this data [29]. Upon its start, the CEP engine subscribes and listens to a configurable set of event sources. The process meta-model includes a special class of process step—*TriggeredEvent*—allowing the definition of high-level events within a process. In order to define a specific pattern within the stream of low-level sensors events, which will lead to the triggering of a high-level event, an EPL statement (*Event Processing Language* [29]) can be specified as part of the high-level event's attributes. When executing a process instance, the process engine will call the execution method specific to the *TriggeredEvent* process step, which registers a listener for the EPL pattern at the CEP engine. The CEP engine then analyzes the incoming stream of low-level events looking for registered patterns. Upon recognizing a pattern, the corresponding listeners are informed and the high-level events are activated within the process instance, which continues with the execution (cf. Fig. 2). New sources and types of events can be added to the CEP engine at runtime via sensor specific adapters/wrappers. The EPL pattern has to be defined by the process modeler.

Figure 3 shows the special meta-model class for defining high-level events and the communication between the process engine and the CEP engine during the execution of a process instance. After starting the process (1), a listener for the given EPL statement is registered with the CEP engine (2). The CEP engine triggers a high-level event if Alice's average blood pressure drops below 100 mmHg (systolic) and 60 mmHg (diastolic) over a period of 180 s (cf. EPL in Fig. 3). Sensors built in the fitness tracker measure her vital signs and publish the respective events to the CEP engine (3.1–3.3). Upon the detection of the EPL pattern, the CEP engine informs the process engine through the listener (4). The process engine then activates the high-level event *HealthAlarm* in the process and continues with the

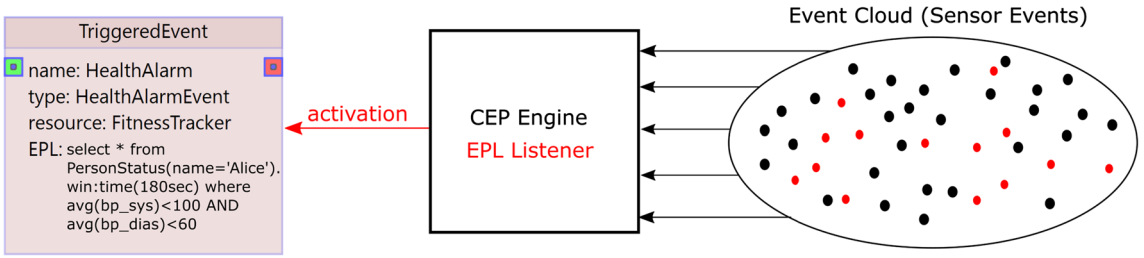

Fig. 2 Complex event processing applied in PROtEUS

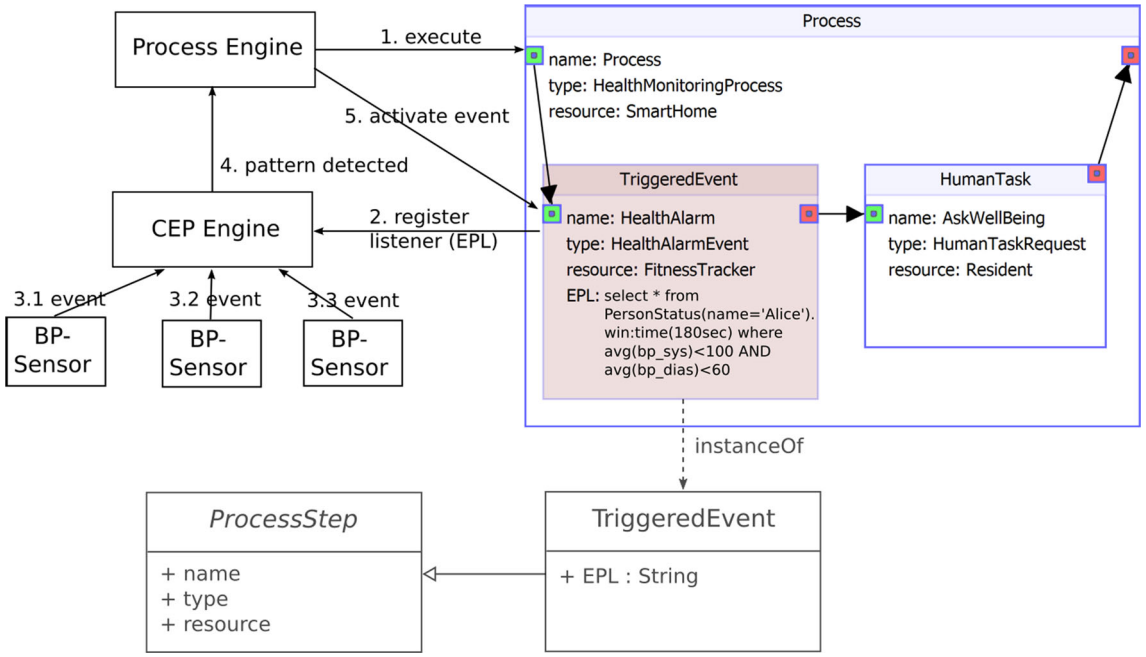

Fig. 3 Communication between process engine and CEP engine during execution

execution (5), i. e., it activates the subsequent *HumanTask* process step.

4.3 Automated data flow

Automating the processing and routing of data flow among process steps, services, sensor and actuators as well as humans requires more formal descriptions of data and input/output of processes and services than it is possible to model with current state-of-the-art workflow languages (e. g. BPMN or BPEL). The process meta-model applied for modeling PROtEUS processes enables the definition of typed input and output data ports for every process step. These types comprise simple atomic data types (e. g. Integer, Boolean, String) and more complex self-defined types including lists, sets and combinations of simple types. The input and output data defined on the process level will be transformed automatically to input/output parameter for the specific service calls corresponding to the process step subclass or presented as input/output forms to the user. The process modeler is also able to describe mappings of specific (sub) data types to data of subsequent process steps. In addition, special process steps allow various data-related operations to be performed in their ingoing/outgoing data (e. g., duplication, assembly or disassembly of complex data). More complex binary data can be integrated into processes by its resource identifier on local or remote storage.

Figure 4a shows the process meta-model elements used to define typed data ports for process steps and mappings between data types. In Fig. 4b, the flow and mapping of process data to input data of a REST service call (specialization of *ProcessStep* class) and the remapping of the service

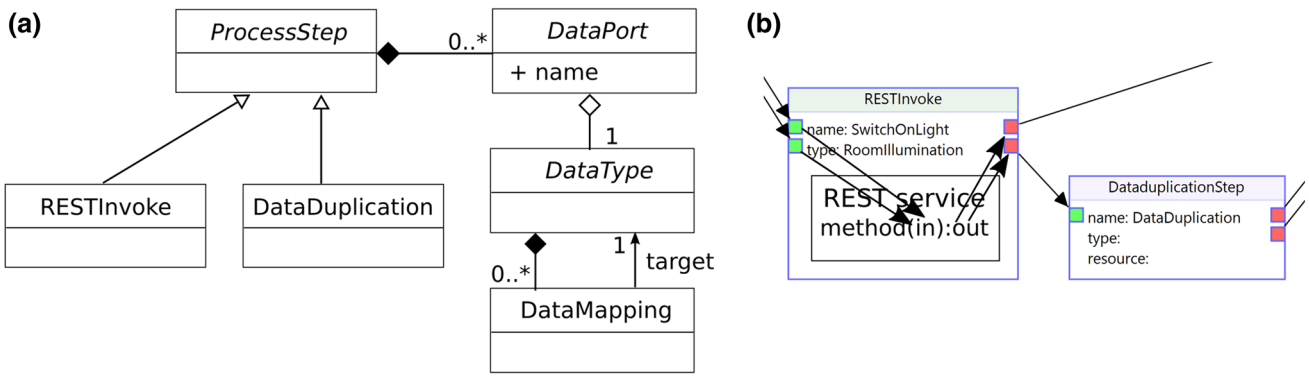

Fig. 4 **a** Meta-model classes for data flow modeling. **b** Data flow on process and service level

response are presented. This mapping is done automatically based on meta-data in the process model and the corresponding attributes of the service parameters in the XML or JSON-based service description. The subsequent process step of type *DataduplicationStep* copies the ingoing data to all outgoing data ports.

4.4 Dynamic service invocation

With Web servers and pervasive services also being integrated into embedded systems, it becomes possible to transparently network more and more heterogeneous devices, Web-based applications and things with each other. To encapsulate functionality and provide standardized and unified interfaces for heterogeneous resources, we follow the widely used service-based approach for controlling actuators and actively invoking remote functionality from within a process. To realize this, we provide two ways for invoking service methods. First, if process resources are known and static at design time, we encourage direct service allocation using the respective resource identifiers within subclasses of the process steps specific to the service to be called. This includes specializations for various service protocols and platforms (e. g., SOAP, REST or OSGi [2] services). The meta-model enables the extension of these types to support further protocols and services. Second, in case of dynamic or context-sensitive services, their dynamic discovery and invocation are supported using semantic queries in combination with the *Semantic Access Layer* (SAL) middleware [24]. Process modelers provide the semantic queries specifying the required service capabilities as well as context constraints per process task (cf. Fig. 5). The semantic queries are evaluated by the SAL on its internal knowledge base, which contains domain-specific information about sensors, actuators and context. The SAL's reply contains the identifiers of the process resources matching the capabilities and constraints defined in the query [24].

All service calls from within PROtEUS are executed by the *Service Invoker* component. It provides access to external Web services and the SAL via protocol specific adapters. Adapters for communicating with the servers have to be implemented for each new type of service. Within PROtEUS, the functionality provided by actuators and other physical devices or virtual applications is currently encapsulated by one of these service types. Upon reaching a specific service invocation process step, the process engine passes necessary attributes from the process model (e. g., URI, method names, parameters or the semantic query) to the service invoker. The service invoker creates requests according to the specific type of service. The process engine continues after the service invoker received a response from the server and passes the result back to the engine. Ingoing and outgoing data are mapped automatically between the high-level process step and the corresponding service. In the case of a semantic query, the service invoker sends the query to the SAL via a REST interface. Depending on the type of semantic query, the SAL either returns sensor readings or invokes a suitable service method. Resulting data are then transferred back to the service invoker.

Thus far, we can call a large variety of services on remote servers from within a process. However, there is also a need for deploying local services only accessing in-house devices and applications (e. g., for security reasons, only allowing access to the door opener from the local network). For this reason, we also integrate a *Local Service Platform* into the process execution environment allowing the local operation of self-developed services. The OSGi-based platform

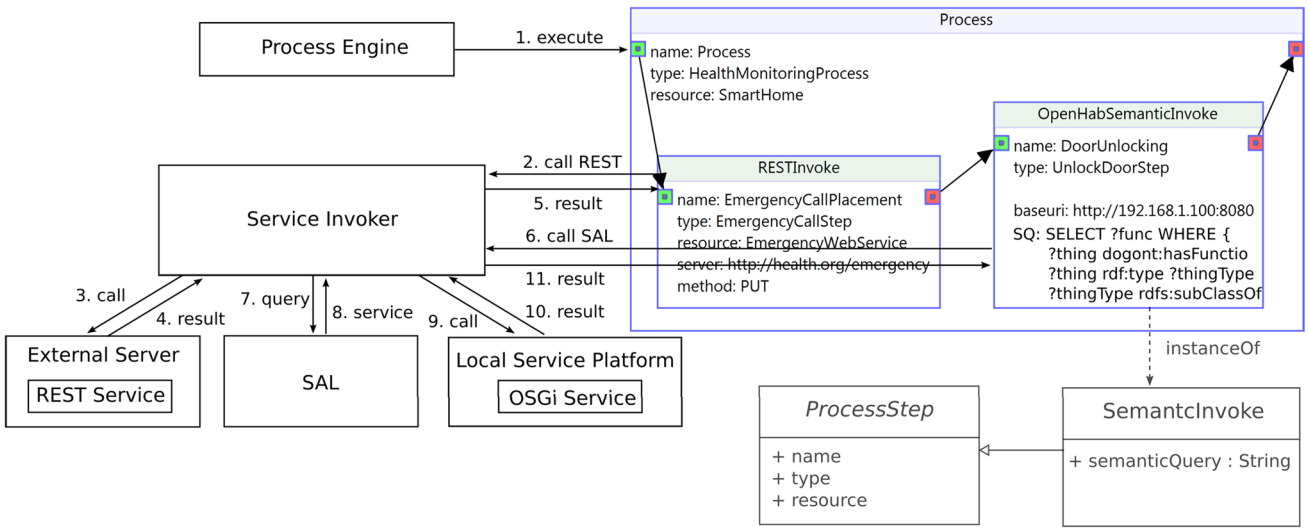

Fig. 5 Message flow during execution of remote and local service invocation via SAL

enables the deployment of bundles and services at runtime. Furthermore, in addition to using the SAL, the integrated service registry allows clients to search for and dynamically invoke services at runtime. The service's location and further attributes used for service discovery are specified in the process model. Figure 5 shows the flow of messages during the invocation of a remote REST service (2–5) for placing an emergency call, the query of the semantic access layer SAL (6–8) by the *OpenHabSemanticInvoke* process step for finding a service to open the door, and the call of the corresponding local OSGi service (9–11) found by the SAL via the service invoker.

4.5 Ubiquitous human access

The users are important factors and stakeholders within CPS. Automated processes may require user interactions in order to, e. g., solve manual tasks, enter data or handle errors that have occurred-often in a timely manner. Analogous to the WS-HumanTask and BPEL4People concepts known from WS-BPEL [27], we introduce a special process step called "HumanTask" into the process meta-model (cf. Fig. 6). When called by the process engine, the *Human Task Handler* is responsible for sending interaction requests to the interactive devices connected to the execution system. The engine waits for the human task handler to deliver a user response according to the specific human task before it continues with the execution. Furthermore, the WebSocket server provides publish/subscribe access to real-time monitoring information regarding the process execution for arbitrary clients, as well as access to control functionality for PROtEUS via the Process Manager component (cf. Fig. 6).

The communication related to human tasks is based on a publish/subscribe mechanism using the WebSocket server, which enables ubiquitous real-time interactions. Interactive clients capable of handling human tasks subscribe to these requests and can be notified in case of relevant changes. The human task handler sends out messages containing the description of the human task to all subscribers. On the client side, a user interface is displayed, which presents the human task step's attributes and parameters. After a response is sent back to the human task handler, the provided user data are incorporated into the corresponding process instance and execution continues. Figure 7 shows the message flow between the process engine and an interactive client subscribed to human tasks (0–7). With respect to the scenario process, either a user response is received (5.1) or a timeout is triggered automatically after a predefined period of time (5.2). If Alice confirms her well-being, the process finishes. In case she answers negatively or a timeout is triggered, the emergency call is placed.

4.6 Implementation

PROtEUS is implemented as a Java-based prototype and is currently employed within a Smart Home lab. As the Eclipse Modeling Framework (EMF) [53] provides a large set of tools for creating meta-models, models and implementations in an automated way, we based the process meta-model and implementation of the engine on Ecore. An integrated modeling environment realized as an Eclipse Plugin based on Graphiti [9] supports the process designer with defining processes for PROtEUS [47]. The event processing engine Esper [8] is used for implementing the CEP engine component. The model-based Smart Home middle-

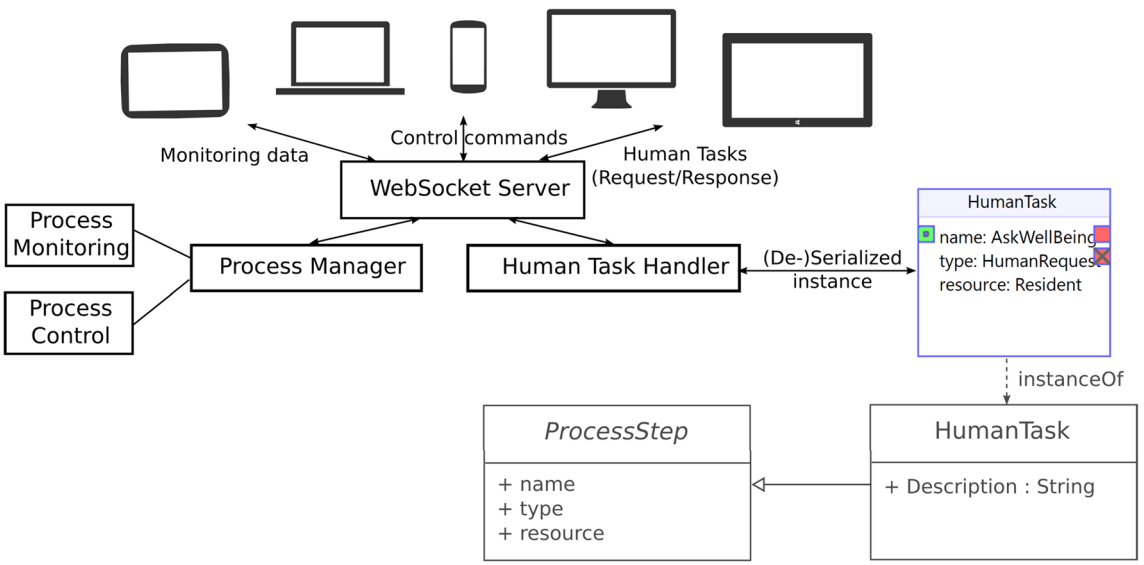

Fig. 6 Interacting with PROtEUS via different devices and modalities

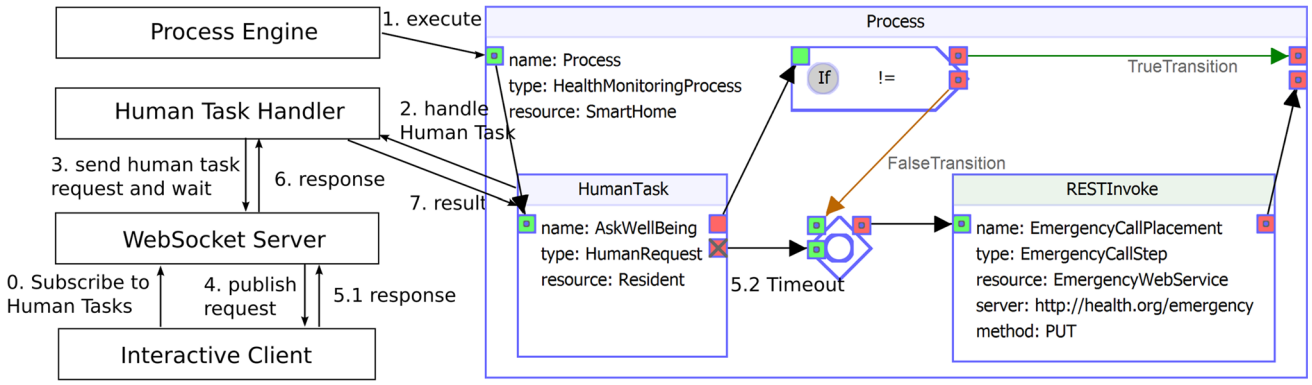

Fig. 7 Message flow during execution of a human task

ware OpenHAB [52] is able to collect and unify sensor data from various sensors. It serves as the main source of event data for the CEP engine in PROtEUS and is used in combination with the DogOnt ontology by the SAL to find Smart Home services [24]. PROtEUS's service platform is based on the OSGi component platform [2]. Using protocol specific adapters, the service invoker is able to call OSGi services running on the local platform or services on remote servers. We implemented adapters for calling SOAP, REST and XML-RPC services; local Java classes; to invoke robot services on ROS [40]; and to call services on the OpenHAB middleware. To realize the WebSocket server, we use an implementation of the Web Application Messaging Protocol (WAMP) [20]. Clients supporting the processing of human tasks and monitoring of processes are implemented on Android and tabletop devices [49].

5 Toward self-healing CPS workflows

5.1 Synchronization of worlds

Traditional BPM-based implementation approaches for processes rely on service-oriented architectures to invoke Web services running on servers and performing the desired tasks [36]. On the client side (i. e., within the process execution system), the servers' responses are used to evaluate the outcome of service calls (i. e., success or failure) and proceed with process execution. This approach of solely relying on immediate software responses is often not sufficient enough when taking the physical aspects of CPS into account. Additional means are required to verify the correct execution of processes and activities that also have an effect within the real world (cf. Sect. 3). A synchronization mechanism has to be introduced in order to provide a consistent projection of the process's virtual properties and state to its corresponding real world equivalent.

As described in Scenario 2 (cf. Sect. 2), an activity within a CPS workflow may invoke an actuator's functionality to switch on the light in a specified room and therefore influence the physical world. The actuator's control software could report the successful execution of the operation back to the process engine. However, current "smart" objects are often not able to verify the actual real-world effect of their operations. In case of the light switch, the light bulb could be broken, which leads to an inconsistent view of the virtual world (*light is on*) and the physical world (*light is off*). Applying the MAPE-K loop for self-adaptive systems [18], *Monitoring* and *Analysis* of real-world data from sensors could be used to relate the effects of process execution to changes in physical properties and therefore detect cyber-physical inconsistencies. The *Plan* and *Execute* phases consist of finding and executing compensation actions in case of errors for continuing with the process execution as intended by the process modeler. The *Knowledge* aspect of the MAPE-K loop corresponds to the context of the CPS and process knowledge in terms of goals and additional properties, which is stored and accessed in an external knowledge base. Figure 8a illustrates the synchronization problem in CPS workflows.

5.2 Self-healing cyber-physical workflows

PROtEUS serves as a basis for implementing the MAPE-K loop for CPS workflows to detect and repair inconsistencies between the physical and cyber worlds. It contains the core components to evaluate sensor data based on predefined patterns, detect divergent real-world effects, find compensating actions and execute replacement processes or services. The explanations of the basic system architecture and modeling concepts in Sect. 4 account for the specific challenges of process execution in CPS. These concepts can be applied to implement a basic feedback loop to achieve the goal of synchronizing the virtual and the real-world states of the process instances and introduce a basic self-healing mechanism for

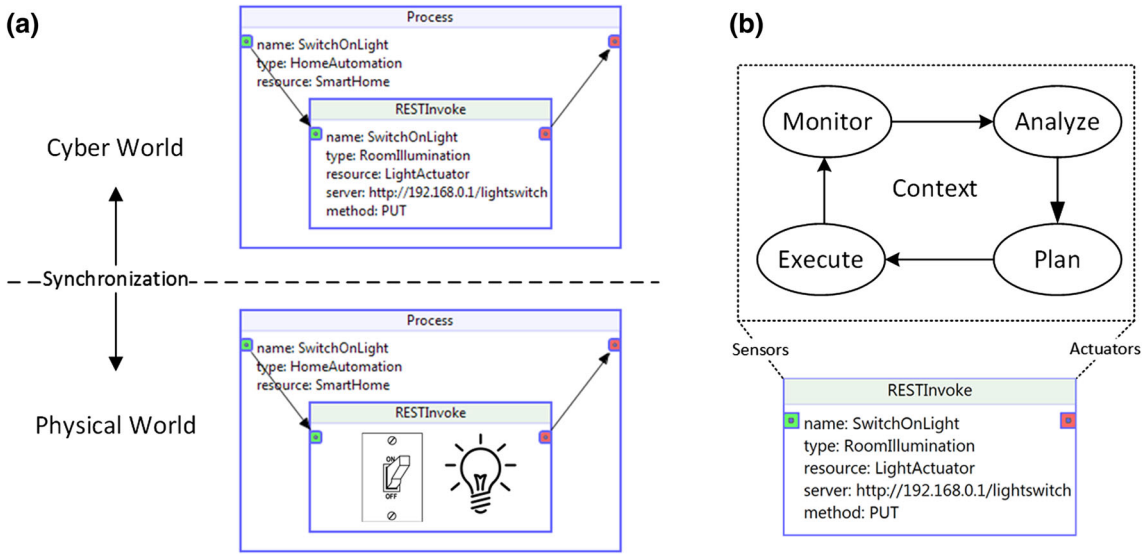

Fig. 8 **a** Synchronization of cyber and physical world. **b** Applying the MAPE-K loop to a process activity

CPS workflows. Figure 8b shows the idea of applying the MAPE-K loop to processes on an activity level using sensors and actuators in the following way:

- **Monitor:** Attached to the particular process or activity, an event containing an EPL statement that defines the changes within the physical context in a certain time frame as result of the execution has to be modeled. An event listener for the EPL pattern is activated upon execution of the activity. From that point on, the CEP engine will actively listen to the event data from the sensors defined in the EPL statement.
- **Analyze:** The CEP engine analyzes the stream of event data from all sensors that are connected to PROtEUS. If an EPL listener is active for a certain pattern, the corresponding sensor data are evaluated by the CEP engine. In case the pattern within the sensor, data are detected by the CEP engine in the defined time frame, the high-level process event is triggered, and the successful execution of the activity can be assumed. If the time period defined in the EPL statement is exceeded, an inconsistency between the virtual execution and the actual physical execution of the activity is likely to have appeared.
- **Plan:** The detection of a cyber-physical inconsistency requires a compensating action to be performed to resynchronize the cyber and the physical world Petri nets. Based on the semantic meaning of the activity or process (e. g., described by its goal or intent), a semantically equivalent and compatible process or service can be searched in an external repository or via the semantic access layer, or the activity/process instance could be adapted structurally. Finding a suitable replacement or adaptation strategy is the main goal of the planning phase. However, with CPS workflows also affecting the physical world, executing rollbacks or repetitions of process steps or service calls is not always a feasible option due to their effects on the real world.
- **Execute:** In the execution phase, the adapted process or the identified compensation action/service is executed by the basic PROtEUS system (i. e., via the process engine or the service invoker). This could be preceded by a simulation of the effects of the execution on the CPS. Upon execution of the adapted process or activity, the MAPE-K loop has to be repeated in order to check the successful execution of the compensation action and confirm cyber-physical consistency. As a last resort for repairing the state of a process activity in case no successful compensations are found or executed, a *Human Task* can be triggered to ask the user to take care of the problem. Again, this human task could be evaluated by the MAPE-K feedback loop and the process execution could continue automatically after the pattern in the sensor data is recognized.
- **Knowledge:** All of the MAPE phases described before require access to context data and process-related knowledge. The EPL patterns assume a known data model for sensors and other event data in the monitoring phase and require the explicit definition of changes within the context data; the planning of compensations needs knowledge about the process's goal and service capabilities to find and execute an alternative action. The ontological model within the SAL's knowledge base acts as the central component for describing and finding suitable sensors and actuators from the Smart Home domain. It will be the basis of future developments with respect to automating and improving the *Analyze* and *Plan* phases.

This MAPE-K loop can be implemented as a generic subprocess applying the same process modeling notation and execution engine that is used for regular processes. Regarding the individual MAPE phases as repeating subprocesses consisting of event-driven behavior and active service calls could increase the autonomy of process management and eventually lead to self-adaptive processes. Case study 2 in Sect. 7 illustrates the practical application of the MAPE-K concept for the Home Automation scenario. With the help of the MAPE-K loop, we are able to extend the commonly known ACID properties of transactional and distributed systems by the dimension of *Real Word Consistency* (ACID-*R*) for the special case of workflows in CPS. A more elaborate investigation and discussion of this concept will be part of our future work.

6 Case studies

In order to provide a proof-of-concept evaluation of our approach, we conducted two case studies employing a prototype of PROtEUS in a Smart Home setting regarding the scenarios presented in Sect. 2. We will discuss these case studies and related challenges (Sect. 3; **C1–C5**) to evaluate to what extend PROtEUS is able to meet the requirements identified in previous sections. The focus of this work is on presenting a concept and implementation of an integrating system for process execution in CPS and showing that all addressed challenges can be met by the concepts introduced in previous sections.

6.1 Health monitoring

We modeled all steps from the health monitoring scenario (cf. Sect. 2) as a process according to the meta-model. Processes can be modeled with the help of the Graphiti-based PROtEUS process model editor [47]. It provides simple drag and drop functionality to create generic processes for various domains by domain experts. The health monitoring process model ready to be executed by PROtEUS is shown in Fig. 9. An active instance of the *HealthMonitoringProcess* listens for a *HealthAlarmEvent* in the HealthAlarm process step. The EPL pattern modeled in this high-level event (cf. EPL Statement in Fig. 2) defines the activation of the event when Alice's average blood pressure drops below 100/60 mmHg for at least 180 s, which indicates postural hypotension. Analyzing the sensor readings from the fitness tracker, the CEP engine triggers the high-level process event when the pattern appears within the sensor data (**C1**). The subsequent process step is activated and a *HumanTask* request is sent out to a tablet device asking for the resident's well-being (**C4**). The IF process step defines that if the response data received are positive (**C2**), the process terminates; in case of a negative response or the activation of a timeout (defined as a special type of port), a REST service placing an emergency call is being invoked (**C3**). Afterward, a subprocess of type *SecureOpeningProcess* is instantiated. An EPL statement for sensing the physical world by means of a fingerprint scanner is modeled within a high-level event. The event is activated when a person having the active role of a "medic" is authenticated successfully at the main entrance door (**C1**). This leads to the dynamic lookup and invocation of a service capable of unlocking the door. The SAL finds a local OSGi service within the Smart Home to trigger the unlocking of the door (**C3**) and the process instance terminates.

Figure 10 shows the finished health monitoring process in the execution perspective of the PROtEUS environment. Here, the execution trace and state variables as well as port values can be viewed by domain experts for all process elements. To make the process execution more accessible to non-expert users, a mobile app enabling a simplified process management (cf. Fig. 11a) and handling of human tasks (cf. Fig. 11b) is also available (**C4**). The process management view lists available and running processes and provides detailed information for every process (instance) upon selection. The human task view notifies the user automatically about new requests and displays a simple dynamic user interface presenting ingoing data and forms for providing outgoing data as well as additional information.

6.2 Home automation

The home automation process described in Sect. 2 is shown in Fig. 12. After instantiation of the *MorningRoutineProcess*, an event listener for the EPL statement defined in the *WakeUp* event is registered at the CEP engine. According to the event pattern, the event is triggered when the evaluation of sensor data from Alice's fitness tracker by the CEP engine indicates that she is not in REM sleep and the current time is between 6.30 a.m. and 7.30 a.m. (**C1**). Following the two services–one REST-based service for switching on the light and one SOAP-based service for playing the alarm tune–are called in parallel (**C3**). The process activity responsible for illuminating the room is annotated with the MAPE-K loop symbol, which indicates that the process engine has to execute additional steps to ensure cyber-physical consistency (**C5**) by means of the MAPE-K loop (cf. Sect. 5). This subprocess is executed in the following way, illustrated by the sequence chart in Fig. 13:

- **Monitor**: The process activity *SwitchOnLight* is extended with a high-level event that contains an EPL (*EPL1*) statement defining an increase in the luminance level in the bedroom within 5 s as effect of executing the activity:

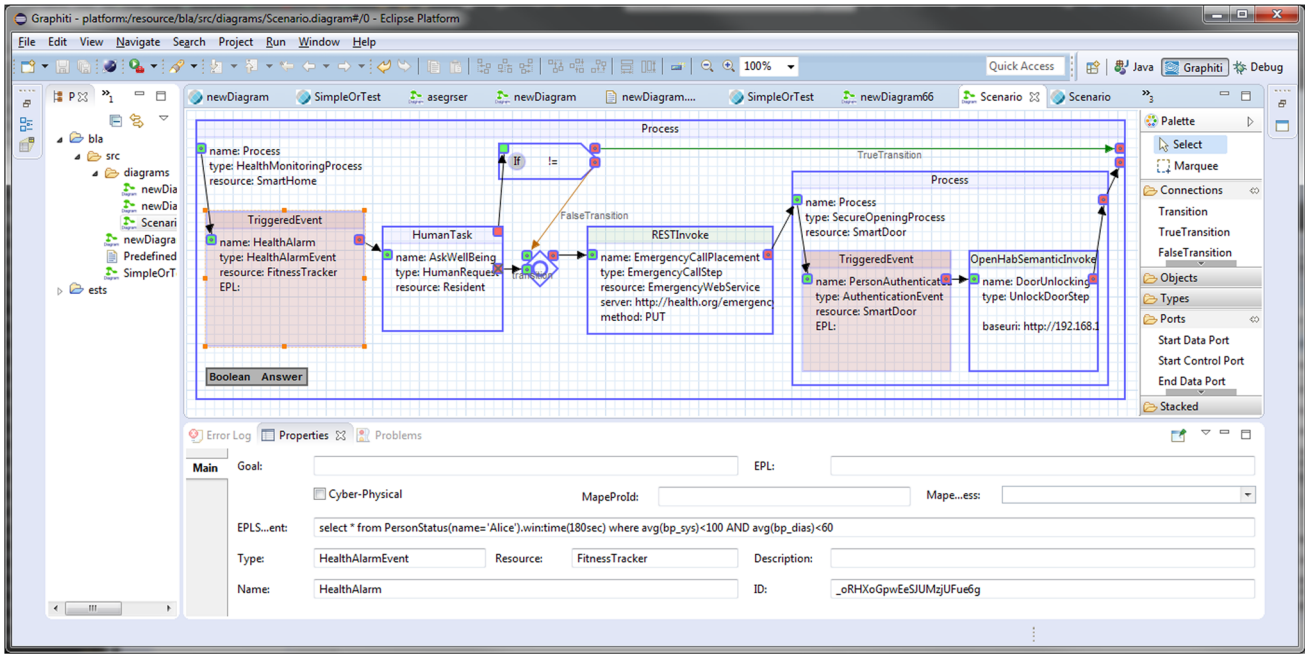

Fig. 9 Health monitoring process from Scenario 1 (cf. Sect. 2.1) in PROtEUS editor

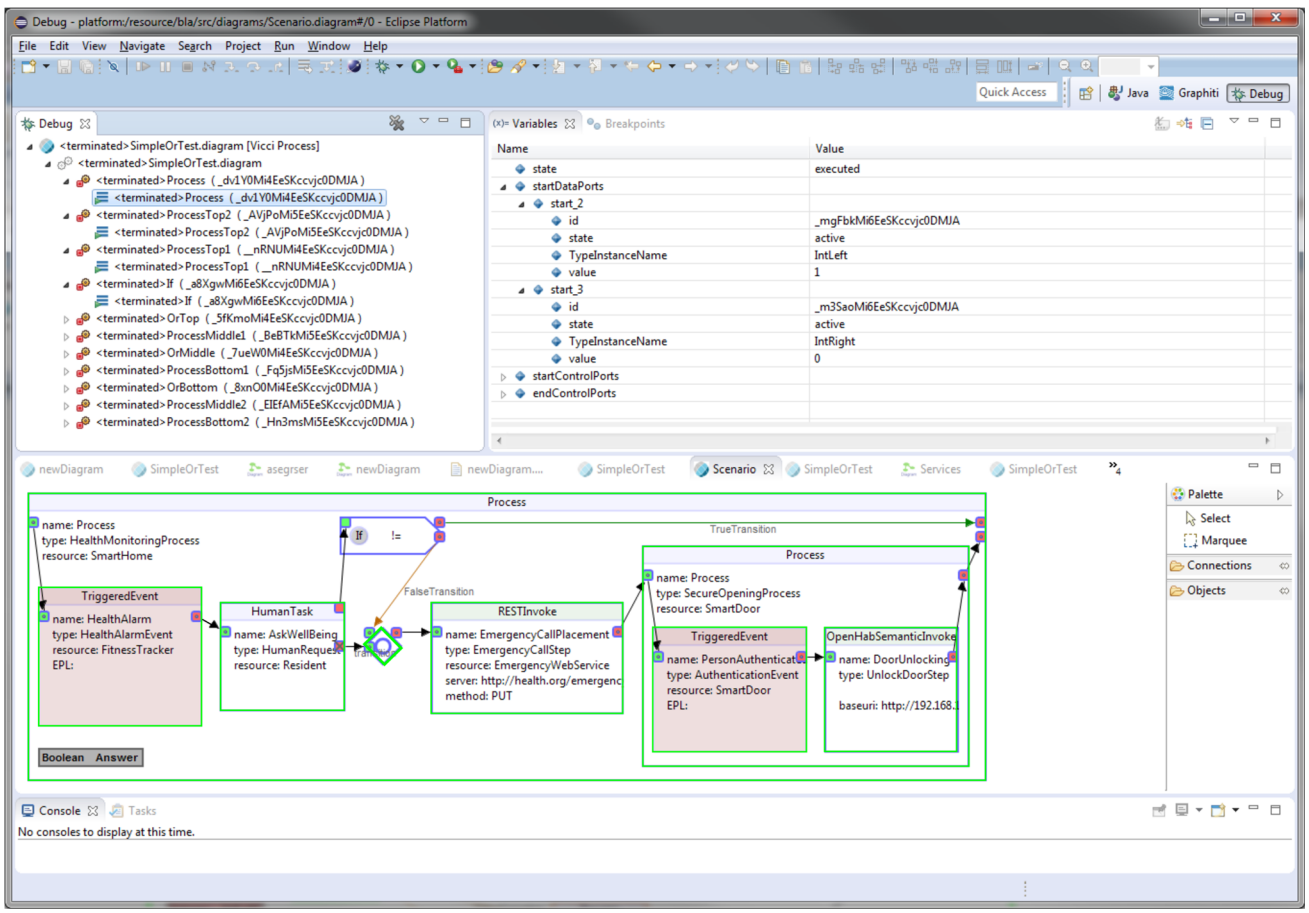

Fig. 10 Health monitoring process from Scenario 1 (cf. Sect. 2.1) in PROtEUS executor

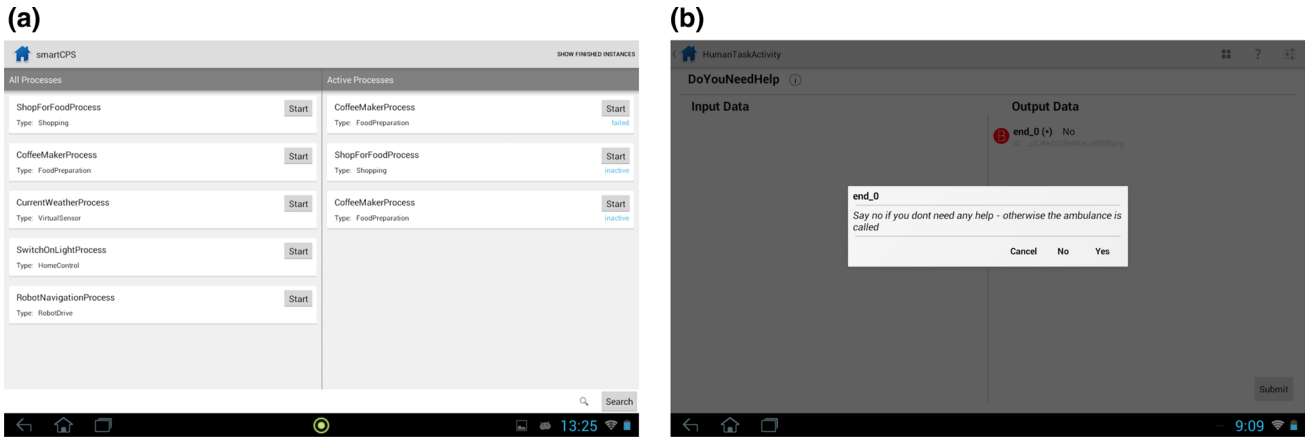

Fig. 11 Mobile app for **a** managing processes and **b** handling human tasks in the Smart Home domain

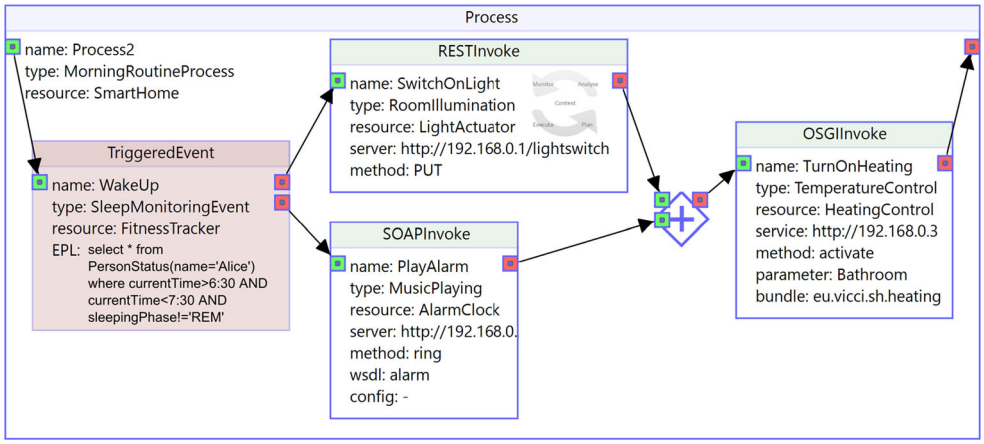

Fig. 12 Home automation process from Scenario 2 (cf. Sect. 2.2)

SELECT * from Light(Room = 'Bedroom').win:time(5 s) where avg(value) < 10

This pattern leads to the triggering of the event if there is no increase in the room's lighting over 10 lux within the next 5 s, which indicates that there is a malfunction of the light switch and the room is still dark. A listener for the EPL pattern is registered in parallel to the execution of the SwitchOnLight activity and data from the bedroom's light sensor are then analyzed.

- **Analyze**: The CEP engine processes data from all connected sensors. As the light sensor sends only values below the 10 lux threshold after the invocation of the light control service, an inconsistency between the service response (*success*) and the real-world state can be assumed. As the result of this inconsistency, the planning phase is initiated.
- **Plan**: In the planing phase, a compensating action or adaptation of the process is computed. Our current approach of finding a suitable compensation is based on the semantics of the process activity and its goal. An additional *Type* attribute describes the domain-specific meaning of the process (here: *RoomIllumination*), which is used for finding a replacement process of similar type. The semantic access layer is queried in order to identify a compensation for the failed process activity, i. e., to find a service and therefore a device capable of illuminating the room. This requires additional context information to be taken into consideration–the opening of the window blinds is only suitable if there is already daylight shining in from the outside at around 7 a.m., i. e., in summer time. The result of this query is passed back to the process engine: call the window shutter service by executing the *OpenWindowShutter* process in the bedroom.
- **Execute**: The compensating process is executed and the MAPE-K loop is repeated to verify the successful lighting of the bedroom.
- **Knowledge**: The realization of this scenario relies partially on the explicit modeling of knowledge (e. g., in the EPL patterns and process descriptions) and on the information contained in the knowledge base of the SAL, which includes a domain ontology (DogOnt for the Smart

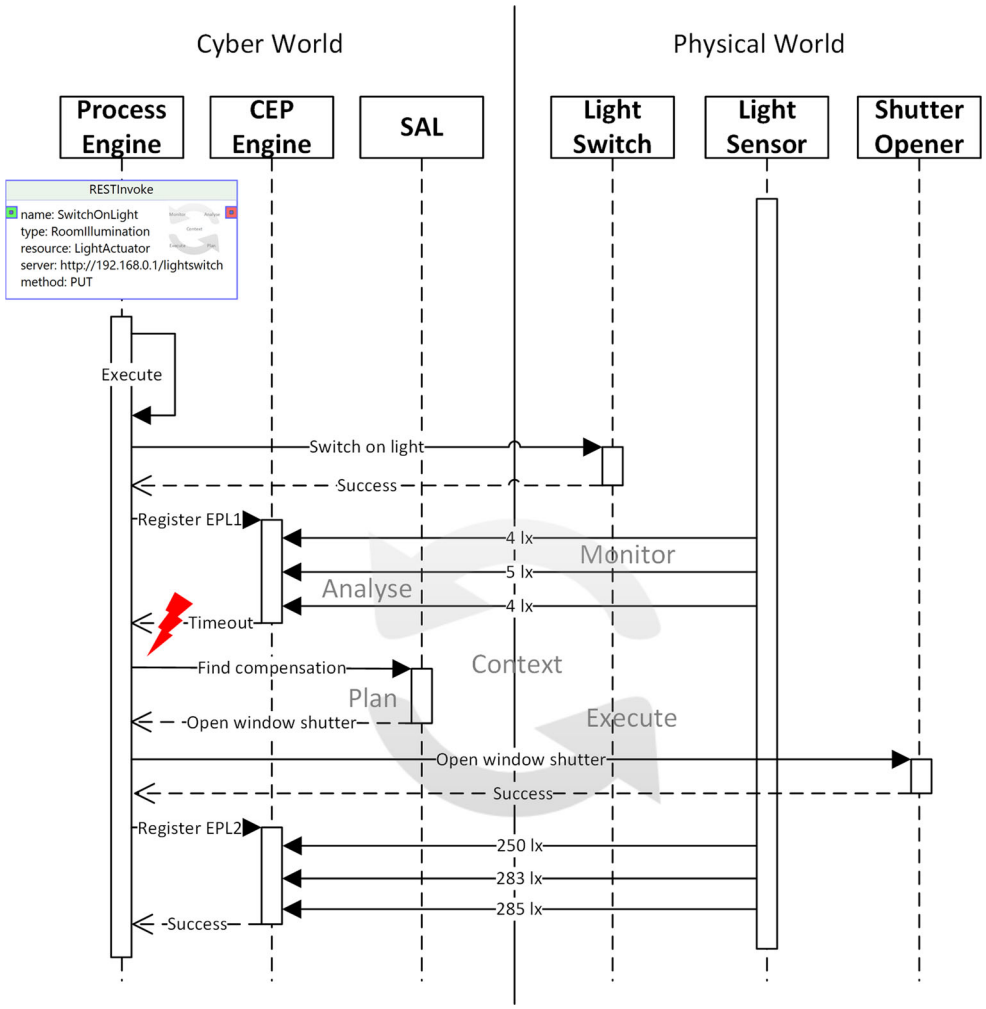

Fig. 13 Sequence chart for the MAPE-K loop in Scenario 2

Home domain [24]) describing capabilities of actuators and sensors.

Again, the process activity that opens the window blinds is extended with a high-level event containing the following EPL (*EPL2*) pattern:
SELECT * from Light(Room = 'Bedroom').win:time(20 s) where avg(value) < 10

The CEP engine evaluates the event stream from the light sensor with respect to the new pattern. As the opening of the window blinds usually takes longer than switching on the lights, sensor data will be analyzed for a longer period of time. The event stream for the second process activity in Fig. 13 shows an increase in the luminance levels, which indicates that the activity has been executed successfully and that cyber-physical consistency has been restored. The process execution then continues with the automated morning routine. On receiving the confirmation of completion for both service invocation process steps, the AND join is activated and the following process step calls the heating control in order to turn on the heating in the bathroom. This step again could be marked as CPS process activity requiring the MAPE-K loop to be executed in order to ensure cyber-physical consistency (e. g., based on the increase in temperature over a certain period of time).

7 Discussion

After a demonstration of the concepts for basic process execution in CPS and of the self-healing capabilities of PROtEUS by means of two case studies. This section discusses advantages and disadvantages of the concepts applied in PROtEUS to meet the identified challenges for executing processes in cyber-physical systems.

7.1 Event abstraction and processing

The CEP engine integrated into PROtEUS allows for an efficient processing of large amounts of low-level sensor events

(up to 50,000 events\s) from various sources [17]. High-level event abstractions in process models are necessary in order to cope with the number of possible event sources within CPS. EPL statements provide expressive semantics to define temporal and logical dependencies—and even basic forms of computations—among subsets of low-level events that will lead to the activation of high-level events [29]. As event-driven architectures are the predominant paradigm within CPS, the ability to process and react to simple and complex events is an important requirement for process-enabled systems controlling CPS. The meta-model supports the definition of events and their properties in the form of a specialized process step class containing a description of an event pattern. As this concept can be applied in a generalized way, other types of process meta-models (e. g. BPMN [5], BPEL or YAWL) can be extended to support this form of event abstraction, too. Introducing a new type of event into the process meta-model or annotating existing events with an EPL statement is sufficient to define an interface for connecting a CEP engine to the corresponding process engine. Upon reaching a high-level event defined in a process, the engine creates a listener, which will be notified if the low-level event pattern is detected. In this way, hardware-related sensing and high-level processes can be connected.

As the EPL pattern enables the scalable processing of event data on different levels of granularity–e. g., for a single sensor, but also for a complex combination of sensors or even on the level of event types–the complexity of the process model can be reduced significantly [17]. The formalism focuses on the type of event rather than on the corresponding device instance, which enables a more flexible coupling of devices at runtime as not all the event source have to be known at modeling time. Compared to related approaches [32] that treat individual sensors as individual resources, our approach is more suitable for CPS with a high number and fluctuating availability of resources. However, sensor data have to be unified before being injected into the CEP, which employs its own event data model. The semantic description of sensor data provides remedy to this issue as it facilitates the modeling of EPL conditions and correlates event streams. As long as PROtEUS is able to convert event data into its data format, arbitrary event sources can be used and added–even at runtime. Nevertheless, the precision of high-level event recognition–and thus, the recognition of the current context–from low-level sensor data depends on the accuracy of the EPL pattern provided by the process modeler.

7.2 Automated data flow

The process meta-model provides means for formalizing the flow of data between process steps via data ports for ingoing and outgoing data. As these ports are typed and annotated with additional information, we can automatically map data between the low-level services called from a process step (input and output parameters) and the corresponding high-level process steps based on these meta-data. This increases the level of automation for processes in CPS as there is no need for additional manual interaction to orchestrate the data flow. We achieve more sophisticated ways of expressing data flow with the introduction of specialized types of process steps for mapping, (de)construction and replication of complex process data types as well as data mapping mechanisms among subtypes of complex data. Due to the model-based description of ingoing and outgoing data, we can also offer suitable interaction modalities to the user for providing syntactically correct human task data. However, we currently do not support any form of semantic validation of user input.

7.3 Dynamic service invocation

With the a service-based approach for actively invoking functionality in PROtEUS, a wide range of devices and applications can be called from within a process instance. The service invoker supports several standardized as well as proprietary service protocols by implementing corresponding adapters (cf. Fig. 4) and, therefore, acts as a middleware enabling a homogeneous view on CPS components. The process meta-model allows for an easy extension of the supported service types via subinheritance. Loading the service parameters from a standardized interface definition (e. g., a WSDL file for SOAP services) facilitates the semiautomatic modeling of a service call and its parameters as well as a direct linking to the corresponding process ports. A drawback of following a service-oriented approach for actuators is that their functionality has to be encapsulated in a service deployed on a server. This can introduce an overhead with respect to the effort for implementing and running the service as well as an increased response time. Therefore, we provide a service platform for deploying local services based on OSGi, which also yields a basic level of security as not all services should be accessible on remote cloud servers (e. g., the door opener). The local deployment restricts access to clients from within the local network only. Further protocols may govern access to ensure security, but this is out of scope of this work.

In order to cope with the dynamic availability of CPS components, we provide two concepts for allowing dynamic discovery and invocation of CPS components. First, the service invoker is able to forward semantic queries to the SAL to find resources capable of executing the current task in the defined context at runtime. The SAL includes a semantic knowledge base providing a homogeneous view on capabilities and properties of CPS components that can be used to allocate and invoke suitable IoT services. However, this type of dynamic service invocation introduces an overhead due to the query processing. As this delay may increase with the size of the domain knowledge base, direct service calls should be

preferred in case of known and static resources [24]. Second, the integrated OSGi service platform provides a local lookup functionality for registered services based on process parameters. These mechanisms increase PROtEUS's capability of handling the varying availability of CPS resources for statically modeled process steps. Complementary to that, concepts for increasing the flexibility of processes by means of process adaptations may also be introduced into PROtEUS to handle variability [15].

7.4 Ubiquitous human access

The human task handler allows for publish/subscribe real-time access to interact with processes in the form of high-level human tasks. Clients capable of providing interactions receive requests for human tasks, which specify the required action, data and type of interaction in the form of serialized process step instances. Responses only have to include the data necessary for the process instance to continue. This leaves the processing of ingoing and outgoing task data to the client, which is completely implementation agnostic. Arbitrary applications and devices can be used for handling human tasks sent by PROtEUS to receive and respond to human task requests and provide data in case of errors as well as access monitoring information. This allows for developing and connecting modern interactive devices and ubiquitous applications to PROtEUS to offer (possibly generated) context-dependent, multimodal user interfaces for process interaction and reaction to process-related events [45].

Upon state changes in the process engine, monitoring messages are published to all subscribers, which allows for (near) real-time monitoring of process instances. In addition, the process manager and WebSocket server enable the control of the execution environment, i. e., the control of process instances. Hence, users are provided with extensive means for interacting with CPS processes through various interactive clients. A timeout mechanism for process steps is integrated into PROtEUS to handle unpredictable and safety-critical behavior (e. g., errors, missing responses or missing data). We implemented a management system for PROtEUS on a mobile Android tablet and a tabletop device [49]. Simplified and easy to understand user interfaces will increase the user experience and provide access to complex backend systems (e. g., process management systems) even for non-expert users.

7.5 Synchronization of worlds and self-healing

As shown in Case Study 2 (cf. Sect. 6.2), PROtEUS provides the basic components for implementing a first working prototype of using additional sensor data to detect and repair inconsistencies between the execution of the virtual process instance and the state of the actual physical world as part of a MAPE-K loop subprocess. Using EPL statements as patterns/event conditions in the *Analyze* phase is a solution for the direct evaluation of event streams in correlation with the real-world effects of process execution. However, the additional consideration of more comprehensive contextual data and historical data also requires more sophisticated means of persisting and correlating events, e. g., in databases and ontologies, as well as more advanced formalisms for defining event and context conditions, e. g., based on logic or graphs. The *Plan* phase can also be enhanced by various related approaches considering more advanced planning algorithms and adaptive workflows [12,15,37,42,47]. A completely process-driven approach of implementing the MAPE-K feedback loop may lead to an increase in autonomy in process management due to the capability of self-adaptation of processes to unanticipated situations and errors as well as to emergent behavior.

7.6 Summary of discussion

The PROtEUS process execution system integrates various components for handling the specific properties of processes within CPS beyond the state of the art. PROtEUS discusses a model-driven architecture of a CPS control system based on processes, services and events, which consists of established concepts and components. Specialized solutions lack the capability of handling the identified set of CPS properties: event abstraction and processing, automated data flow, dynamic service invocation, ubiquitous human access, and the synchronization of worlds and self-healing regarding cyber-physical consistency (ACID-R). The case studies discussed in the previous section show that PROtEUS and the underlying process meta-model meet these requirements as a proof of concept and, therefore, provide a basis for the execution of interactive and dynamic processes in CPS. PROtEUS is able to bridge the gap between hardware-related processes and high-level human-centered workflows by integrating low-level sensor events and data as well as interactions and human tasks into model-based processes. The discussion in this section shows that by using PROtEUS, an increase in autonomy for the execution of processes in dynamic cyber-physical environments can be reached.

8 Related work

Cyber-physical systems research relates to a multitude of new and established research fields. The challenges for process execution in CPS identified in Sect. 3 have each been addressed to some extent by various related research. We will discuss related approaches with respect to the challenges in the following sections.

8.1 Event abstraction and processing

Barros et al. conducted an intensive investigation of the expressivity and semantics of BPMN and BPEL with respect to the concept of events [4]. Their results show that both process languages only support a limited set of common patterns for complex events in business processes. In particular, the integration of context events, inter-event relations and the simultaneous consumption of multiple events is very limited in BPMN and BPEL, which is why we decided to include more sophisticated event abstractions and processing concepts. Current process execution environments focus on executing high-level service-based business processes and, therefore, are only able to handle certain amounts of abstract events. Sungur et al. propose a meta-model extension for BPMN to support the modeling and processing of wireless sensor networks [54]. This approach still comes with significant modeling costs and limited scalability, which limits its suitability for large-scale CPS. In [44], Schiefer et al. present a framework for defining event-triggered rules for sensing and responding to business situations. In [56], Tuysuz et al. discuss a framework for interactive mobile workflows also integrating sensor and user events as well as Web services. This approach enables event-based behavior in workflows on a basic level. As we focus on executing workflows in event-driven CPS, we need more elaborate methods for the integration and processing of a large number of sensors and events from various sources. Therefore, we integrate the concept of events and CEP on different abstraction levels into the process execution environment.

Bülow et al. and Hermosillo et al. propose approaches for using CEP to monitor business process execution based on information from heterogeneous event logs [11,23]. These works do not consider external sensor data, but will be interesting for our future work with respect to the integration of process log data into PROtEUS. Wombacher [64] discusses an approach for correlating sensor and workflow data for observing the physical effects of business process activities and dealing with uncertainties in sensor events. We primarily focus on using sensor data as is for situation and context recognition in order to generate high-level events within processes. However, the integration of Wombacher's approach will be a part of our future work to also compensate for defective sensor readings. In [32] and [19], the authors propose extensions to BPMN and BPEL to model additional event sources with a special focus on Internet of things devices and sensors. While these approaches are first steps toward the integration of real-world physical data into business processes, using these extension is very costly considering the modeling complexity for a large number of sensors as every event source has to be considered individually. We decided to apply CEP mechanisms based on the definition of EPL patterns to enable the integration of low-level events into processes on arbitrary levels of granularity—from single-sensor instances to large networks of virtual and physical event sources. In analogy to the work of Baumgraß et al. [5], we use annotated process activities to define the EPL patterns to be evaluated by a CEP engine.

8.2 Automated data flow

Closely related to the handling of events is the formalization, abstraction and automated processing of data in processes. In contrast to the limited possibilities of formalizing the flow of data in current workflow languages (e. g., BPMN [63]), we see a strong need for expressing complex flows of typed data within high-level processes inside CPS. Advanced mechanisms enabling the composition of and operation on complex data for formalizing high-level process data flow are described by Montagnat et al. [33]. We use these approaches for the handling of data flow between the tasks of high-level CPS processes and mapping to low-level service invocations in a more sophisticated way and to integrate data from sensors and others sources into processes. The workflow language YAWL [58] allows for the definition of typed input and output data within various scopes of tasks of a workflow. We extend this data modeling by mappings, (de)compositions and replications of high-level data to access and distribute data on the process level. On the other hand, it is not feasible to define events and data flow for every low-level sensor and actuator of a CPS in a high-level process model. In contrast to workflow languages only enabling the modeling of single data instances, we use queries and pattern matching for the processing of larger amounts low-level CPS data.

8.3 Dynamic service invocation

Various service-based approaches for integrating heterogeneous dynamic resources into workflows including the aspects of service description [7,50], discovery [34], composition [39,50] and dynamic assignment of resources [7,25] already exist. We also apply a service-based approach for the integration of actuators and software components into our process execution environment. In [35], the authors propose a workflow system supporting long-running transactions for handling the fluctuation of available resources during process execution. As this work addresses an important issue for CPS–the dynamics of components–we incorporate concepts from it into our own execution system in combination with the semantic access layer concept described in [24]. PROtEUS enables the dynamic assignment of resources and services at runtime with the help of additional process annotations and semantic requests that are used for querying local or external service registries and knowledge bases.

8.4 Ubiquitous human access

In current process execution systems, the aspect of human interaction within processes and the access to suitable user interfaces is often realized through Web-based client or desktop applications. BPEL4People and WS-HumanTask [1] provide a framework for integrating user interactions into processes on a formal level. An approach for the introduction of implicit interaction into pervasive workflows is presented in [21]. Regarding service-based business processes, Web-based applications are often sufficient for providing necessary data and conducting high-level tasks. However, the increased focus on the users in CPS requires more sophisticated means for context-aware interactions within processes and with the process execution than it is possible with current BPM systems. Our aim is to provide ubiquitous access via multimodal user interfaces to enable real-time interactions and monitoring of processes from arbitrary interactive clients, independent of type, location or modality. As described in [56] and [35], this kind of ubiquitous access to workflows is of increasing importance for future human-centered information systems to increase the user experience and usability. Chakraborty and Hui propose an approach for enabling pervasive access to business processes using multiple devices and communication channels as well as considering additional context factors [14]. Adapting this approach, we transfer the concepts of automated generation of dialogs for human–process interactions based on model information to PROtEUS. As the development of new interaction techniques made significant progress in recent years, we aim at also supporting more modern interactive devices and additional modalities (e. g., audio and touch-based interactions), which can be selected automatically based on the users' current context.

8.5 Synchronization of worlds and self-healing

CPS workflow systems require a close coupling of physical context data and processes in order to link the effects of process execution to changes in the digital and physical world. In [65], Wombacher discusses ways to correlate physical objects and business workflows based on sensor data and changes within workflow states. He points out that the synchronization between process state changes and sensor data changes is an interesting challenge for CPS-like systems, but he does not provide a solution for this problem. Baumgraß et al. propose an event-driven process execution and monitoring system in [6] to relate event sources and processes. A workflow management system that uses situation recognition techniques based on sensor data from production machines is proposed by Wieland et al. in [62]. In order to adapt processes in case of the detection of errors and failures, process templates are executed as subworkflows depending of the specific type of error situations. We will investigate the application of approaches for situation recognition to enhance the *Analyze* phase for PROtEUS and also study the use of templates for adapting processes in case of errors. However, defining templates for a large number of possible error cases seems not to be feasible, which is why we could benefit from declarative modeling approaches at this point [41]. Abstracting from sensor data to context information in a broader sense, Herzberg et al. propose an event processing platform correlating external context information with business process events. This platform will be helpful to enrich our context models with additional external information and therefore improve the *Monitor* and *Analyze* phases within our implementation of the MAPE-K loop. In [60] Weidlich et al. discuss an approach for optimizing event patterns for process monitoring by using process model knowledge and event processing, which can be applied to automatically adapt the EPL patterns in successive MAPE-K iterations for CPS activities.

A more comprehensive framework for integrating IoT resources into process-aware information systems is proposed by Dar et al. [16]. It enables the integration of events from IoT devices on a single device granularity as well as the dynamic service replacement and message-based distributed execution. While this approach fulfills some of the identified requirements for CPS process execution, large-scale event processing, human interactions and the maintenance of cyber-physical consistency are possible only to a limited degree. The SmartPM system by Marella et al. is able to adapt to faulty process instances based on models of *expected reality* and *physical* reality and recover from a potential gap between these two worlds [31]. In particular, the adaptation algorithms are worth investigating and integrating into our MAPE-K loop approach for achieving a more generally applicable solution for self-healing processes. The combination with PROtEUS's event processing mechanisms may provide a more powerful integration and processing of heterogeneous cyber-physical event sources within business processes for CPS.

8.6 Summary of related work

With respect to established workflow engines, YAWL's engine–as an example of a widely used process engine in academia–also follows a Petri net-based approach for executing processes [59]. An example of a modern cloud-based process engine is the Cloud Process Execution Engine (CPEE) [30]. In order to meet all the challenges identified in Sect. 3, substantial extensions to both engines and additional components are necessary. We therefore decided to use a self-developed engine connecting all components and additions for executing CPS workflows as the core component of PROtEUS.

Looking at related work, we find that several aspects have already been solved in order to be applied within the context of process execution in cyber-physical systems. We integrate these existing solutions and components into our own process execution system. However, the combination of the identified CPS requirements (e. g., the high-level processing of events and data as part of the process execution, the integration of dynamic components, the handling of user interactions and the synchronization between the cyber and physical world) has not been addressed in a satisfying way by state-of-the-art process execution environments. Existing engines are only able to fulfill subsets of the requirements identified in Sect. 3. There is a considerable gap between hardware-oriented processes and human-focused workflows as well as between virtual world process execution and real-world process execution. The previous sections presented how PROtEUS addresses these issues by enabling a closer integration of humans and things into a process-aware information system for cyber-physical systems, which facilitates a tighter coupling of the digital world and physical world and allows for self-healing workflow execution based on MAPE-K feedback loops.

9 Conclusion

In this paper, we presented PROtEUS–an integrated system for process specification and execution in cyber-physical systems. The execution of processes in CPS poses new challenges that cannot be completely handled by current workflow engines. In particular, the data-centric and event-driven nature of CPS resulting from the combination of various low-level sensors, actuators, things and software components requires process execution systems that are able to integrate a heterogeneous set of resources on an active and reactive basis. For CPS, there is a need to handle the dynamics of resource constraint, loosely coupled devices as well as to support user interactions. Therefore, we developed a comprehensive process execution environment consisting of a core engine for executing model-based processes, a complex event processing engine for the integration and processing of low-level sensor data and a service invoker for calling on-site or external services. Services can be deployed on a local platform and found at runtime by means of an integrated registry or an external semantic knowledge base. Remote access for clients is provided through a WebSocket server enabling remote procedure calls and publish/subscribe access for process control, high-level process interaction and monitoring purposes. Processes are based on a component-based meta-model that allows for the composition of process steps and the definition of control and data flow. By combining these existing components into an integrated execution environment, we are able to cope with the challenges of using processes for controlling cyber-physical systems beyond the state of the art. The gap between hardware-related sensing and actuating in processes and human interactions in abstract workflows is reduced by PROtEUS. With PROtEUS being the base system, MAPE-K feedback loops can be implemented for processes and process activities to detect inconsistencies and restore consistency between virtual world process execution and real-world process execution by means of additional sensor information and process adaptations.

Regarding future work, we will conduct more comprehensive performance studies in order to evaluate the feasibility of our prototype within smart environments. To optimize the usage of available resources and reduce availability issues caused by a centralized approach, we will decentralize parts of the process execution system within a peer–super-peer network. The distribution of the process engine and management components across a hierarchical overlay network structure as described in [48] allows for creating scalable and resource-efficient process execution systems that can be applied to large-scale systems of systems (e. g., Smart Factories and Smart Cities). We will also introduce more sophisticated semantics for describing properties and relations of processes and resources as well as role-based inference mechanisms for assigning resources [50], finding services and reconfiguring processes at runtime. More sophisticated context models and adaptation mechanisms will help improving the implementation of the MAPE-K control loop, which will also be applied to check consistency and conformance during distributed process execution on peers. Finally, the linking and correlation of real-world sensor data, process event logs and MAPE-K loops will probably open up interesting new areas in the field of process monitoring extending existing concepts toward *Cyber-physical Process Mining*.

References

1. Agrawal, A., Amend, M., Das, M., Ford, M., Keller, C., Kloppmann, M., König, D., Leymann, F., Müller, R., Pfau, G., et al.: Web Services Human Task (ws-humantask). White Paper (2007)
2. Alliance, O.: Osgi Service Platform, Release 3. IOS Press, Amsterdam (2003)
3. Baheti, R., Gill, H.: Cyber-physical systems. Impact Control Technol. **12**, 161–166 (2011)
4. Barros, A., Decker, G., Grosskopf, A.: Complex events in business processes. In: Business Information Systems, pp. 29–40. Springer (2007)
5. Baumgraß, A., Botezatu, M., Ciccio, C.D., Dijkman, R., Grefen, P., Hewelt, M., Mendling, J., Meyer, A., Pourmirza, S., Hagen, V.: Towards a methodology for the engineering of event-driven process applications. In: Proceedings First International Workshop on Process Engineering, pp. 1–12 (2015)
6. Baumgrass, A., Ciccio, D., Claudio, C., Dijkman, R., Hewelt, M., Mendling, J.J., Meyer, A.A., Pourmirza, S.S., Weske, M.M., Wong, T.: Get controller and unicorn: event-driven process execution and monitoring in logistics. In: CEUR Workshop Proceedings (2015)

7. Bellur, U., Narendra, N.: Towards service orientation in pervasive computing systems. In: International Conference on Information Technology: Coding and Computing, 2005. ITCC 2005, vol. 2, pp. 289–295 (2005). doi:10.1109/ITCC.2005.280
8. Bernhardt, T., Vasseur, A.: Esper: Event Stream Processing and Correlation. In: ONJava. http://www.onjava.com/lpt/a/6955, OReilly (2007)
9. Brand, C., Gorning, M., Kaiser, T., Pasch, J., Wenz, M.: Development of High-Quality Graphical Model Editors. Eclipse Magazine (2011). http://www.eclipse.org/graphiti/documentation/files/EclipseMagazineGra. Accessed 1 Aug 2016
10. Broy, M., Cengarle, M., Geisberger, E.: Cyber-physical systems: imminent challenges. In: Calinescu, R., Garlan, D. (eds.) Large-Scale Complex IT Systems. Development, Operation and Management. Lecture Notes in Computer Science, vol. 7539, pp. 1–28. Springer, Berlin (2012). doi:10.1007/978-3-642-34059-8_1
11. Bülow, S., Backmann, M., Herzberg, N., Hille, T., Meyer, A., Ulm, B., Wong, T.Y., Weske, M.: Monitoring of business processes with complex event processing. In: Business Process Management Workshops, pp. 277–290. Springer (2013)
12. Burkhart, T., Loos, P.: Flexible Business Processes—Evaluation of Current Approaches. Proc. Multikonferenz Wirtsch. **2010**, 1217–1228 (2010)
13. Cao, J., Jarvis, S.A., Saini, S., Nudd, G.R.: Gridflow: workflow management for grid computing. In: Proceedings of the 3st International Symposium on Cluster Computing and the Grid, CCGRID '03, pp. 198–205. IEEE Computer Society, Washington, DC, USA (2003)
14. Chakraborty, D., Lei, H.: Pervasive enablement of business processes. In: Proceedings of Second IEEE Annual Conference on Pervasive Computing and Communications, PerCom, pp. 87–97 (2004). doi:10.1109/PERCOM.2004.1276848
15. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Comput. Sci. Res. Dev. **23**(2), 81–97 (2009)
16. Dar, K., Taherkordi, A., Baraki, H., Eliassen, F., Geihs, K.: A resource oriented integration architecture for the internet of things: a business process perspective. Perv. Mob. Comput. **20**, 145–159 (2015). doi:10.1016/j.pmcj.2014.11.005. http://linkinghub.elsevier.com/retrieve/pii/S1574119214001862
17. Dayarathna, M., Suzumura, T.B.: A performance analysis of system S, S4, and Esper via two level benchmarking. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8054, pp. 225–240 (2013). doi:10.1007/978-3-642-40196-1_19. http://www.scopus.com/inward/record.url?eid=2-s2.0-84882741975&partnerID=40&md5=853c6cc716722cc36b074bc762edc7d5
18. De Lemos, R., Giese, H., Müller, H.A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N.M., Vogel, T., et al.: Software engineering for self-adaptive systems: a second research roadmap. In: Software Engineering for Self-Adaptive Systems II, pp. 1–32. Springer (2013)
19. Domingos, D., Martins, F., Cândido, C., Martinho, R.: Internet of things aware WS-BPEL business processes: context variables and expected exceptions. J. UCS **20**(8), 1109–1129 (2014)
20. Fette, I., Melnikov, A.: The Websocket Protocol (2011). https://tools.ietf.org/html/rfc6455<UrlBlockedError.aspx. Accessed 1 Aug 2016
21. Giner, P., Cetina, C., Fons, J., Pelechano, V.: Implicit interaction design for pervasive workflows. Pers. Ubiquitous Comput. **15**(4), 399–408 (2011). doi:10.1007/s00779-010-0360-2
22. Grefen, P., de Vries, R.R.: A reference architecture for workflow management systems. Data Knowl. Eng. **27**(1), 31–57 (1998). doi:10.1016/S0169-023X(97)00057-8. http://www.sciencedirect.com/science/article/pii/S0169023X97000578
23. Hermosillo, G., Seinturier, L., Duchien, L.: Using complex event processing for dynamic business process adaptation. In: 2010 IEEE International Conference on Services Computing, pp. 466–473 (2010). doi:10.1109/SCC.2010.48. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5557204
24. Huber, S., Seiger, R., Kuehnert, A., Schlegel, T.: Using semantic queries to enable dynamic service invocation for processes in the internet of things. In: 2016 IEEE International Conference on Semantic Computing (ICSC), pp. 214–221 (2016). doi:10.1109/ICSC.2016.75
25. Kalasapur, S., Kumar, M., Shirazi, B.: Dynamic service composition in pervasive computing. IEEE Trans. Parallel Distrib. Syst. **18**(7), 907–918 (2007). doi:10.1109/TPDS.2007.1039
26. Kephart, J., Kephart, J., Chess, D., Boutilier, C., Das, R., Kephart, J.O., Walsh, W.E.: An Architectural Blueprint for Autonomic Computing. IBM, Armonk (2003)
27. Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., Trickovic, I.: WS-BPEL Extension for People-BPEL4People. In: Joint White Paper, IBM and SAP, vol. 183, p. 184 (2005)
28. Lee, E.: Cyber physical systems: design challenges. In: 2008 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), pp. 363–369 (2008). doi:10.1109/ISORC.2008.25
29. Luckham, D.: The Power of Events, vol. 204. Addison-Wesley, Reading (2002)
30. Mangler, J., Rinderle-Ma, S.: Cpee-cloud process execution engine. In: BPM (Demos)'14, pp. 51–51 (2014)
31. Marrella, A., Mecella, M., Sardina, S.: Smartpm: an adaptive process management system through situation calculus, indigolog, and classical planning. In: Principles of Knowledge Representation and Reasoning, pp. 1–10. AAAI Press, US (2014)
32. Meyer, S., Ruppen, A., Magerkurth, C.: Internet of things-aware process modeling: integrating IoT devices as business process resources. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7908, pp. 84–98 (2013). doi:10.1007/978-3-642-38709-8_6
33. Montagnat, J., Glatard, T., Lingrand, D.: Data composition patterns in service-based workflows. In: Workshop on Workflows in Support of Large-Scale Science, 2006. WORKS '06, pp. 1–10 (2006). doi:10.1109/WORKS.2006.5282350
34. Montagut, F., Molva, R.: Enabling pervasive execution of workflows. In: 2005 International Conference on Collaborative Computing: Networking, Applications and Worksharing, p. 10 (2005). doi:10.1109/COLCOM.2005.1651227
35. Montagut, F., Molva, R., Golega, S.T.: The pervasive workflow: a decentralized workflow system supporting long-running transactions. IEEE Trans. Syst. Man Cybern. C **38**, 319–333 (2008)
36. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: state of the art and research challenges. Computer **40**(11), 38–45 (2007)
37. Pesic, M., van der Aalst, W.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) Business Process Management Workshops. Lecture Notes in Computer Science, vol. 4103, pp. 169–180. Springer, Berlin (2006). doi:10.1007/11837862_18
38. Poovendran, R.: Cyber-physical systems: close encounters between two parallel worlds [point of view]. Proc. IEEE **98**(8), 1363–1366 (2010)
39. Qian, Z., Wang, Z., Xu, T., Lu, S.: A dynamic service composition schema for pervasive computing. J. Intell. Manuf. **23**(4), 1271–1280 (2012). doi:10.1007/s10845-010-0410-7
40. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3, p. 5 (2009)

41. Reijers, H.A., Slaats, T., Stahl, C.: Declarative modeling—an academic dream or the future for bpm? In: Business Process Management, pp. 307–322. Springer (2013)
42. Richly, S., Schmidt, S., Assmann, U.: A semantic-BDI-based approach to realize cooperative, reflexive workflows. In: Proceedings of the World Congress on Intelligent Control and Automation (WCICA), pp. 1680–1685 (2010). doi:10.1109/WCICA.2010.5554771
43. Scheer, A.W., Nüttgens, M.: ARIS Architecture and Reference Models for Business Process Management. Springer, Berlin (2000)
44. Schiefer, J., Rozsnyai, S., Rauscher, C., Saurer, G.: Event-driven rules for sensing and responding to business situations. In: Proceedings of the 2007 Inaugural International Conference on Distributed Event-based Systems, DEBS '07, pp. 198–205. ACM, New York, NY, USA (2007). doi:10.1145/1266894.1266934
45. Schlegel, T., Vidakovi, K., Dusch, S., Seiger, R.: Management of interactive business processes in decentralized service infrastructures through event processing. J. King Saud Univ. Comput. Inf. Sci. **24**(2), 137–144 (2012). doi:10.1016/j.jksuci.2012.03.001. http://www.sciencedirect.com/science/article/pii/S1319157812000134
46. Seiger, R., Huber, S., Schlegel, T.: Proteus: An integrated system for process execution in cyber-physical systems. In: Gaaloul, K., Schmidt, R., Nurcan, S., Guerreiro, S., Ma, Q. (eds.) Enterprise, Business-Process and Information Systems Modeling. Lecture Notes in Business Information Processing, vol. 214, pp. 265–280 (2015). doi:10.1007/978-3-319-19237-6_17
47. Seiger, R., Keller, C., Niebling, F., Schlegel, T.: Modelling complex and flexible processes for smart cyber-physical environments. J. Comput. Sci. (2014). doi:10.1016/j.jocs.2014.07.001
48. Seiger, R., Niebling, F., Schlegel, T.: A distributed execution environment enabling resilient processes for ubiquitous systems. In: 2014 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 220–223 (2014). doi:10.1109/PerComW.2014.6815205
49. Seiger, R., Struwe, S., Matthes, S., Schlegel, T.: A resilient interaction concept for process management on tabletops for cyber-physical systems. In: Yamamoto, S. (ed.) Human Interface and the Management of Information. Information and Knowledge in Applications and Services. Lecture Notes in Computer Science, vol. 8522, pp. 347–358. Springer (2014). doi:10.1007/978-3-319-07863-2_34
50. Shen, J., Yang, Y., Yan, J.: A p2p based service flow system with advanced ontology-based service profiles. Adv. Eng. Inf. **21**, 221–229 (2007)
51. Shi, J., Wan, J., Yan, H., Suo, H.: A survey of cyber-physical systems. In: 2011 International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1–6 (2011). doi:10.1109/WCSP.2011.6096958
52. Smirek, L., Zimmermann, G., Ziegler, D.: Towards universally usable smart homes-how can myui, urc and openhab contribute to an adaptive user interface platform. In: IARIA Conference, Nice, France, pp. 29–38 (2014)
53. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: Eclipse Modeling Framework. Pearson Education, New York (2008)
54. Sungur, C.T., Spiess, P., Oertel, N., Kopp, O.: Extending BPMN for wireless sensor networks. In: 2013 IEEE 15th Conference on Business Informatics, pp. 109–116 (2013). doi:10.1109/CBI.2013.24. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6642865
55. Talcott, C.: Cyber-physical systems and events. In: Software-Intensive Systems and New Computing Paradigms, pp. 101–115. Springer Berlin Heidelberg (2008)
56. Tuysuz, G., Avenoglu, B., Eren, P.: A workflow-based mobile guidance framework for managing personal activities. In: 2013 Seventh International Conference on Next Generation Mobile Apps, Services and Technologies (NGMAST), pp. 13–18. doi:10.1109/NGMAST.2013.12
57. Van Der Aalst, W.M.: Three good reasons for using a petri-net-based workflow management system. In: Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC96), pp. 179–201. Citeseer (1996)
58. Van Der Aalst, W., ter Hofstede, A.: YAWL: yet another workflow language. Inf. Syst. **30**(4), 245–275 (2005). doi:10.1016/j.is.2004.02.002
59. Van Der Aalst, W.M., Aldred, L., Dumas, M., ter Hofstede, A.H.: Design and implementation of the YAWL system. In: CAiSE, vol. 3084, pp. 142–159. Springer (2004)
60. Weidlich, M., Ziekow, H., Gal, A., Mendling, J., Weske, M.: Optimizing event pattern matching using business process models. IEEE Trans. Knowl. Data Eng. **26**(11), 2759–2773 (2014)
61. Weiser, M.: The computer for the 21st century. Sci. Am. **265**(3), 94–104 (1991)
62. Wieland, M., Schwarz, H., Breitenbucher, U., Leymann, F.: Towards situation-aware adaptive workflows: Sitopta general purpose situation-aware workflow management system. In: 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), pp. 32–37. IEEE (2015)
63. Wohed, P., van der Aalst, W., Dumas, M., ter Hofstede, A., Russell, N.: On the suitability of bpmn for business process modelling. In: Dustdar, S., Fiadeiro, J., Sheth, A. (eds.) Business Process Management. Lecture Notes in Computer Science, vol. 4102, pp. 161–176. Springer, Berlin (2006). doi:10.1007/11841760_12
64. Wombacher, A.: A-posteriori detection of sensor infrastructure errors in correlated sensor data and business workflows. In: Proceedings of the 9th International Conference on Business Process Management. BPM'11, pp. 329–344. Springer, Berlin (2011)
65. Wombacher, A.: How physical objects and business workflows can be correlated. In: Proceedings of 2011 IEEE International Conference on Services Computing, SCC 2011, pp. 226–233 (2011). doi:10.1109/SCC.2011.24

Ronny Seiger received his Diploma in computer science from Technische Universität Dresden in 2011. Since 2012, Ronny Seiger has been a Ph.D. student and member of the Software Engineering of Ubiquitous Systems and Software Technology groups at TU Dresden. His research interests include workflows and processes, Cyber-physical Systems, Internet of Things, robotics, distributed systems and software engineering. His main focus is on applying workflow technologies for automating processes in Cyber-physical systems.

Steffen Huber is a research assistant at the Junior Professorship in Software Engineering of Ubiquitous Systems within the DFG Research Training Group RoSI since November 2013. He received his Diploma in media computer science from Technische Universität Dresden in October 2013. His research interests include context-sensitive processes, role-based modeling, semantic technologies, Cyber-physical Systems, Internet of Things, and sensor data analysis. In his Ph.D. thesis, Steffen Huber investigates role-based modeling and execution of processes in the Internet of Things.

Thomas Schlegel is full professor at Karlsruhe University of Applied Sciences. His research interests focus on ubiquitous systems with a human (HCI) and technological (models, systems) perspective. He is head of the Institute of Ubiquitous Mobility Systems (IUMS) and holds the INIT endowed chair. As a professor in the faculty of Information Management and Media (IMM) he is one of the key professors in Transportation Management responsible for Information Technology. Before, he has been in research positions at different companies, Fraunhofer society, University of Stuttgart and as a Junior Professor at Technische Universität Dresden. With more than 80 publications and serving as editor, reviewer and PC member in different fields, he is strongly engaged in research as well as education with systems and process related topics since the start of his academic career.