A Case Study for Workflow-based Automation in the Internet of Things

Ronny Seiger, Uwe Aßmann
Software Technology Group
Technische Universität Dresden
D-01062 Dresden, Germany
Email: {ronny.seiger, uwe.assmann}@tu-dresden.de

Steffen Huber
Institute of Machine Tools and Control Engineering
Technische Universität Dresden
D-01062 Dresden, Germany
Email: steffen.huber@tu-dresden.de

***Abstract*—The application of workflow technologies in the context of Internet of Things (IoT) presents a young and vibrant research field. Introducing a dedicated workflow layer on top of typical software architectures to model and execute repetitive tasks among IoT devices, systems and services facilitates the flexibility, reuse, configuration and programming of processes in the IoT. In this work we present the *PROtEUS* IoT workflow management system (WfMS) and its application for automating typical processes in a Smart Home case study. We conduct various experiments to show the system's capabilities and suitability to be used as an IoT WfMS interacting with sensors, actuators, robots, humans and smart objects. PROtEUS integrates components for dynamic service selection, complex event processing, human interactions and self-adaptation, which makes it a WfMS able to cope with new IoT-related challenges.**

I. Introduction

Internet of Things (IoT) environments consist of a large number of interconnected heterogeneous devices. Developing applications and automation routines that involve multiple devices and entities–from sensors, actuators, embedded systems, mobile devices up to Cloud servers as well as humans and smart objects–currently requires advanced programming and software engineering skills. However, especially in the context of *Smart Homes*, users want to be provided with tools to easily configure and control their surrounding IoT appliances and to compose simple processes to automate repetitive tasks.

To facilitate this need, we propose the introduction of a dedicated *Workflow Layer* upon existing architectural software stacks for IoT usually consisting of a *Hardware Layer*, which is manipulated by specific drivers and control applications; a *Middleware Layer* to unify the heterogeneous hardware infrastructure and mediate calls to the respective devices; and a *Service Layer* to make the functionality remotely accessible (cf. Figure 1 [1]). Workflow technology provides at this point a flexible, reusable way of programming and orchestrating sequences of service calls and activities across device and system boundaries to automate repetitive tasks in IoT [2].

The application of business process technologies in the context of IoT is still a young and vibrant research field that has to cope with various new challenges [2]–[5]. With our work, we focus on providing an IoT workflow notation and execution system to include the entities of a complex IoT environment–multiple sensors and actuators, static and dynamic services, humans, smart objects–and describing and enacting interactions among these entities on the business process level. The respective worfklow management systems (WfMS) have to be able to deal with unexpected behaviour and dynamically adapt to new situations as these IoT processes also interact with the physical world, which is more unreliable and a source for new errors influencing the execution. We present a practical case study regarding the application of a specific WfMS for IoT [2] in the context of a *Smart Home* as an IoT environment. We briefly present the components of the WfMS and show how the system is applied to conduct various experiments in the smart home, including the interaction with sensors, actuators, robots, humans and smart objects.

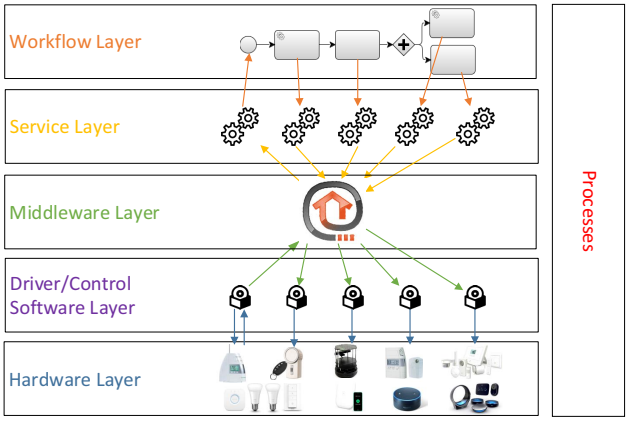

Fig. 1. Workflows on Top of Common Architectural Layers in IoT [1].

II. Workflow Management System for IoT

The *PROtEUS* system presented in [2] serves as the basic IoT WfMS to model and execute processes in our work as it proved to be a feasible workflow system for IoT [6], [7]. The formalism used to describe the following workflows is a more technical process description language developed to enable the modelling of all IoT entities–complex sensors, dynamic actuators/services, humans and smart objects–interacting with each other on the business process level [8]. Its main building blocks are high-level events triggered by specified low-level event (EPL) patterns, invocations to various types of static or

978-1-5386-6585-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICSA-C.2018.00011

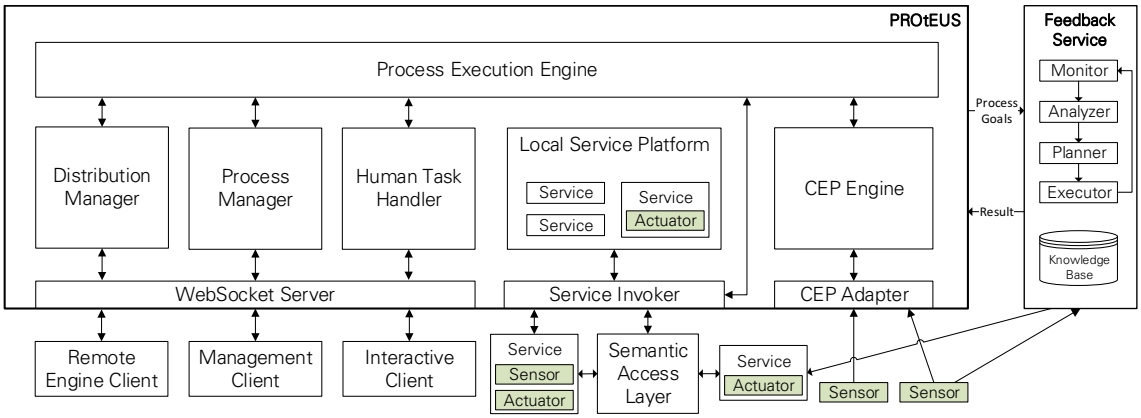

Fig. 2. Architecture of the PROtEUS Workflow Management System [2].

dynamic services, human tasks for manual activities, as well as logic elements to connect these process components to higher-level programs. Typed ports are used to define data and control flow among the process steps. In addition, criteria regarding the success and failure of a process step to be analysed within MAPE-K loops can be specified in goals and objectives [9]. Figure 2 presents an overview of PROtEUS's system architecture. The WfMS features the following components:

- *Process Execution Engine*: Executes process instances according to their underlying process models and communicates with all other components.
- *Process Manager*: Provides interfaces to manage, control and monitor the execution of processes.
- *CEP Engine*: Enables complex event processing (CEP) within multiple IoT sensor event streams to detect specific patterns defined in the corresponding process steps [2].
- *Local Service Platform*: Enables the local deployment and discovery of Web services.
- *Service Invoker*: Supports the invocation of various types of Web services to invoke IoT device functionality based on standard or proprietary protocols.
- *Human Task Handler*: Distributes manual tasks that are part of a process requiring human interactions.
- *Distribution Manger*: Enables the distributed execution of subprocesses via subcontracting on remote WfMS's [10].
- *Web Socket Server*: Enables the bi-directional communication and interaction with users and other instances of the WfMS via Pub/Sub and RPC.

In addition, PROtEUS interacts with the *Semantic Access Layer* (SAL) to dynamically discover and invoke services based on required functionality and context constraints defined in an ontology [11]. That way, the dynamic availability of mobile and resource-constraint IoT devices can be handled.

PROtEUS also interacts with the *Feedback Service* (FB), which implements a generic *MAPE-K* feedback loop [12] to enable self-adaptation of workflows with respect to predefined goals in case of errors and unanticipated situations [9]. The FB uses external sensor data specified in the goal related to the execution of a specific process step (Monitor) to detect success or error of the execution based on specific conditions defined in the goal (Analysis). The sensor data is described and stored in the knowledge base (K). Special compensation queries and a compensation repository can be used and extended to define and execute compensation strategies (Plan, Execute). That way, the execution system is able to verify the successful task execution or to detect undesired workflow effects and remedy errors via self-adaptation–in our cases to replace the process resources and re-execute the respective process steps.

III. Smart Home Scenario Processes

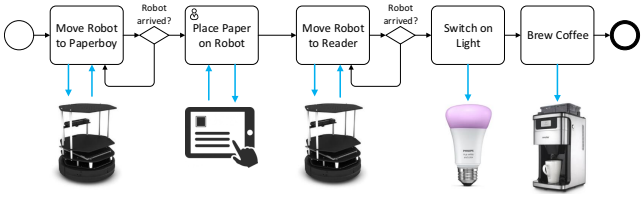

Fig. 3. Abstract Morning Routine Process.

In the following we describe two complex processes within a smart home that serve as examples to illustrate the interaction of the individual components of the WfMS.

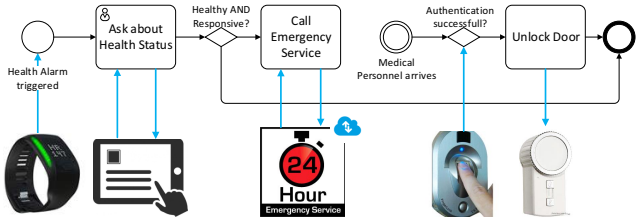

Fig. 4. Abstract Emergency Process.

A. Morning Routine Process

Fig. 3 shows the *Morning Routine* process aimed at increasing the comfort of the resident by automating parts of the

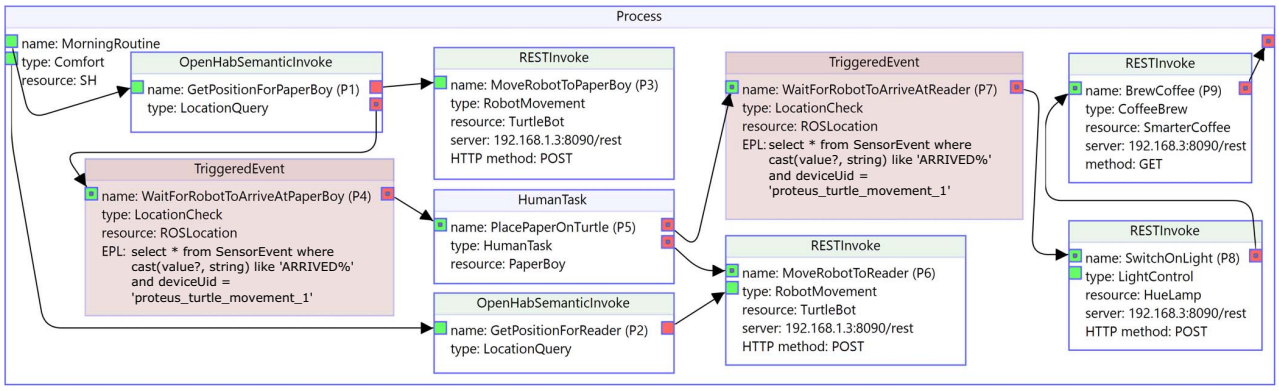

Fig. 5. Detailed Morning Routine Process.

morning routine in the smart home [13]. First, a mobile service robot is sent to the paper delivery boy at the front door. Once the robot arrives, it emits an event, which triggers a human task to be sent to the delivery boy's tablet device. After the paper is placed on the robot and the human task is confirmed, the robot drives to the resident. Upon the robot's arrival, the light in the kitchen is switched on and subsequently, the coffee maker is triggered to start brewing coffee.

B. Emergency Process

Fig. 4 shows the *Emergency* process aimed at health monitoring of a resident and providing medical assistance in an ambient assisted living (AAL) setting [14]. The process triggers an emergency event once the WfMS detects a sudden drop in the resident's vital signs monitored by a fitness tracker. The user is asked to confirm the health status using a mobile device. In case of a negative response or a timeout, an emergency service is called. Once the medics authenticate themselves at the door, it is opened automatically.

IV. Evaluation

The goal of this paper is to provide a comprehensive evaluation of a WfMS for IoT in a smart home case study. To show its feasibility with respect to fulfilling new requirements that IoT environments pose [2], we conduct various practical experiments related to the scenario processes.

A. Experiments

The following experiments were conducted in a smart home lab environment. We use several sensors and actuators from multiple vendors, service robots and mobile Android devices, which are all connected to the OpenHAB smart home middleware [11]. A central control computer (Intel Core i7, 4x3.1 GHz, 8 GB RAM, 32 GB SSD, 2 TB HDD, Ubuntu Linux 14.04) runs the middleware as well as the PROtEUS WfMS[1] including Feedback Service[2] and SAL[3] described in Sec. II. The processes are modelled using a more technical process notation for IoT and its graphical representation [8].

[1]https://github.com/IoTUDresden/proteus
[2]https://github.com/IoTUDresden/feedback-service
[3]https://github.com/IoTUDresden/openhab2-addons

B. Morning Routine Process

TABLE I
Execution Times for the Morning Routine Process.

ID	Process Step	Duration (in ms)
P1	GetPositionForPaperBoy	34
P2	GetPositionForReader	42
P3	MoveRobotToPaperBoy	35
P4	WaitForRobotToArriveAtPaperBoy	23631
P5	PlacePaperOnTurtle	13218
P6	MoveRobotToReader	27
P7	WaitForRobotToArriveAtReader	50859
P8	SwitchOnLight	13
P9	BrewCoffee	15586
	MorningRoutine	103.334

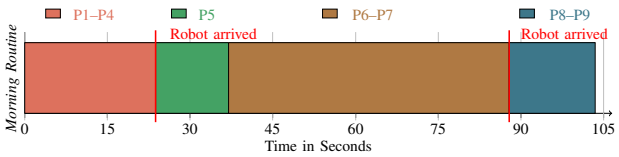

Fig. 6. Instance of the Morning Routine Process over Time.

This experiment is conducted to show the interaction of the PROtEUS WfMS with IoT sensors and actuators as well as humans. Fig. 5 presents the detailed *Morning Routine* process to be executed. A service robot is sent to the paper delivery boy via a RESTful service call to the robot's Web server *(P3)*. The REST call is parametrised with the current position of the paper delivery boy retrieved by executing a semantic process step, which contains a *SPARQL* query for the SAL's knowledge base *(Step P1)* [11]. The robot's arrival is detected based on a specific EPL pattern by the CEP engine, which listens for the robot's "arrived" event in a special process step *(P4)* [2]. Following, the paper delivery boy is asked to place the paper on the robot via a *HumanTask (P5)* and confirm this task on his tablet device. The robot is then sent to the reader *(P6)* based on the result of the executed *SPARQL* query to retrieve the

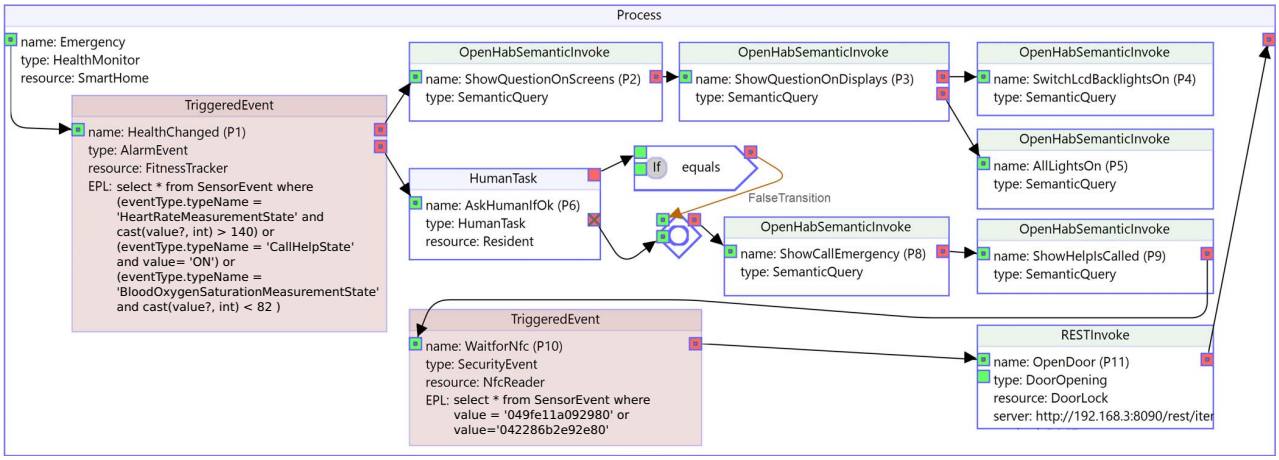

Fig. 7. Detailed Emergency Process.

Listing 1. SPARQL Query for Retrieving all Dimmer Switches.

```
SELECT ?func ?dimmer
WHERE {
 ?dimmerClass rdfs:subClassOf* dogont:
     DimmerSwitch .
 ?onClass rdfs:subClassOf* dogont:OnCommand .
 ?dimmer rdf:type ?dimmerClass .
 ?dimmer dogont:hasFunctionality ?func .
 ?func dogont:hasCommand ?cmd .
 ?cmd rdf:type ?onClass . }
```

reader's current position *(P2)*. The robot's arrival is detected via a complex event as defined in the *EPL* pattern *(P7)*. Finally, the light switch and coffee machine are triggered using RESTful service calls to the middleware *(P8, P9)*.

Results: The execution times of the individual process steps for an instance of the *Morning Routine* process are shown in Table I and the corresponding flow of activations over time in Fig. 6. We see that purely virtual workflow tasks, e. g., the retrieval of specific locations from the SAL (*P2, P3*); the invocation of services to send a robot to specific locations (*P3, P6*); or to switch on the light (*P8*) are executed within only a few milliseconds. The actual physical tasks, e. g., the driving of the robots and publishing the corresponding events (*P4, P7*); the confirmation of the human task (*P5*); and the brewing of coffee (*P9*) take significantly longer as they rely on interactions with the physical world. The service invocations related to process steps P3, P6 and P8 rely on asynchronous Web services, which is why their execution times are relatively short compared to the synchronous call in process step P9 waiting for a response from the Web service. However, to confirm the execution of the asynchronous calls, we need to listen for the corresponding events from the robots with the help of the CEP engine in dedicated process steps (P4, P7).

C. Emergency Process

This experiment is conducted to show the interaction of the PROtEUS WfMS with humans, robots, sensors, actuators and the dynamic service discovery via the SAL. Fig. 7 presents the detailed *Emergency* process to be executed. A fitness tracker worn by the resident sends its data to the CEP engine, which

TABLE II
EXECUTION TIMES FOR THE EMERGENCY PROCESS.

ID	Process Step	Duration (in ms)
P1	HealthChanged	27916
P2	ShowQuestionOnScreens	181
P3	ShowQuestionOnDisplays	47
P4	SwitchLcdBacklightsOn	57
P5	AllLightsOn	111
P6	AskIfHumanOk	15046
P7	OR	7
P8	ShowCallEmergency	131
P9	ShowHelpIsCalled	55
P10	WaitForNfc	21213
P11	OpenDoor	20
	Emergency	64389

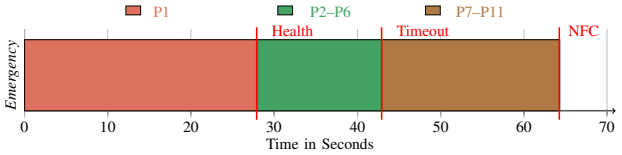

Fig. 8. Execution of an Instance of the Emergency Process over Time.

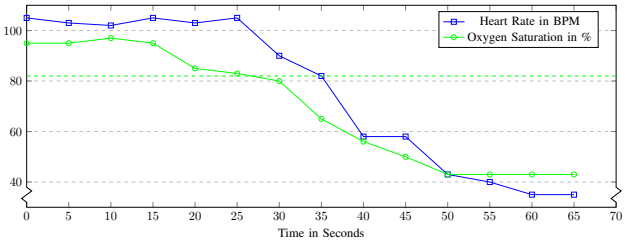

Fig. 9. Simulated Data of the Fitness Tracker.

evaluates the data based on an *EPL* pattern. An *AlarmEvent* is triggered upon the detection of a critical health situation *(P1)*, i. e., if the heart rate is above 140 BPM or the oxygen saturation is below 82 %. Following, several process steps are executed to show warning messages on all displays *(P2, P3)* and switch on all lights *(P4, P5)* to get the resident's attention. The SPARQL query used in process step P5 to retrieve and activate all dimmer switches is shown in Listing 1. In parallel, a human task is sent to the resident to confirm the well-being *(P6)*. In case of a negative response or a 15 seconds timeout, an emergency call is sent and messages are shown regarding the emergency call *(P8, P9)*. The Kodi media player and all displays are used to show the corresponding messages and inform the resident. The process engine then waits for an event regarding the authentication of medical personal detected by the CEP engine, which listens for specific identifiers of NFC tags emitted by an NFC reader *(P10)*. After the successful authentication, the door is unlocked using a RESTful service call to the door lock actuator *(P11)*.

Results: The execution times of the individual process steps for an instance of the *Emergency* process can be found in Table II and the corresponding flow of activations over time in Fig. 8. Similar to the results of the *Morning Routine* process, the durations of the purely digital computations are within the magnitude of a few milliseconds *(P3, P4, P5, P7, P9, P11)* up to approx. 200 ms for the execution of complex queries by the SAL *(P2, P8)*. Compared to that, the physical tasks and events take significantly longer as they rely on interactions with humans. The health alarm *(P1)* is triggered after approx. 28 s based on artificial sensor data related to the oxygen levels from the fitness tracker (cf. Fig. 9). As there was no response to the human task *(P6)*, the process timed out and continued automatically after 15 s. The NFC reader produced an authentication event after approx. 21 s that was detected by the CEP engine *(P10)*.

D. Coffee Maker Process

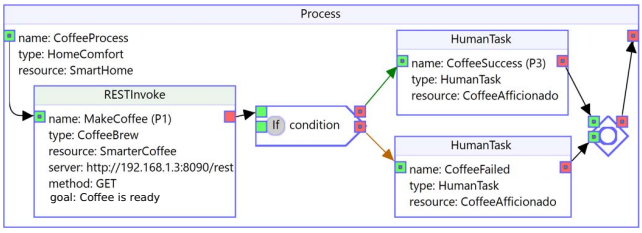

Fig. 10. Detailed Coffee Maker Process.

This experiment is conducted to show the interaction of the PROtEUS WfMS with IoT sensors, actuators and humans, and to use the Feedback (FB) Service to verify the successful task execution with the help of an additional sensor in the MAPE-K feedback loop. Fig. 10 presents the detailed *Coffee Maker* process to be executed. The command to start brewing one cup of coffee is sent via a RESTful service call to the middleware *(P1)*. The process step includes a goal "Coffee is ready" for the verification of the process step as shown in Listing 2 [9]. The goal defines an objective stating that the coffee is brewed successfully when an external infra-red temperature sensor measuring the temperature of the coffee cup (*contextPaths*) detects a temperature above 37°C within 3 minutes after starting the brewing subprocess (*satisfiedCondition*). If this temperature is not reached within 180 s, an error is assumed (*compensationCondition*). The goal is passed to the FB Service (cf. Fig. 2) where the associated temperature is monitored and analysed continuously according to the objective. The FB Service's result is transferred back to the WfMS where a human task is activated to either report the failure or success of the process *(P3)*.

Listing 2. Goal and Objective for MakeCoffee Process Step.

```
"MakeCoffee" : {
"name":"Coffee is ready",
"objectives":[ {
 "name":"coffee temperature > 37 degrees within
      3 minutes",
 "satisfiedCondition":"#coffeeTemp > 37",
 "compensationCondition":"#objective.created.
     isBefore(#now.minusSeconds(180))",
 "contextPaths":[
  "MATCH (ctemp {name: '
      State_tinkerforge_irTemp_irTemp_1'})-[:
      hasStateValue]->(value)",
  "WHERE toFloat(value.realStateValue)>0",
  "RETURN toFloat(value.realStateValue) AS
      coffeeTemp, id(ctemp) AS stateId"
 ]  }  ]  }
```

TABLE III
EXECUTION TIMES FOR THE COFFEE PROCESS.

ID	Process Step	Duration (in ms)
P1	MakeCoffee	114.940
P2	IF	38
P3	CoffeeSuccess	35645
P4	OR	9
	CoffeeProcess	150.632

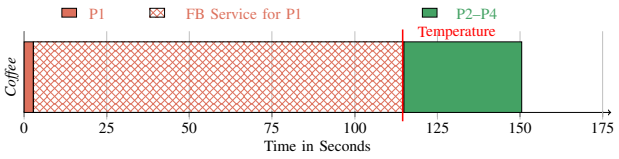

Fig. 11. Execution of an Instance of the Coffee Process over Time.

TABLE IV
EXECUTION OF THE INDIVIDUAL MAPE PHASES BY THE FEEDBACK SERVICE FOR THE MAKECOFFEE STEP.

Metrics \ Phase	M	A	P	E	Loop	FB Service
Iterations (#)	–	148	–	–	148	1
Duration (ø in ms)	–	6	–	–	1023	112.135

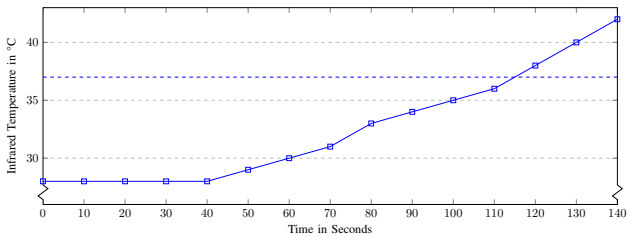

Fig. 12. Values from the Infra-red Temperature Sensor over Time.

Results: The execution times of the individual process steps for an instance of the *Coffee Maker* process can be found in Table III and the corresponding flow of activations over time in Fig. 11. The service call to the coffee maker (*P1*) was supervised by the Feedback (FB) Service in accordance with the specified goal (cf. Listing 2). Table IV shows the number of iterations of the MAPE-K feedback loop and its durations. During these executions, 148 sensor values provided by the additional infra-red temperature sensor were analysed until the specified threshold was reached after approx. 115 s (cf. Fig. 12) and the process step executed successfully. The monitoring of sensor values is performed continuously. There was no need to initiate the planning or execution phases of the feedback loop as the condition for successful task execution (temperature over 37°C) was fulfilled positively within the defined time frame of 180 seconds. It took the user approx. 36 s to confirm the successful brewing via the respective human task (*P3*).

E. Robot Navigation Process

This experiment is conducted to show the interaction of the PROtEUS WfMS with service robots and to use the Feedback Service to verify the successful task execution based on additional sensors in a MAPE-K feedback loop. The service robot used in this experiment suffers from frequent problems with its internal SLAM-based navigation system. It often gets stuck, assumes incorrect positions on its internal map, or reports its arrival despite not reaching the physical location. We therefore use an external more accurate location tracking system to compare its internal position with the external location sensors. Fig. 13 presents the detailed *Robot Navigation* process to be executed. First, the current position of the resident is retrieved via the SAL *(P1)*. Then, a process loop (*MoveTurtleAndCheckForSuccess*) is triggered to move the robot to the resident using a MAPE-K enabled process step *(MoveTheTurtle)* with a corresponding goal *Robot Position*. The process step is executed successfully if the robot emits an "arrived" event and reached specific coordinates in the room that can be verified by the FB Service using sensor data from the external tracking system. The process loop and the process then terminate. If the external coordinates indicate a wrong position of the robot assuming an incorrect location, the FB service will trigger the repetition of the process loop and with that the repetition of the *MoveTheTurtle* process step.

Results: The execution times of the process steps for an instance of the *Robot Navigation* process can be found in

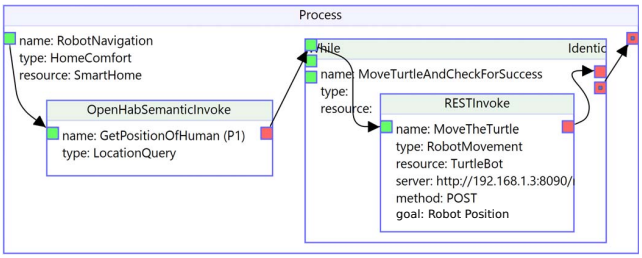

Fig. 13. Detailed Robot Navigation Process.

TABLE V
EXECUTION TIMES FOR THE ROBOTNAVIGATION PROCESS.

ID	Process Step	Duration (in ms)
P1	GetPositionOfHuman	1079
P2	MoveTheTurtle (1)	22924
P3	MoveTheTurtle (2)	7530
P4	MoveTheTurtle (3)	10065
P5	MoveTheTurtle (4)	7759
	MoveTurtleAndCheckForSuccess	48278
	RobotNavigation	49357

Fig. 14. Execution of the RobotNavigation Process over Time.

TABLE VI
EXECUTION OF THE INDIVIDUAL MAPE PHASES BY THE FEEDBACK SERVICE FOR THE MOVETHETURTLE STEP.

ID	Process Step	#M	#A	#P	#E	Duration (in ms)
P2	MoveTheTurtle (1)	–	17	1	1	21084
P3	MoveTheTurtle (2)	–	6	1	1	7202
P4	MoveTheTurtle (3)	–	9	1	1	10028
P5	MoveTheTurtle (4)	–	6	0	0	7333

Table V and the corresponding flow of activations over time in Fig. 14. The retrieval of the human's position via the SAL (*P1*) for this run takes longer than similar process steps of other processes due to some internal configuration and starting issues with the middleware services. In total, the process loop sending the robot to the human's position and verifying its success based on the external coordinates was executed four times (*P2–P5*), i. e., the robot got stuck, cancelled the navigation or reported wrong internal coordinates due to obstacles in the room. Table VI shows the executions of the MAPE-K feedback loop by the FB Service for the individual repetitions of the *MoveTheTurtle* process step instances. Monitoring of the external and internal robot location data was performed continuously. The number of sensor data analysed depends on the duration of the process step until the robot's internal system reports an "arrived" event. For P2–P4 the Feedback Service determined a mismatch between the robot's internal system

and the external more accurate tracking system when the robot published the respective "arrived" events. For this scenario, the planner derived the repetition of the *MoveTheTurtle* process step as a suitable compensation strategy based on a predefined compensation query on a compensation repository [9]. The executor of the MAPE-K loop terminates the control loop and reports the unsuccessful execution back to the workflow engine, which triggers the *MoveTurtleAndCheckForSuccess* process loop to be repeated. For P5, the robot's internal coordinates and "arrived" event can be confirmed by the FB Service using the external coordinates, which leads to the process instance terminating successfully. Fig 15 shows the path of the robot on its internal map to get from the starting point to the human as controlled by the WfMS.

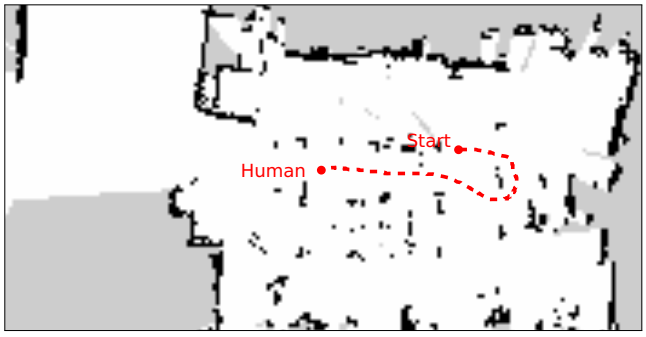

Fig. 15. Path of the Robot for the Robot Navigation Experiment.

V. Discussion

With the case study, we are able to show and verify the behaviour of the IoT WfMS in real world experiments. From the use case set in the context of a smart home, we derived requirements for a WfMS used to automate processes that involve resource-constraint and fluctuating devices, heterogeneous networks of sensors and other event sources, smart objects as well as humans as process resources. The components and workflow language of the WfMS were chosen to fulfil these requirements–CEP for complex sensor processing, human tasks for human interaction, SAL for dynamic service selection, and Feedback Service for process verification and self-adaptation. By linking the workflow and task executions with associated sensor data, we are able to verify the correct interactions of the WfMS with the IoT entities as well as with the physical environment and vice versa. The workflow language provides a suitable formalism to model all these aspects and interactions on the business process level.

The WfMS shows near real-time performance regarding purely virtual task executions with execution times in the order of a few milliseonds to a 100 ms per task. The executions of *cyber-physical* tasks take significantly longer due to the interactions with the physical world and physical process resources (e. g., robots or humans) performing the necessary tasks in the real world. By using additional sensors, we are able to monitor and verify these executions in the physical world. The overall performance of the PROtEUS WfMS shows its feasibility as a WfMS for IoT in the context of smart homes. More real-time and safety-critical tasks and processes in the smart home and other smart spaces have to be investigated with respect to performance and physical interactions of the WfMS, though. Additional experiments in a smart home showing the self-adaptation for smart lighting processes can be found in [9] and for distributed process execution on mobile robots in [10].

VI. Related Work

A comprehensive survey discussing business process management for IoT can be found in [3]. Domingos et al. propose an extension of the *WS-BPEL* workflow language to support context variables and sensors as well as communication paradigms (request/reply, pub/sub) for IoT [15]. In [16] Glombitza et al. discuss an approach of executing WS-BPEL processes on IoT sensor nodes and wireless sensor networks (WSN) based on a service-oriented architecture. Tranquilini et al. present an extension of the wideley used process notation *BPMN 2.0* to model and program WSN's based on business processes [13]. Extensions of BPMN 2.0 to represent IoT devices, IoT services and things in business processes as new resources and lanes are discussed by Meyer et al. [17], [18]. To be able to handle more complex streams of sensor data in real-world business processes, Baumgraß et al. propose to use a CEP engine as component of the WfMS [19]. These approaches discuss concepts regarding new IoT-related entities that are a subset of challenges that we address with the PROtEUS WfMS. They mostly focus on new concepts without providing extensive evaluations based on real world IoT data.

A more complex WfMS to be used in a mobile context to support mobile worker with executing tasks in business processes is presented by Giner et al. [20]. Similarly, a flexible light-weight workflow system to be used on mobile devices for mobile task execution is discussed by Pryss et al. in [21]. Dar et al. propose the use of business processes in the context of smart homes and AAL to provide users with assistance in [14]. Their WfMS supports distributed execution, dynamic service discovery and replacement as well as event-based communication. A device-to-device-based WfMS for industrial IoT is shown by Mass et al. in [22]. The authors discuss how to migrate decentralized business processes between unreliable IoT devices in case of device failures or disconnects. With *SitOPT* Wieland et al. present an adaptive WfMS that is able to recognize situations based on sensor data and situation templates and to adapt processes in case of failures by replacing the respective resources or workflow tasks [23]. The *SmartPM* workflow system by Marrella et al. is capable of detecting mismatches between the expected outcome of the workflow execution and the actual effects based on sensor data [24]. Using a smart planning algorithm and dynamic goal refinement based on artificial intelligence, the authors are able to derive suitable compensation strategies to compensate these mismatches and unanticipated errors [25].

The PROtEUS WfMS presented in this work is designed to address multiple challenges that arise with the complex behaviour and interactions of IoT environments [6], [9]. With

complex sensors, actuators, robots, smart objects and humans as well as dynamic services involved, the IoT workflows shown in our case study require a flexible, self-adaptive and interactive WfMS including a corresponding modelling notation that go beyond the requirements and aspects addressed by the related works discussed in this section. Compared to these approaches that often only provide a discussion of new concepts, a proof-of-concept implementation or a short evaluation based on simulated data, we show the behaviour of the IoT WfMS in a real world context based on practical experiments and actual data in our case study. Our proposed workflow language provides a more advanced formalism to model the complex interactions among entities in an IoT environment on the business process level as high level programs, which are relatively easy to model. By linking the workflow executions with their real world effects and vice versa, we are able to implement flexible active and reactive IoT processes and verify the correct interaction of all WfMS components also with respect to the physical dimension of the IoT environment.

VII. Conclusion

We presented the *PROtEUS* WfMS specifically designed to cope with new challenges that arise with applying process technologies to IoT environments. The WfMS features components to process complex sensor streams from various IoT devices; to dynamically select IoT services at runtime based on availability, functionality and context factors; to interact with complex actuators and humans; and to self-adapt in case of unanticipated behaviour. The case study that we conducted in a smart home shows the successful execution of complex workflows involving sensors, actuators, robots, humans and smart objects in various experiments. By linking the individual task and workflow executions to the respective effects in the physical and virtual worlds with the help of additional sensor data, we are able to verify the correct executions and behaviour of the WfMS and involved IoT entities. With our smart home experiments, we are able to show the feasibility of PROtEUS as a suitable IoT WfMS. Regarding future work, we will conduct further case studies in the context of *Smart Factories* and *Smart Buildings*, where new requirements especially related to safety and real-time have to be fulfilled.

Acknowledgment

This research has received funding under the grant number 100268299 (“CyPhyMan”) by the European Social Fund (ESF) and the German Federal State of Saxony. Kudos to André Kühnert for helping with the experiments.

References

[1] S. Bandyopadhyay, M. Sengupta, S. Maiti, and S. Dutta, “Role of middleware for internet of things: A study,” *Int. Journal of Computer Science and Engineering Survey*, vol. 2, no. 3, pp. 94–105, 2011.

[2] R. Seiger, S. Huber, and T. Schlegel, “Toward an execution system for self-healing workflows in cyber-physical systems,” *Software & Systems Modeling*, pp. 1–22, 2016.

[3] C. Chang, S. N. Srirama, and R. Buyya, “Mobile cloud business process management system for the internet of things: a survey,” *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, p. 70, 2016.

[4] J. Mendling, B. Baesens, A. Bernstein, and M. Fellmann, “Challenges of smart business process management: An introduction to the special issue,” *Decision Support Systems*, 2017.

[5] C. Janiesch, A. Koschmider, M. Mecella, B. Weber, A. Burattin, C. Di Ciccio, A. Gal, U. Kannengiesser, F. Mannhardt, J. Mendling *et al.*, “The internet-of-things meets business process management: Mutual benefits and challenges,” *arXiv:1709.03628*, 2017.

[6] R. Seiger, S. Huber, and P. Heisig, “Proteus++: A self-managed iot workflow engine with dynamic service discovery,” in *Central European Workshop on Services and their Composition (ZEUS)*, 2017.

[7] S. Huber, R. Seiger, A. Kühnert, and T. Schlegel, “A context-adaptive workflow engine for humans, things and services,” in *Proc. of the International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. New York, NY, USA: ACM, 2016, pp. 285–288.

[8] R. Seiger, C. Keller, F. Niebling, and T. Schlegel, “Modelling complex and flexible processes for smart cyber-physical environments,” *Journal of Computational Science*, vol. 10, pp. 137 – 148, 2015.

[9] R. Seiger, S. Huber, P. Heisig, and U. Aßmann, “Toward a framework for self-adaptive workflows in cyber-physical systems,” *Software & Systems Modeling*, Nov 2017.

[10] R. Seiger, S. Herrmann, and U. Aßmann, “Self-healing for distributed workflows in the internet of things,” in *IEEE International Conference on Software Architecture (ICSA 2017) Workshops*, 2017.

[11] S. Huber, R. Seiger, A. Kuehnert, V. Theodorou, and T. Schlegel, “Goal-based semantic queries for dynamic processes in the internet of things,” *Intl. Journal of Semantic Computing*, vol. 10, no. 02, pp. 269–293, 2016.

[12] J. Kephart, D. Chess, C. Boutilier, R. Das, and W. E. Walsh, “An architectural blueprint for autonomic computing,” *IBM*, 2003.

[13] S. Tranquillini, P. Spieß, F. Daniel, S. Karnouskos, F. Casati, N. Oertel, L. Mottola, F. Oppermann, G. Picco, K. Römer *et al.*, “Process-based design and integration of wireless sensor network applications,” *Business Process Management*, pp. 134–149, 2012.

[14] K. Dar, A. Taherkordi, H. Baraki, F. Eliassen, and K. Geihs, “A resource oriented integration architecture for the Internet of Things: A business process perspective,” *Pervasive and Mobile Computing*, vol. 20, pp. 145–159, 2015.

[15] D. Domingos, F. Martins, and C. C, “Internet of Things Aware WS-BPEL Business Processes - Context Variables and Expected Exceptions,” *J. UCS 20.8*, vol. 20, no. 8, pp. 1109–1129, 2014.

[16] N. Glombitza, S. Ebers, D. Pfisterer, and S. Fischer, “Using bpel to realize business processes for an internet of things,” in *Int. Conference on Ad-Hoc Networks and Wireless*. Springer, 2011, pp. 294–307.

[17] S. Meyer, A. Ruppen, and C. Magerkurth, “Internet of things-aware process modeling: Integrating IoT devices as business process resources,” *Lecture Notes in Computer Science*, vol. 7908 LNCS, pp. 84–98, 2013.

[18] S. Meyer, A. Ruppen, and L. Hilty, “The things of the internet of things in bpmn,” in *Advanced Information Systems Engineering Workshops*, 2015, pp. 285–297.

[19] A. Baumgraß, M. Botezatu, C. D. Ciccio, R. Dijkman, P. Grefen, M. Hewelt, J. Mendling, A. Meyer, S. Pourmirza, and V. Hagen, “Towards a Methodology for the Engineering of Event-driven Process Applications,” in *Proc. First Int. Workshop on Process Engineering*, 2015, pp. 1–12.

[20] P. Giner, C. Cetina, J. Fons, and V. Pelechano, “Developing mobile workflow support in the internet of things.” *IEEE Pervasive Computing*, vol. 9, no. 2, pp. 18–26, 2010.

[21] R. Pryss, M. Reichert, A. Bachmeier, and J. Albach, “Bpm to go: Supporting business processes in a mobile and sensing world,” 2015.

[22] J. Mass, C. Chang, and S. N. Srirama, “Wiseware: A device-to-device-based business process management system for industrial internet of things,” in *Internet of Things (iThings), Green Computing and Communications (GreenCom), Cyber, Physical and Social Computing (CPSCom) and Smart Data (SmartData), IEEE Inter. Conf. on*, 2016, pp. 269–275.

[23] M. Wieland, H. Schwarz, U. Breitenbucher, and F. Leymann, “Towards situation-aware adaptive workflows: Sitopta general purpose situation-aware workflow management system,” in *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*. IEEE, 2015, pp. 32–37.

[24] A. Marrella, M. Mecella, and S. Sardina, “Intelligent process adaptation in the smartpm system,” *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 2, pp. 25:1–25:43, Nov. 2016.

[25] H. Raik, A. Bucchiarone, N. Khurshid, A. Marconi, and M. Pistore, “Astro-captevo: Dynamic context-aware adaptation for service-based systems,” in *Services, IEEE 8th World Congress on*, 2012, pp. 385–392.