Retrofitting of Workflow Management Systems with Self-X Capabilities for Internet of Things

Ronny Seiger, Peter Heisig, and Uwe Aßmann

Software Technology Group, Technische Universität Dresden, Germany
{ronny.seiger, peter.heisig, uwe.assmann}@tu-dresden.de

Abstract. The Internet of Things (IoT) introduces various new challenges for business process technologies and workflow management systems (WfMS's) to be used for managing IoT processes. Especially the interactions with the physical world lead to the emergence of new error sources and unanticipated situations that require a self-adaptive WfMS able to react dynamically to unforeseen situations. Despite a large number of existing WfMS's, only few systems feature self-x capabilities to be used in the dynamic context of IoT. We present a retrofitting process and generic software component based on the MAPE-K feedback loop to add autonomous capabilities to existing WfMS's. Using a smart home example process, we show how to retrofit different WfMS's in an invasive and non-invasive way. Experiments and a brief discussion confirm the feasibility of our retrofitting processes and software component to add self-x capabilities to service-oriented WfMS's in an IoT context.

Keywords: Workflow Management Systems · Self-management · Internet of Things · Retrofitting.

1 Introduction

The application of Business Process Management (BPM) technologies in the context of the Internet of Things (IoT) is a new and vibrant research field as it promises easily configurable, flexible and reusable processes to be modelled, executed and analysed among the typical IoT entities including sensors, actuators, smart objects and humans. However, with these novel interactions and application domain, new challenges for both research fields arise that need to be addressed [9, 4, 13]. Especially the new dimension of interactions with the physical world and associated sensors and actuators introduces additional requirements for Workflow Management Systems (WfMS's) as the IoT devices are mobile, embedded and more constraint regarding their resources. The IoT entities and environment are the source of new errors, imprecisions and unforeseeable situations for the process execution that an *IoT WfMS* has to cope with.

A common approach for dealing with these new kind of unanticipated situations is to implement feedback loops and adaptation mechanisms to make a system self-aware and self-adaptive [7]. These mechanisms usually rely on goals

specifying aspects regarding the expected outcome of particular actions, adaptation strategies for dealing with unexpected behaviour, and interactions with sensors and effectors of the respective target systems to monitor and manipulate the components of the self-managed system [11]. Several approaches investigate the implementation of autonomic capabilities for specific WfMS's–with and without relations to IoT. Despite a very large number of existing WfMS's being actively used in industry and academia, these implementations are tied to specific proprietary WfMS's and not reusable within other systems, though.

In this work, we present a general framework and process for retrofitting existing (*legacy*) WfMS's with self-x capabilities (self-awareness, self-adaptation, self-healing, etc.) based on the *MAPE-K* control loop from engineering self-adaptive software systems and autonomic computing [5, 7]. We discuss two ways of retrofitting service-oriented WfMS's and existing processes using an implementation of the MAPE-K loop by a generic software component (*Feedback Service*) to realize self-x mechanisms with respect to arbitrary quality criteria. We show how to retrofit four different WfMS's with the help of a simple scenario process from the smart home domain as an example of an IoT environment.

2 Smart Home Scenario Process

The smart home is an excellent example of an IoT environment. It consists of various sensors for environmental factors (e. g., light levels, temperature, humidity) and actuators for controlling domestic appliances (e. g., light switches, thermostats, service robots) as well as smart objects equipped with sensing technologies (e. g., RFID and NFC). All these IoT devices interact with each other, with the physical environment and other objects, and most importantly with the residents of the smart home. Due to the nature of these interactions with the physical world and the entities being more imprecise, unreliable and unforeseeable, new sources of errors emerge for smart home control applications. Therefore, a WfMS orchestrating the interactions among all involved smart home entities on the business process level needs to be able to adapt the processes and itself to deal with unanticipated situations and errors.

Fig. 1 shows a simple smart home process in BPMN 2.0 notation. The process only switches on a specific light via a service call. In our case, a middleware for IoT (*OpenHAB*[1]) provides a RESTful web service interface to all sensors and actuators, including a HomeMatic dimmer switch for controlling the light. The process shows the typical interaction between the virtual and physical world in IoT environments. The WfMS calls the IoT middleware and with that the control software of the light dimmer to influence the physical environment. After executing this process, the light levels are expected to have reached a certain threshold in the physical world and the room is assumed to be lit up (*assumed state*). However, simple errors like a burnt or worn off light bulb may lead to an incorrect assumption of the lighting state (*light is on*) compared with the

[1] https://www.openhab.org/

actual state in the physical world (*light is off*), which may not be detected by the WfMS or the dimmer switch. To detect and remedy this kind of issue, the workflow execution has to be self-aware and self-adaptive to support the self-healing of the process. For our scenario, an additional light sensor is used to detect the broken light bulb and an alternative dimmer switch is triggered to light up the room. We call this detection and repair of inconsistent physical and virtual process execution states *Cyber-physical Synchronization* [20].

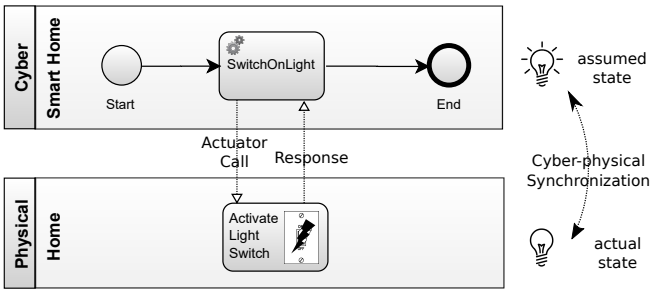

Fig. 1. Scenario Process Showing the Synchronization of Cyber and Physical Worlds.

3 MAPE-K as Framework for Autonomous Capabilities

The MAPE-K control loop is the most common approach to implement self-x capabilities for software systems operating in IoT and Cyber-physical Systems (CPS) [16]. In this section, we introduce the MAPE-K (Monitor-Analyse-Plan-Execute over a shared Knowledge) feedback loop [5] as the basic framework for adding autonomic capabilities to WfMS's.

3.1 The MAPE-K Feedback Loop applied to Processes

The application of the MAPE-K feedback loop [5] to manage processes and individual process steps is discussed in detail in [20]. In general, data from additional virtual and physical sensors is used to *Monitor* and *Analyse* the process execution with respect to certain success and error criteria defined in *Goals*. In case of unexpected behaviour, a compensation strategy is selectedby the *Planner* and performed by the *Executor*. The necessary *Knowledge* regarding sensors, actuators and compensation strategies is contained in a shared knowledge base [20].

We use *Goals* to specify relevant data and success/error criteria for a process used within the MAPE-K loops [20]. Fig. 2 shows our proposal of a generic extension of a component-oriented workflow meta-model [22] to specify these goals. The basic concept is the *ProcessStep*, which can be composed of other process steps or an atomic activity. By adding the *ManagedStep* interface to the abstract *ProcessStep* class, a *Goal* can be specified for the process step (i. e., on the level

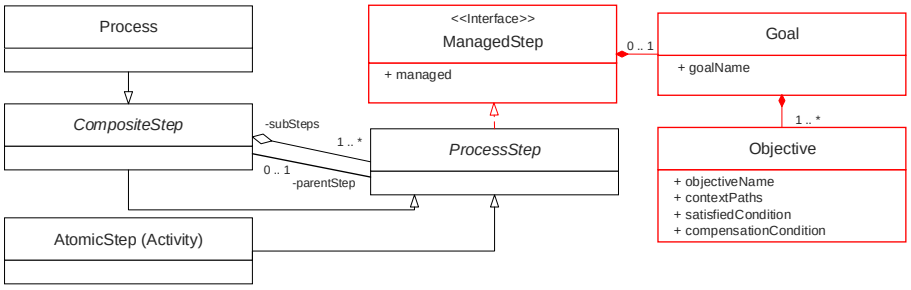

Fig. 2. Meta-model Extensions to Specify Goals for Self-management of Process Steps.

Listing 1.1. Goal and Objective for SwitchOnLight Process Step.

```
1  {"name": "enough light for working",
2  "objectives": [
3  {"name": "kitchen light > 600 lux in 2 seconds",
4  "satisfiedCondition": "#light > 600",
5  "compensationCondition": "#objective.created.isBefore
6          (#now.minusSeconds(2))",
7  "contextPaths": [ "MATCH (thing)-[:type]->
8          (sensor {name: 'LightSensor'})",
9   "MATCH (thing)-[:isIn]->(room {name: 'Kitchen'})",
10  "MATCH (thing)-[:hasState]->(state:LightIntensity)",
11  "MATCH (state)-[:hasStateValue]->(value)",
12  "WHERE toFloat(value.realStateValue) > 0",
13  "RETURN toFloat(value.realStateValue) AS light" ]}]}
```

of atomic activities, subprocesses and entire processes). This goal aggregates one or more *Objectives* defining data to be monitored in the *contextPaths*, success criteria for the process execution in the *satisfiedCondition*, and error criteria in the *compensationCondition*. Listing 1.1 presents the exemplary goal "enough light for working" (Line 2) for our smart home process *SwitchOnLight*. The goal contains one objective "kitchen light > 600 lux in 2 seconds" (Line 4). The satisfied condition defines the success criterion for the process step as reaching a light intensity of at least 600 Lux (Line 5). If this light value is not reached within 2 seconds, an error is assumed as defined in the compensation condition (Line 6). The context paths specify that the sensor values of the "LightSensor" in the room "Kitchen" should be monitored and analysed (Lines 7–12).

3.2 Implementation in the Feedback Service

We implemented the MAPE-K loop (*Autonomic Manager*) as a component-based micro-service called *Feedback Service*[2] (FBS) with components for each

[2] https://github.com/IoTUDresden/feedback-service

phase of the MAPE-K loop [20]. During process execution, the "Legacy" WfMS sends the goal and instance information regarding a *managed* process step to the FBS (cf. Fig. 3). Monitoring agents being part of the *Monitor* collect and update sensor data in the *Knowledge Base* continuously. The Knowledge Base is a central graph database storing all information regarding the IoT entities and process execution. Significant changes in relevant (according to the context paths) data (*Symptoms*) are forwarded to the *Analyser* component. The Analyser evaluates the execution based on the criteria defined in the satisfied condition (success) or compensation condition (error). If the satisfied condition is evaluated positively, then the FBS terminates the MAPE-K executions and the WfMS continues with the "regular" process execution. If the compensation condition is evaluated positively, then a *Change Request* is sent to the *Planner* to search for a compensation strategy. The Planner uses the determined *mismatch* contained in the change request and an extensible *Compensation Repository* to find suitable replacement resources and commands to be executed by the respective effectors. The derived *Change Plan* is then enacted by the *Executor* instructing the IoT actuators. In our example, the planner queries the knowledge base for an alternative process resource (dimmer switch) in the same context (kitchen) able to influence the same context factors (light levels) as the original resource and measured by the respective sensors (light sensor) specified in the context paths [20].

4 Retrofitting Process

After introducing the MAPE-K control loop as framework for self-managed software systems and its implementation for process-aware information systems by the Feedback Service (FBS), we show how to add self-x capabilities to existing service-oriented WfMS's. We distinguish between an *invasive* and a *non-invasive* retrofitting process. In general, we exploit the fact that most WfMS's are designed to orchestrate invocations of web services and applications to also call the FBS in parallel to the "original" service calls defined in the business processes.

4.1 Invasive Retrofitting

The invasive retrofitting process requires modifications to the WfMS's underlying workflow meta-model as well as to its execution logics. The meta-model has to be adapted as proposed in Sec. 3 to support the specification of *Goals* and *Objectives* for a process, subprocess and atomic process step. Fig. 3 shows the required changes regarding the execution behaviour of the "legacy" WfMS when executing a managed process step. In parallel to executing the basic process activity, the WfMS has to evaluate if this activity should be *managed* and issue a service call containing the respective goal to the FBS. The WfMS then has to wait/listen for a response from the FBS concerning the execution of the MAPE-K loop and the *State* of the goal (success or no success) [20]. The process execution continues w. r. t. this result–either execute the next process steps or initiate the WfMS's error handling mechanisms when the FBS is not able to fix the issue.

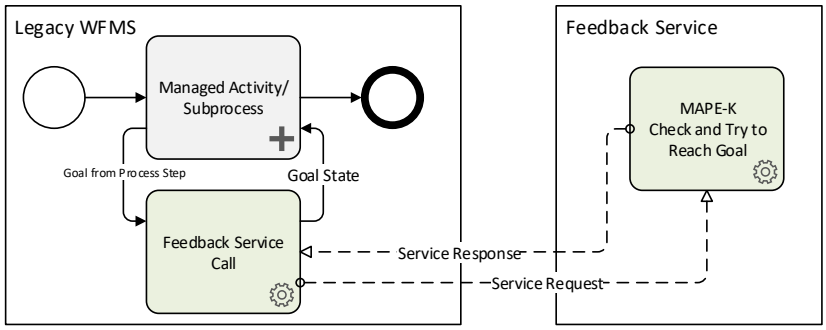

Fig. 3. Retrofitting of Existing WFMS's with the Feedback Service.

4.2 Non-invasive Retrofitting

While the invasive retrofitting process requires modifications to the meta-model and execution logics of the WfMS, the non-invasive retrofitting process relies on modifying the existing process models. The additional task of invoking the FBS depicted in Fig. 3 has to be modelled as an explicit process activity describing an (asynchronous) call to the FBS in parallel to the managed process step. This call contains the respective goal as input parameter. The process execution continues once the "regular" process step and FBS call were executed successfully.

5 Evaluation

In this section, we show the application of both retrofitting processes to existing WfMS's. The chosen WfMS's are open source software projects that rely on different workflow notations and execution engines. The process to be executed is the smart lighting process described in Sec. 2. We "break" the lamp to be switched on by removing its bulb, which cannot be detected by the respective WfMS or HomeMatic switch, i. e., the light is still off. The FBS uses an additional TinkerForge light sensor and a second dimmer switch to check and repair the "broken" process. This process is executed in a controlled lab environment once per WfMS. The WfMS's, FBS and OpenHAB run on a central computer (Intel Core i7, 4x3.1 GHz, 8 GB RAM, 32 GB SSD, 2 TB HDD, Ubuntu Linux 14.04). The processes and services used in the experiments can be found on GitHub[3].

5.1 Invasive Retrofitting of Existing WfMS's

PROtEUS The *PROtEUS* WfMS is designed to be used in the context of IoT and CPS [21]. It relies on a component-based architecture [21] and process meta-model [22]. We extended the meta-model and execution behaviour as suggested in Section 3. Fig. 4 shows the process model of the smart lighting process. The

[3] https://github.com/IoTUDresden/fbs-retrofit

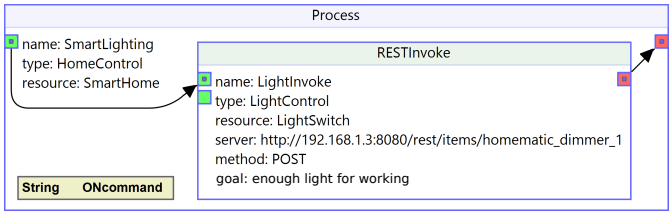

Fig. 4. Retrofitted Smart Lighting Process for PROtEUS.

basic process activity is a RESTful service call to the IoT middleware to switch on the light. This process step is augmented with the *goal* attribute from Listing 1.1. The execution of the *LightInvoke* process step–an asynchronous REST service to the middleware–took 193 ms with PROtEUS. The parallel invocation, error detection and compensation of the broken light bulb by the FBS dimming up the second light stepwise from 85 Lux to 657 Lux took approx. 24 s. A more detailed performance evaluation of PROtEUS and the FBS can be found in [20].

5.2 Non-invasive Retrofitting of Existing WfMS's

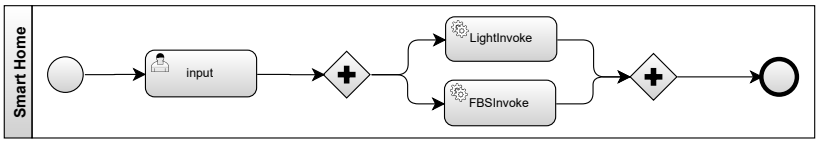

Fig. 5. Retrofitted Smart Lighting Process for Activiti.

Activiti The extended BPMN 2.0 smart lighting process executed with Activiti[4] 6.0.0 is depicted in Fig. 5. The *Input* process step is used to provide input parameters (goals). The basic process activity is the *LightInvoke* service task to call a custom service triggering the dimmer switch via OpenHAB. In parallel, the *FBSInvoke* service task calling the Feedback Service with the goal parameters is specified. Activiti relies on intermediate services that are locally deployed for executing external functionality. In our example, these services are custom implementations that delegate the calls to the actual RESTful services provided by the IoT middleware and FBS. The execution of the basic *LightInvoke* step took 459 ms with Activiti. The parallel invocation of the Feedback Service dimming up the second light stepwise from 89 Lux to 766 Lux took approx. 22 s.

YAWL Engine The extended YAWL smart lighting process executed with the YAWL engine[5] 4.2 is depicted in Fig. 6. The YAWL system also relies on

[4] https://www.activiti.org/
[5] http://www.yawlfoundation.org/

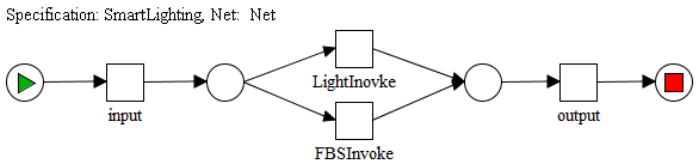

Fig. 6. Retrofitted Smart Lighting Process for YAWL.

custom services/classes that are deployed locally to invoke external applications or services. The *input* and *output* services print input/output parameters for debugging purposes. The *LightInvoke* step is the basic process activity that calls the IoT middleware delegated via our custom service to activate the light dimmer. The *FBSInvoke* service is the "retrofitting" process step that invokes the FBS with input parameters (goals) in parallel. The execution of the basic *LightInvoke* step took 171 ms with the YAWL system. The invocation of the FBS dimming up the second light from 81 Lux to 685 Lux took approx. 22 s.

Apache ODE The extended WS-BPEL smart lighting process executed with Apache ODE[6] 1.3.8 is depicted in Fig. 7. The first process step is used to receive input data and assign it to the following requests. The *LightCall* branch contains the assignment of input parameters to the following basic *LightInvoke* step, which calls a custom service to then call OpenHAB to switch on the light. In the parallel *FBSCall* branch, the *FBSInvoke* process step invokes a custom service to execute the FBS with goals provided as input parameters. Apache ODE also requires us to provide intermediate services to delegate the service calls to the actual RESTful web services of the middleware and FBS. The execution of the basic *LightInvoke* step took 98 ms with Apache ODE. The invocation of the FBS dimming up the second light from 93 Lux to 639 Lux took approx. 24 s.

6 Discussion

With the MAPE-K feedback loop as general framework, we are able to provide a flexible way of retrofitting existing WfMS's with autonomic capabilities. Due to the Feedback Service being implemented as a micro-service, a loose coupling with service-oriented WfMS's is possible. The retrofitting processes rely on an additional invocation of the FBS in parallel to the execution of the basic workflow activity, which is straightforward to realize as most WfMS's main purpose is the orchestration of web services. The invasive way of retrofitting requires minor changes to the workflow meta-model and internal mechanisms of the execution engine, which is not always possible due to inflexible and proprietary software architectures and the lack of support for service invocations–requiring a tighter integration of the FBS as an additional software component of the respective

[6] http://ode.apache.org/

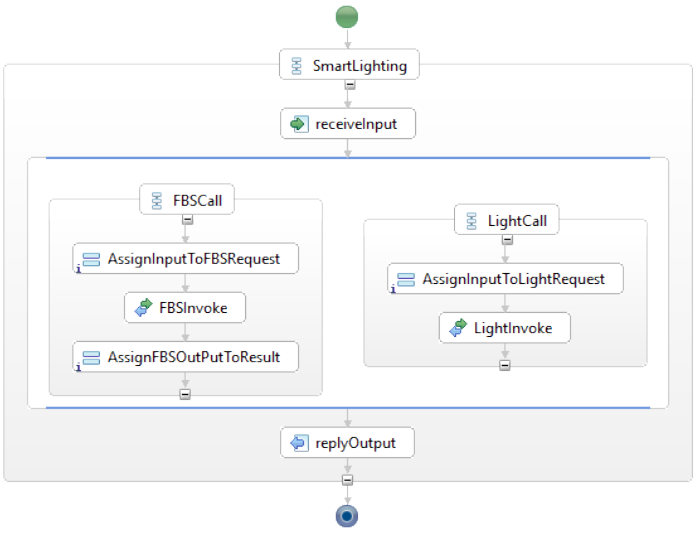

Fig. 7. Retrofitted Smart Lighting Process for Apache ODE.

WfMS. However, compared to the non-invasive way, the modelling of the basic workflows and additional goals can be introduced more naturally into the process landscape and tools. The non-invasive retrofitting process does not require changes to the underlying WfMS but to the existing process models. The modelling of the service call with the goal parameters parallel to the managed process step is less intuitive but maintains compatibility with the basic WfMS. Software or process engineers have to decide about which retrofitting process to choose based on these criteria. The selection of an appropriate "basic" WfMS depends on features and properties the WfMS has to provide and fulfil (e. g., formal verification or scalability) in the respective application domain or enterprise.

The operations executed by the Feedback Service introduce only little overhead to the overall process execution. In our evaluation examples, the major parts of the MAPE-K execution times relate to performing and analyzing actions in the physical world (e. g., increasing light levels by actuators and waiting for the sensors to detect these changes), which are usually much slower than virtual computations. The execution times for the basic service invocations are in similar orders of magnitude for all tested WfMS's. The Feedback Service adds the same execution times to the overall process execution.

Using the Feedback Service on the more abstract level of business processes to manage mostly discrete and asynchronous workflow activities proves to be feasible for our smart home IoT use case. Its suitability for managing more real-time demanding and synchronous processes on layers closer to the hardware remains to be investigated. The component-based architecture of the FBS facilitates the exchange and improvement of the individual algorithms used in the

phases of the MAPE-K loop. Our implementations of the analysis and planning phases are relatively simple but can be exchanged with more sophisticated approaches, e. g., from artificial intelligence as proposed in [14]. The Feedback Service can be used with respect to arbitrary context factors, key performance indicators, services levels, virtual and physical sensor values and other criteria to be considered within the MAPE-K feedback loops. Based on that, various self-x properties (self-awareness, self-adaptation, self-healing, self-optimization, self-configuration, etc.) can be added to existing WfMS's.

7 Related Work

A large number and variety of WfMS's exist and are widely used in academic and industrial contexts. Various works discuss the realization of autonomic capabilities for WfMS's in general (e. g., the MABUP system [18] or ViePEP [8]) and in the context of IoT and CPS (e. g., SmartPM [14], SitOPT [23] or WiseWare [15]). These approaches tie their realization of self-x mechanisms to specific self-developed WfMS's, which prevents reuse with other systems. With our implementation of the MAPE-K control loop in the Feedback Service [20], we are able to flexibly couple the autonomic service with other "legacy" WfMS's. Various works discuss aspects regarding the implementation of adaptive and flexible business processes most prominently as part of the *ADEPT* workflow system [6] and follow-up works [17]. *Worklets* [1] and *Exlets* [2] provide mechanisms for dynamic flexibility and exception handling in workflows. These approaches do not relate to IoT and they do not implement feedback loops to achieve self-* capabilities. However, they could be integrated into the planning and execution phases of the proposed MAPE-K loop for workflows to implement more sophisticated planning and adaptation strategies.

A general approach to implement self-managed systems is proposed in [11] and follow-up works. The retrofitting of a manufacturing system with fault-tolerance and security based on an external event-coordination layer is proposed in [24]. In [12] the authors present a way to add adaptivity to an existing scientific workflow engine by new components realizing the MAPE-K loop. Sensors provide data about the status of jobs running on the computation grid; once resources are available, the workflow engine is instructed to execute new workflows. The retrofitting of assembly processes with more high-level business processes based on components and services to facilitate synchronization across workflows is discussed in [3]. A general methodology for retrofitting autonomic capabilities onto legacy systems is presented in [19]. Sensors gather information (*Probes*) from the legacy systems that are analysed by *Gauges*, decision and coordination of adaptations are performed by controllers instrumenting the effectors of the legacy system. These works discuss specific retrofitting processes or general frameworks for adding autonomous capabilities to legacy systems based on feedback loops. None of the approaches discusses this retrofitting specifically for WfMS's. Our retrofitting processes and software component can be used with arbitrary WfMS's in IoT but also in more traditional business process domains.

8 Conclusion

In this work, we presented an approach for retrofitting existing workflow management systems (WfMS's) with autonomous capabilities to be able to handle unanticipated situations that emerge with new properties of IoT environments. We presented an invasive and a non-invasive way of retrofitting WfMS's using a generic software component that implements the MAPE-K feedback loop from autonomic computing. Our approach proves feasible for adding self-healing mechanisms to existing WfMS's executing smart home processes, but it can also be applied in wider business process contexts with respect to arbitrary self-x capabilities and service-oriented WfMS's. The application of our retrofitting framework to WfMS's operating in production contexts to implement self-adaptive workflows for *Industry 4.0* is subject to future work [10]. This domain usually imposes more real-time and safety-related constraints on the process executions.

Acknowledgements

This research has received funding under the grant number 100268299 by the European Social Fund (ESF) and the German Federal State of Saxony.

References

1. Adams, M., Ter Hofstede, A.H., Edmond, D., Van Der Aalst, W.M.: Worklets: A service-oriented implementation of dynamic flexibility in workflows. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 291–308. Springer (2006)
2. Adams, M., Ter Hofstede, A.H., Van Der Aalst, W.M., Edmond, D.: Dynamic, extensible and context-aware exception handling for workflows. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 95–112. Springer (2007)
3. Barros, A.P., ter Hofstede, A.H., Szyperski, C.: Retrofitting workflows for b2b component assembly. In: Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International. pp. 123–128. IEEE (2001)
4. Chang, C., Srirama, S.N., Buyya, R.: Mobile cloud business process management system for the internet of things: a survey. ACM Computing Surveys (CSUR) **49**(4), 70 (2016)
5. Computing, A., et al.: An architectural blueprint for autonomic computing. IBM White Paper **31**, 1–6 (2006)
6. Dadam, P., Reichert, M.: The adept project: a decade of research and development for robust and flexible process support. Computer Science-Research and Development **23**(2), 81–97 (2009)
7. 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)
8. Hoenisch, P., Schulte, S., Dustdar, S., Venugopal, S.: Self-adaptive resource allocation for elastic process execution. In: Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on. pp. 220–227. IEEE (2013)

9. Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C., Gal, A., Kannengiesser, U., Mannhardt, F., Mendling, J., et al.: The internet-of-things meets business process management: Mutual benefits and challenges. arXiv:1709.03628 (2017)
10. Jazdi, N.: Cyber physical systems in the context of industry 4.0. In: IEEE International Conference on Automation, Quality and Testing, Robotics. pp. 1–4 (2014)
11. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. In: Future of Software Engineering, 2007. FOSE'07. pp. 259–268. IEEE (2007)
12. Lee, K., Paton, N.W., Sakellariou, R., Deelman, E., Fernandes, A.A., Mehta, G.: Adaptive workflow processing and execution in pegasus. Concurrency and Computation: Practice and Experience **21**(16), 1965–1981 (2009)
13. Leotta, F., Mecella, M., Mendling, J.: Applying process mining to smart spaces: Perspectives and research challenges. In: Advanced Information Systems Engineering Workshops. pp. 298–304. Springer (2015)
14. Marrella, A., Mecella, M., Sardina, S.: Intelligent process adaptation in the smartpm system. ACM Trans. Intell. Syst. Technol. **8**(2), 25:1–25:43 (Nov 2016)
15. Mass, J., Chang, C., Srirama, S.N.: 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. pp. 269–275 (2016)
16. Muccini, H., Sharaf, M., Weyns, D.: Self-adaptation for cyber-physical systems: A systematic literature review. In: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 75–81. SEAMS '16, ACM, New York, NY, USA (2016)
17. Müller, R., Greiner, U., Rahm, E.: Agentwork: a workflow system supporting rule-based workflow adaptation. Data & Knowledge Engineering **51**(2), 223–256 (2004)
18. Oliveira, K., Castro, J., España, S., Pastor, O.: Multi-level autonomic business process management. Enterprise, Business-Process and Information Systems Modeling pp. 184–198 (2013)
19. Parekh, J., Kaiser, G., Gross, P., Valetto, G.: Retrofitting autonomic capabilities onto legacy systems. Cluster Computing **9**(2), 141–159 (2006)
20. Seiger, R., Huber, S., Heisig, P., Aßmann, U.: Toward a framework for self-adaptive workflows in cyber-physical systems. Software & Systems Modeling (Nov 2017)
21. Seiger, R., Huber, S., Schlegel, T.: Toward an execution system for self-healing workflows in cyber-physical systems. Software & Systems Modeling pp. 1–22 (2016)
22. Seiger, R., Keller, C., Niebling, F., Schlegel, T.: Modelling complex and flexible processes for smart cyber-physical environments. Journal of Computational Science **10**, 137 – 148 (2015)
23. Wieland, M., Schwarz, H., Breitenbucher, U., Leymann, F.: Towards situation-aware adaptive workflows: Sitopt: A general purpose situation-aware workflow management system. In: PerCom Workshops. pp. 32–37. IEEE (2015)
24. Xiao, K., Ren, S., Kwiat, K.: Retrofitting cyber physical systems for survivability through external coordination. In: Hawaii International Conference on System Sciences, Proceedings of the 41st Annual. pp. 465–465. IEEE (2008)