Controlling Internet of Things devices with Read-Write Linked Data Interfaces using Data-Driven Workflows

Nico Aßfalg, Leonard Nürnberg, Benjamin Jochum, and Tobias Käfer
uberq@student.kit.edu, ujeng@student.kit.edu, uzebb@student.kit.edu, tobias.kaefer@kit.edu

Institute AIFB, Karlsruhe Institute of Technology (KIT), Germany

Abstract. We present a first rough prototype for our approach to specify and execute agents on Read-Write Linked Data that are given as data-driven workflows, specifically we build on Guard-Stage-Milestone workflows. We showcase our approach in a setting from the Internet of Things, specifically building automation.

1 Introduction

The web can be argued to be at a stage where hypermedia agents could be applied [2], as its foundational technologies also for write access find widespread use, for instance:

- Microservices [10] give access to business functions
- Internet of Things devices with web interfaces (see e. g. the W3Cs Web of Things effort[1]) allow for enacting change using actuators
- Decentralised social networks such as SoLiD[2] allow for standards-based digitised social interaction

Overall, we see that besides the uniform interaction model of REST [4], or HTTP[3], also semantic descriptions using the uniform data model of RDF[4] find considerable uptake. For this data-driven environment called Read-Write Linked Data [1], we want to specify and execute behaviour. Hence, we are currently investigating data-driven approaches such as Guard-Stage-Milestone [5] to workflow modelling for their application as agent specifications for Read-Write Linked Data. Data-driven approaches define the course of action using data, i. e. system state, whereas traditional flow-based approaches use the finishing of activities to define the course of action. Workflows in data-driven approaches are considered more flexible than flow-driven approaches, e. g. it is easier in a data-driven approach to cover all exceptions and corner cases. We think that the data-driven

[1] https://www.w3.org/2016/07/wot-ig-charter.html
[2] https://solid.mit.edu/
[3] https://tools.ietf.org/rfc/rfc7230.txt
[4] https://www.w3.org/TR/rdf11-concepts/

approach provides a much better fit to the web architecture than the flow-driven approach for two reasons: (1) in both, data-driven workflows and on the web, system state is the first-class citizen (cf. REST – Representational State Transfer). (2) compared to traditional environments for workflow management, systems are typically more open and less under central control on the web. Thus, flexibility, ie. the ability to cope with unforseen but possible execution paths in a workflow, is a desired property in a workflow approach for the web. For an example of data-drivenness and flexibility, see our walk-through in Section 3.

The challenge is that properties of the environment play a major role in which workflow approaches are applied in what settings [3]. Therefore, we need to look at the assumptions of our environment Read-Write Linked Data when applying workflow technologies. Specifically, we found that Read-Write Linked Data is fundamentally different from traditional environments where workflow technologies are applied, e. g. databases, in the following regards:

The absence of events in HTTP In HTTP, there is no way for subscribing to events, which a controller could track for changes in the environment.

Reasoning and querying under the Open-World Assumption Under the open-world assumption, we cannot consider information we cannot derive as false, i. e. we cannot rely on negation when specifying the controller.

In previous works, we have been working towards this challenge: As a basis for applying workflow approaches for specifying behaviour, we defined ASM4LD, a model of computation for the environment of Read-Write Linked Data [7]. Thus, we can specify agent behaviour using rules. Based on this model of computation, we provided an approach to specify flow-driven workflows [8] and presented a corresponding demo [9]. In this demo, we present a data-driven approach instead.

Also, there is previous work in workflow management. Hull et al. developed the Guard-Stage-Milestone approach [5] for central data bases instead of the web architecture. Pautasso investigated how to retrofit REST into flow-based workflows [11], whereas we fit an approach for data-driven workflows into REST.

The demo shows our work on two main parts, a workflow ontology and operational semantics for the Guard-Stage-Milestone workflow language in the ASM4LD model of computation [7]. The contribution of the demo is a first rough implementation for a simple use-case from the Internet of Things. This submission is accompanied by a web page[5]. We published a paper on the formal aspects of the approach [6].

The paper is structured as follows: In Section 2, we present the architecture of our demo and the set-up. Next, in Section 3, we describe how a visitor to our demo booth can interact with our demo. Last, in Section 4, we conclude.

2 Architecture and Set-Up

Figure 1 shows the components of our implementation for storing and executing data-driven workflows to control IoT devices. Hence, we have four components or

[5] `http://people.aifb.kit.edu/co1683/2019/gsm/semantics-demo/`

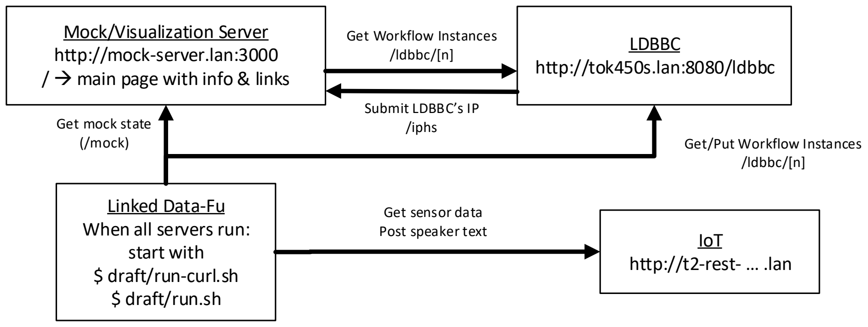

Fig. 1. The Demo System's Architecture

component families: (1) A Linked Data Platform container to maintain workflow models and instances, (2) an execution engine for the workflows, (3) a grapical user interface (4) the actual devices to control. All components are connected via a network switch that can be coupled with a notebook or a wifi router. We will now in turn provide more detail on the components:

Linked Data Basic Basic Container (LDBBC) The LDBBC is an implementation[6] of a Linked Data Platform Container[12]. We use LDBBC to maintain data about workflow models and instances as Linked Data. LDBBC is written in Java and we deploy it on Eclipse Jetty, a Servlet Container.

Linked Data-Fu Linked Data-Fu[7] is a Linked Data processing engine written in Java [13]. Linked Data-Fu can be programmed in a subset of the Notation3 language's syntactical features to interact with and reason over Linked Data. Specifically, we deploy rules for our prototypical operational semantics of a workflow language[8] on Linked Data-Fu.

Visualization We wrote a component in Node.js and Express that allows to visualize the workflow's state stored on the LDBBC. This visualisation reads the Linked Data about the workflow instance state and shows it as a simplified GSM workflow model with small annotations. The left part of Figure 2 shows a screenshot of this UI.

IoT Devices We have a number of physical and virtual IoT devices in our demo. For the physical devices, we use a set of Tessel 2[9] IoT boards with modules for RFID sensors, light sensors, and loudspeakers. Each board runs a Node.js/Express server providing a REST API to get state information with measurements as Linked Data, or to receive messages to be read out using the speaker. For the virtual devices, we rely on a similar implementation as

[6] `https://github.com/kaefer3000/ldbbc`
[7] `https://linked-data-fu.github.io/`
[8] `https://github.com/nico1509/data-driven-workflows`
[9] `https://tessel.io/`

the visualisation UI (see the right part of Figure 2). Running out of boards, we implemented a counter using this UI.

3 Walkthrough

The scenario of our demo will be an evacuation workflow. The workflow represents the programming of an automatic evacuation control system. When approaching our stand, the visitor will see: a set of IoT devices with RFID sensors attached to interact with the workflow (used for representing alarm buttons and doors that have to be closed), speakers for notifications by the workflow (used for giving evacuation instructions), and a screen showing status information about workflow instances (used for visualising the evacuation progress and for explanations on the overall system).

We show the workflow on the left side of Figure 2. Each activity, called *stage* in Guard-Stage-Milestone consists in an action, ie. here: an HTTP request that changes something, a guard, ie. a condition on the system state that must hold before the action is executed, and a milestone, ie. a flag that is set after the action finished. The guards and the milestones reflect the *data-drivenness* of this workflow approach: They determine the course of action based on system state, in contrast to flow-driven approaches such as BPMN, which only take the finishing of other activities in the arrows into account. As guards and milestones can get invalidated, deviations in the course of action, which call for flexibility in the models, can easier be covered in a data-driven approach:

The control system then works as follows. The visitor initiates the evacuation workflow by placing an RFID card on the reader that represents a fire alarm button. On the speaker, the workflow then announces a fire alarm, and that (1) emergency will arrive, people have to be (2) evacuated and (3) counted. The visitor then can fulfil stages by placing and releasing RFID cards on corresponding readers. The operational semantics poll the system state including whether cards are on the RFID readers. The presence of cards on readers is then part of our conditions. One of the features of the data-driven approach is that stages can be invalidated, and thus need to be re-executed or are halted. For instance, if the initially placed alarm card is removed, the whole emergency procedure is stopped.

4 Conclusion

Using this demo, we want to show our work-in-progress in applying data-driven workflows for controlling Read-Write Linked Data resources to the Semantic Web community. We present a first rough implementation of our approach that ports the Guard-Stage-Milestone approach to the web architecture. We base our demo in the setting of Internet of Things, specifically building automation, where currently a W3C community group is investigating web technologies for application[10].

[10] `https://w3c-lbd-cg.github.io/bot/`

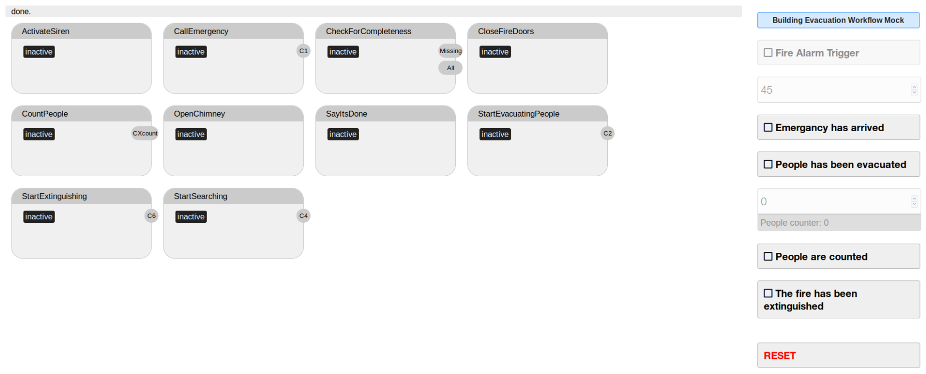

Fig. 2. Our UI to show the workflow progress and the virtual IoT devices.

References

1. Berners-Lee, T: Read-Write Linked Data. Design Issues, (2009). `http://www.w3.org/DesignIssues/ReadWriteLinkedData.html`
2. Ciortea, A, Mayer, S, Gandon, F, Boissier, O, Ricci, A, and Zimmermann, A: A Decade in Hindsight: The Missing Bridge Between Multi-Agent Systems and the World Wide Web. In: Proc. 18th AAMAS (2019)
3. Elmroth, E, Hernández-Rodriguez, F, and Tordsson, J: Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment. Future Generation Computer Systems 26(2) (2010)
4. Fielding, R: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, USA (2000)
5. Hull, R et al.: Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles. In: Proceedings of the 7th International Workshop on Web Services and Formal Methods (WS-FM) (2011)
6. Jochum, B, Nürnberg, L, Aßfalg, N, and Käfer, T: Data-Driven Workflows for Specifying and Executing Agents in an Environment of Reasoning and RESTful Systems. In: Proc. 3rd AI4BPM at the 17th BPM (2019). Accepted
7. Käfer, T and Harth, A: Rule-based Programming of User Agents for Linked Data. In: Proc. 11th LDOW at the 27th WebConf (2018)
8. Käfer, T and Harth, A: Specifying, Monitoring, and Executing Workflows in Linked Data Environments. In: Proc. 17th ISWC (2018)
9. Käfer, T, Lauber, S, and Harth, A: Using Workflows to Build Compositions of Read-Write Linked Data APIs on the Web of Things. In: Proc. Posters & Demonstrations at the 17th ISWC (2018)
10. Newman, S: Building Microservices-Designing Fine-grained Systems. O'Reilly (2015)
11. Pautasso, C: RESTful Web Service Composition with BPEL for REST. Data and Knowledge Engineering 68(9) (2009)
12. Speicher, S, Arwe, J, and Malhotra, A, eds.: Linked Data Platform 1.0. Recommendation, W3C. (2015). `http://www.w3.org/TR/ldp/`
13. Stadtmüller, S, Speiser, S, Harth, A, and Studer, R: Data-Fu: A language and an interpreter for interaction with R/W Linked Data. In: Proc. 22nd WWW (2013)