A Conceptual Architecture for Enabling Future Self-Adaptive Service Systems

Fabian Burzlaff
University of Mannheim
burzlaff@es.uni-mannheim.de

Christian Bartelt
University of Mannheim
bartelt@es.uni-mannheim.de

Abstract

Dynamic integration methods for unknown data sources and services at system design time are currently primarily driven by technological standards. Hence, little emphasis is being placed on integration methods. However, the combination of heterogeneous data sources and services offered by devices across domains is hard to standardize. In this paper, we will shed light on the interplay of self-adaptive system architectures as well as bottom-up, incremental integration methods relying on formal knowledge bases. An incremental integration method has direct influences on both the system architecture itself and the way these systems are engineered and operated during design and runtime. Our findings are evaluated in the context of a case study that uses an adapted bus architecture including two tool prototypes. In addition, we illustrate conceptually how control loops such as MAPE-K can be enriched with machine-readable integration knowledge.

1. Introduction

Currently more and more companies are making their domain-specific data available for third-party usage. Well-known examples are projects like BIG IoT, FIWARE or Industrial Data Space . For example, FIWARE aims at establishing an open data platform in the context of Smart City applications whereas the Industrial Data Space project is closely related to business models that are built upon Industry 4.0 data sources. In general, such platforms integrate different data sources and interfaces by connecting service providers with requesters by using some sort of adapter. For example, necessary semantic translations between the system data model as well as the IoT-device data model are codified within these adapters. Unfortunately, this use-case specific integration knowledge is trapped as it cannot be queried in a structured way. Consequently, when an IoT-device with similar functionality should be integrated with other platform services, integration knowledge cannot be reused automatically.

Current integration approaches utilized at design time are primarily driven by technologies and do not address this problem efficiently [1]. These technologies are often crafted for supporting certain domain standards. However, such standards are mostly the results of tedious and slow voting process. Hence, they are out-of-date as soon as they are available as new devices and data sources enter the IoT market almost daily in a decentralized manner. This means, that such standards are never complete as a basis for full automation even though, by definition, they should be.

Self-adaptive systems are predestined to work within such dynamic and undetermined context as they do include rules and mechanisms for reacting to context changes. However, they mostly rely on the assumption that once all adaptation rules are formalized, the system can perform all necessary adaptations autonomously. It is hard to believe that a system designer can anticipate all possible states an IoT-System can enter and hence provide a consistent view of all possible integration rules. Nevertheless, a machine-readable description of data and service semantics is fundamental for realizing automated device integration and service composition scenarios. Our contribution in this work is a conceptual architecture for future self-adaptive bus systems that allows for an evolutionary definition of device integration rules. Consequently, we also provide an integration approach that handles incomplete context knowledge for self-adaptive system architectures. To show the technical feasibility of our approach, we introduce tool support for formalizing integration knowledge in an incremental way.

1.1. Motivating Example

In order to motivate the challenge of semantic ambiguity of data and services within the IoT domain, we will use an example from the field of homonyms. Consider the verb close. Depending on the context,

URI: https://hdl.handle.net/10125/60184
ISBN: 978-0-9981331-2-6

this term stands for physically moving so that an opening is closed or to make a gap smaller. For example, a mobile robot approaching a door may expose the service close(Object object). Without the implicit integration knowledge of a system integrator, the adequate semantic assignment for this concrete use-case is undecidable for an automatic service matcher without context knowledge.

An application developer also faces other challenges among homonyms when figuring out the meaning of a service regarding request parameters, response elements as well as their types when integrating an API into an overall workflow (see Figure 1).

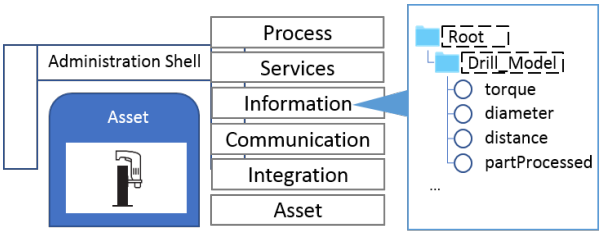

Figure 1. IoT-Interface Example

In case no machine-readable semantic standard for such drill exists, a system integrator must either consult an informal standard (e.g. ISO 16100 for Industrial Automation Systems and Integration) or simulate the drill to assign an unambiguous semantic label to the necessary model elements. Once a crisp semantic is established, an adapter could be programmed to map the device model element to the bus data model in a meaningful way. However, the mental mappings residing in the system integrators head are only stored inside imperative code structures, which in turn are hard to query. Hence, as soon as a similar interface has to be integrated in the future, all work must be repeated without any possibility to reuse already available integration knowledge from within adapters.

1.2. Problem statement and Paper Outline

Current approaches for formalizing adaptation scenarios in self-adaptive systems follow the idea of a revolutionary adaptation plan. If a condition holds, then a plan is executed [2]. However, for such a scenario to work a complete and shared knowledge base must be defined upfront. In this work we will shed light on incorporating a shared knowledge base that allows for an evolutionary formalization approach. Furthermore, we will provide first technical insights on how to deal with incomplete instead of complete integration knowledge in an automated way. To do so, our leading research questions is:

RQ: How can system integrators be enabled to formalize semantic knowledge within self-adaptive service systems and which challenges are still unsolved towards using this knowledge?

The objective of this paper is to conceptually align a novel integration method called Knowledge-driven Architecture Composition and self-adaptive service systems. The key results of this work are the following: First, to present all additional architectural elements needed and to discuss conceptually emerging upstream- as well as downstream consequences. Second, to demonstrate necessary tools and technologies that can be used to overcome ambiguous data elements and semantic shifts for IoT interfaces when using knowledge-driven architecture composition. Third and last, to demonstrate the potential of the introduced method within a simple case-study.

Our paper is structured in the following way: Chapter 2 gives a short overview of an incremental, use-case based integration method and its delamination to existing work. Chapter 3 represents our main contribution: A conceptual architecture as well as an adequate tool support for capturing semantic integration knowledge as a new component for self-adaptive systems. Furthermore, an example is used to exemplify our method as well as its potential benefits for control loops. Chapter 4 answers the second part of our research question by outlining challenges faced during the case study. Finally, chapter 5 concludes our work.

2. Knowledge-driven Architecture Composition and Service Integration Methods

Currently, several actors within the IoT market come up with new models for describing, managing and acting upon IoT devices [3]. However, such standards seldomly formalize the semantics in a machine-readable way. Among other factors, the applicability of negotiated standards is influenced by their claim to be complete. Hence, as soon as a new standard must be supported, manufacturers are forced to support it in a revolutionary big-bang approach (e.g. OPC UA). Current formal scientific solutions are also driven by the goal of being formulate for all possible use cases in a complete manner. For example, SOAP-based descriptions in bus systems must be created for every possible use case that may occur during runtime. Hence, the formalization effort for practitioners tends to be too high as the effort for describing each integration case usually does not pay off. Although such

approaches facilitate automated component coupling scenarios, practitioners currently rely on implementing point-to-point adapters. Consequently, current top-down integration approaches (i.e. standards) are not applicable within the dynamic and decentralized IoT market as new devices with new functionality enter the market almost daily.

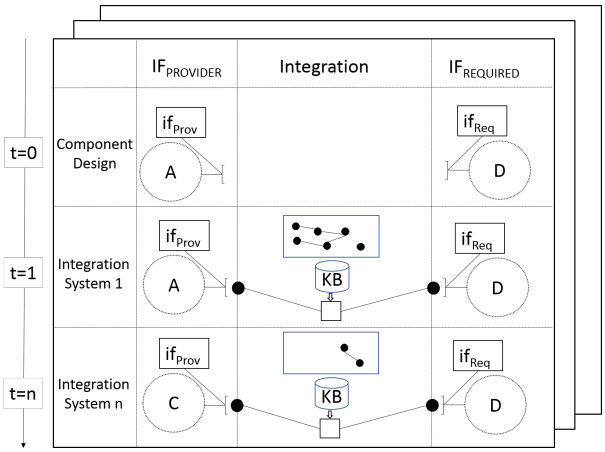

Figure 2. KDAC Principles Illustrated

To tackle this problem, we introduced a novel integration method called Knowledge-driven Architecture Composition (KDAC) that relies on an incremental semantic formalization process. This method explicitly allows for incompleteness of integration knowledge and supports an evolutionary instead of a revolutionary definition (i.e. big-bang formalization) of integration cases (see Figure 2). By using declarative languages based on first-order logic (e.g. OWL-DL), the inherited problem of partially incomplete integration knowledge can be tackled. The rationale for this is the usage of reasoning principles from the field of ontologies.

For example, due to the transitive characteristics of converting temperature units, a reasoner could infer new integration knowledge. So, if the integration knowledge base contains the mathematical conversion formulas from Celsius to Fahrenheit and from Fahrenheit to Kelvin, a logic-based reasoner can directly infer the conversion function from Celsius to Kelvin. Based on the underlying principle such inference procedures and the reuse of machine-readable integration knowledge from previous integration cases can facilitate automated component composition (e.g. plug-and-play principle). Automated component coupling is achievable as soon as all functional service and data characteristics are present and/or can be deduced.

In other words, one can think of the reasoning process over integration knowledge bases as playing the game Sudoku: As soon as enough integration knowledge for an unknown and new integration case is present, the missing information can be calculated based on rigorous mathematical rules.

The novelty of this approach is formalizing semantic integration knowledge per use-case in a bottom-up manner. By focusing on integration knowledge instead of conforming to technological-oriented interface descriptions, it maximizes the effort for formalizing the semantic coupling process as a concrete use-case must be present. Hence, formalization does only take place if there is a specific need for it.

In previous work [4], it was shown that this approach is feasible and could reduce the integration time in the context of the Industrial Internet of Things from 90 to 20 minutes. Furthermore, the evaluation shows that reusability of semantic mappings between endpoints is significantly higher than adapting imperative software adapters. The resulting knowledge base is a promising source for self-adaptive control loops.

2.1. Impacts of KDAC on Self-Adaptive Service Systems

The IoT system integration market is expected to increase from 17.0 billion US$ (2017) to 35.7 billion US$ (2020). One central driver for this are services and applications realized by IoT-devices. Hence, self-adaptive service systems are a promising way for applying a novel integration method for supporting system integrators as they already offer mechanisms for dealing with context uncertainty at different complexity levels. When integrating software components in a semantical way based on partially incomplete integration knowledge, there are several upstream (i.e. system integrator viewpoint looking from technical integration layer) and downstream effects (i.e. user goals and need looking from data and service request layer) from an architectural perspective (see Figure 3):

- An upstream effect is the reusability of integration knowledge for new IoT device. In case each IoT device has a unique resource identifier (URI) attached, the respective device type can be matched against the integration knowledge base and already existing composition knowledge from previous cases can be automatically generated. However, there must be a possibility to tell the system integrator which mapping elements between IoT device model and system model are missing. Nevertheless, as integration knowledge is only formalized if a use-case is present these

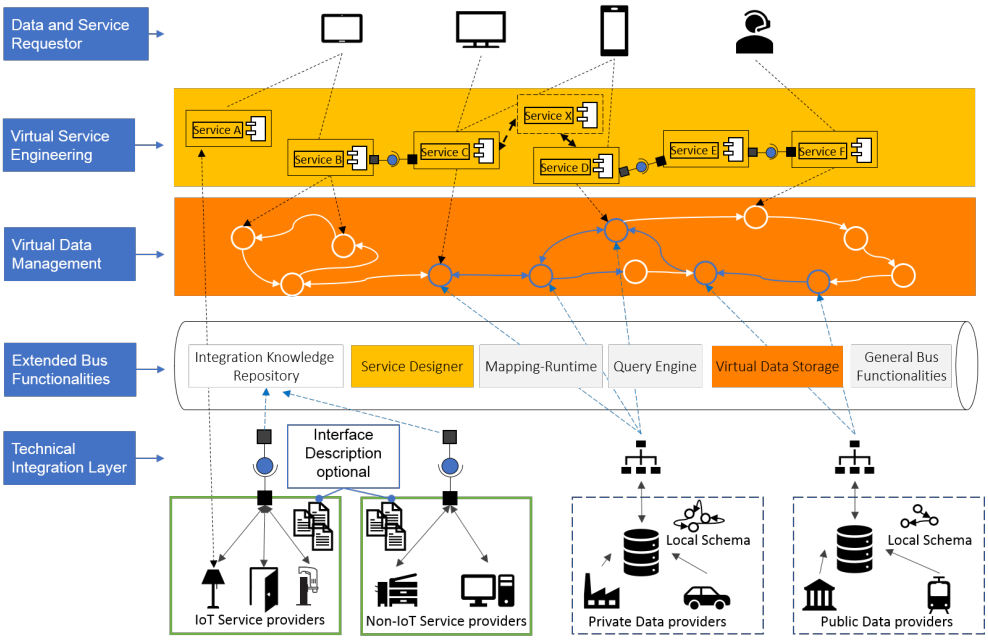

Figure 3. Conceptual System Architecture for a self-adaptive Service System supporting KDAC

<condition, event> combinations can be reused for the definition of more complex adaptation plans. In addition, they must not be made up by adaptation plan designers at design time out of thin air.

- A downstream effect when formalizing integration knowledge in an incremental way is the enrichment of the knowledge base that can be consulted by algorithms that implement the MAPE-K loop. Here the KDAC approach can be seen as a moderator that sets the stage for compositions approaches. Hence, basic communication semantics must not be defined in a revolutionary way and can be utilized by composition algorithms. Subsequently, this also means that in the beginning a human must be present. However, following the assumption that IoT integration cases are similar across multiple sites, an empty knowledge base may initial be filled with generic integration knowledge. This is illustrated as three global boxes in Figure 2.
- Another downstream effect is the way user goals are evaluated. In an evolutionary designed self-adaptive IoT System that realizes user goals by using data and services offered by decentralized IoT components, it would theoretically be feasible to locate missing devices. For example, a user-specific goal realizable by a system residing on one site may be not realizable by a system at another site. However, as the shared integration knowledge-base contains semantic integration knowledge and does not depend on a concrete interface syntax, new IoT devices and/or data sources can be integrated on demand. Now the aforementioned upstream effect of formalizing data and service models can support a plug-and-play-like component integration mechanism. Hence, new user-specific goals can be realized efficiently without altering the overall adaptation logic or even stopping the system.

As a result, this discussion shows that the unique characteristics of incomplete case-bases and an evolutionary instead of a revolutionary approach for formalizing basic composition rules influences the architecture of self-adaptive systems. Despite this observation, user centric self-adaptation plans in such systems require a lot of distinct integrations cases to enable IoT ensembles to fulfill a desired user need in an automated way.

3. Integrating an Evolutionary Maintained Knowledge Base into a Bus Architecture

A central problem for applying knowledge-driven architecture composition to self-adaptive systems is to capture semantic integration knowledge. Hence, a necessary step is the development of adequate tools and technologies. A major problem to resolve ambiguous

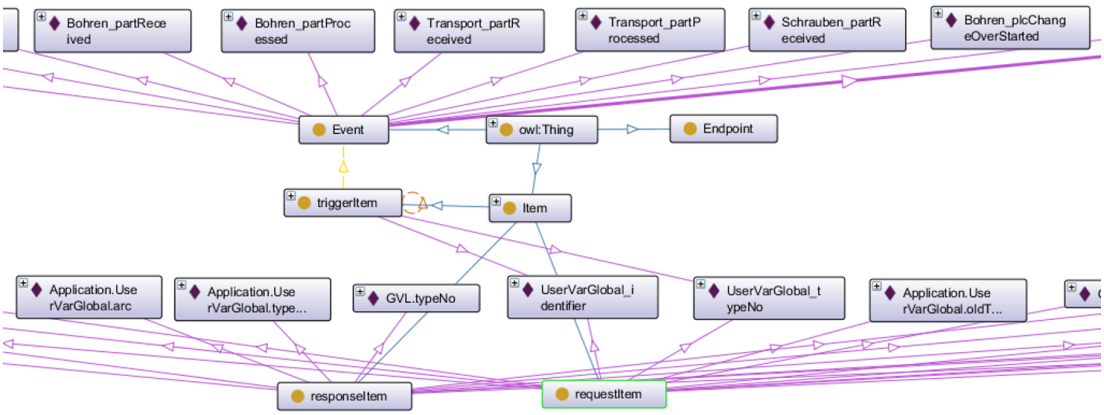

Figure 4. Integration Knowledge Base for trigger-request pattern

data and interface declaration is the formalization of the needed case-specific integration knowledge in an explicit way. To minimize the formalization effort, we explicitly allow for a domain-specific schema. Hence, based on our preliminary results gained from simple client-server architectures [4] the upcoming sections focus on demonstrating the basic tools needed for applying knowledge-driven architecture composition within self-adaptive system.

As hardware as well as technically standardized software gets cheaper (e.g. embedded operating systems), system integration can be seen as the future bottleneck for delivering IoT-Services to users seamlessly. Ideally, devices can be exchanged in self-adaptive IoT-systems (e.g. DeltaIoT [5]) seamlessly. However, integrating a new IoT-Device is currently not only linked to a significant manual integration effort but also relies on shared knowledge bases for control loops.

In the context of self-adaptive systems, one of the most commonly used architectural paradigms are event buses. For this architectural style, one can locate technical service integration issues at the following abstract layers [6]:

- Technical Layer: On this layer, it must be ensured that at least one communication protocol is supported at the network layer from the bus as well as the device that should be attached to it
- Syntactic Layer: On this layer, all data and services must be normalized in a sense that they conform to the overall bus schemata. Furthermore, configuration services may already provide different predefined configuration tactics
- Semantic Layer: On this layer, the relations of different services and their interaction style are defined

Based on the assumption that future service providers are conforming to some syntactical domain standards (e.g. OPC UA for industrial devices), one central challenge is the automation of device integration based on heterogenous service and data semantics. A conceptual self-adaptive bus system architecture that supports automated component composition is illustrated in Figure 3. The central architectural elements needed are sorted according to different layers:

- On the *Technical Integration Layer*, the concrete interfaces of devices are connected on the basis of their communication protocol with the bus communication channels (e.g. REST or HTTP for Web Services and SQL or MQTT for data sources)
- On the *Virtual Data Management Layer*, the different local data schemes from various data sources are integrated
- On the *Virtual Service Engineering Layer*, two types of services reside: The first one is an atomic service that can directly invoke a proprietary service running on a device. The second type of service is a complex service. This means, that such a service can be composed of other existing services or is subject to adaptation logic to fulfill a user goal.
- On the *Data and Service Requester Layer*, services can be accessed by the platform user in order to realize a certain need or a goal.

At the technical integration layer, additional formal interface descriptions are plotted. Although such semantic service descriptions are not commonly used in practice, they may still be present. If there exists

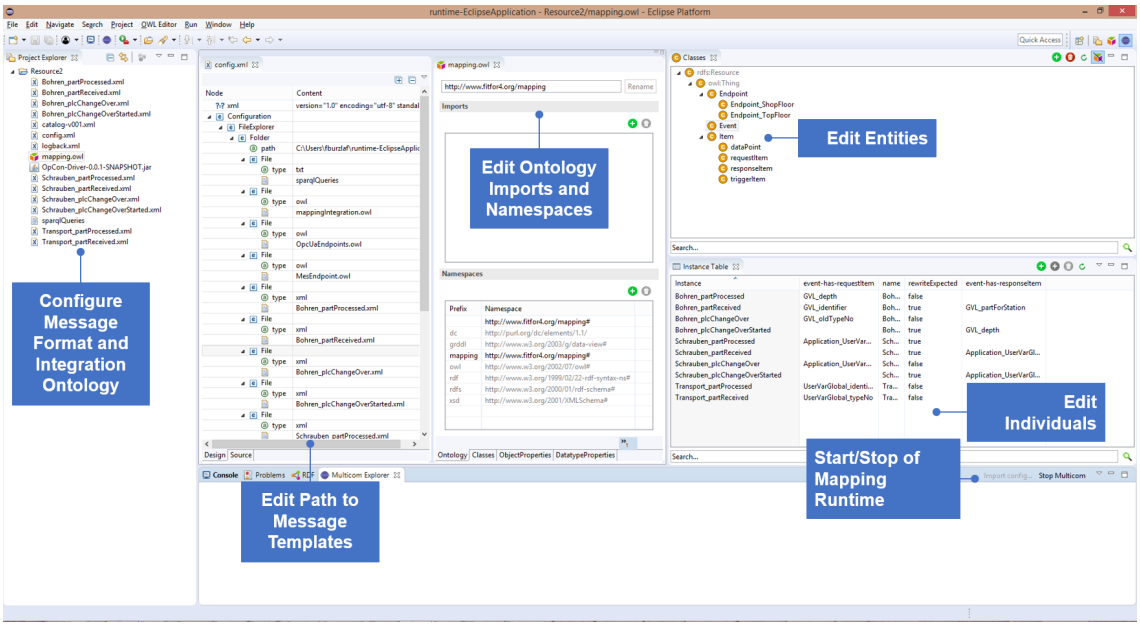

Figure 5. Mapping-Assistant for System Integrators

a symmetrical mapping between interface description as well as the provided interface functionality, then such descriptions are an additional valuable information source for the presented KDAC approach [7]. It was shown that the usage of this integration method can improve the matching result for semantic service matchers such as SAWSDL-MX [8] significantly.

Consequently, our main technological contribution will be focusing on the technical integration layer and its upstream effects on the proposed architecture (i.e. IoT-Device integration). Downstream effects such as user goal realization or service composition will not be focused for the remainder of this paper.

As a first prototypical evaluation, we will showcase the impact of our method by outlining the integration process for our motivating example within a simple case study. Furthermore, we will illustrate how the architectural elements Integration Knowledge Repository and Mapping-Runtime can be utilized by other processes during the adaptation life cycle (c.f. Figure 2). In addition, we will outline how our central theme of incomplete and evolutionary defined integration knowledge can be exploited by control loops.

3.1. Case Study IoT-Device Integration

An IoT-device, such as a drill, typically offers ways to access its information model and execute device-specific functionality. Typically, for sensing purposes values of information model elements such as torque can be read (c.f. Figure 1). For executing an action, at least one value must be written or one function (c.f. Service Layer in Figure 1) must be invoked.

In our use case, the following communication scenarios serves as a basis. There exists a drill device with the information model plotted in Figure 1. An iron rod is placed on the working area and the drill starts drilling a hole inside the rod. After the drilling process is finished, a message with the specific parameters used should be sent to a Manufacturing Execution System (MES). In this case, information elements offered by the drill can be accessed via OPC UA and the MES System offers a TCP/IP interface that can be fed with domain-specific XML telegrams. An XML telegram including several device parameters should be sent to the MES as soon as the value of the trigger item partProcessed is set by the drill device to true. For implementing the knowledge-base, we have chosen the OWL-DL language and used the HermiT 1.3 reasoner as a validation and reasoning engine. Hence, the IoT device provides a domain-specific functionality and the MES system requests a use-case specific information chunk. Regarding the proposed integration approach, the following steps must be taken (c.f. Figure 2).

At t=0, a system integrator must define the sketched integration scenario within the knowledge base. If the needed relationships are not defined within the ontology, the system integrator must define them using the Mapping Assistant (see Figure 5). Here, a new

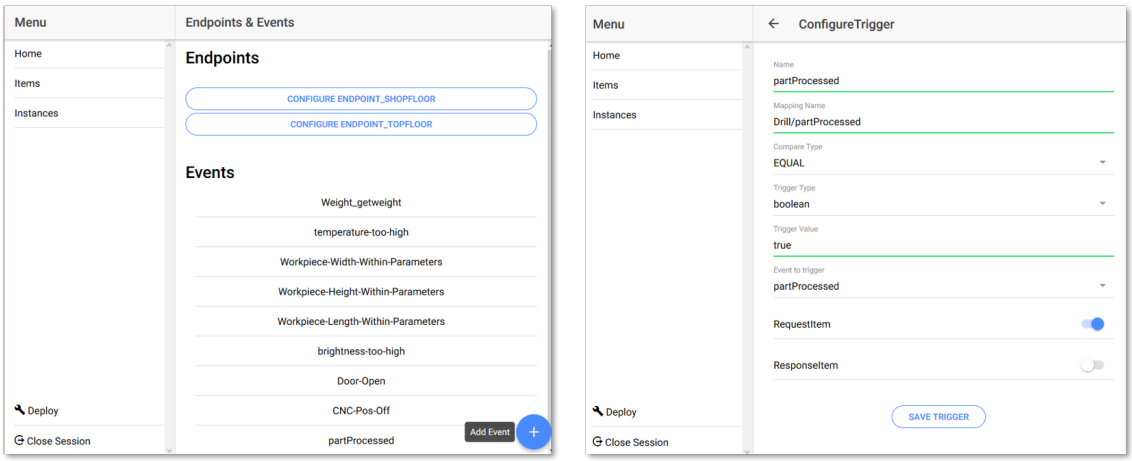

Figure 6. Overview Web App for Shopfloor Integration

<trigger, event> pair is defined as:

IF valueOf(trigger_Item) OPERATOR trigger_value THEN Event(requestItem [], responseItem [])

Next, the system integrator must translate each needed information model element within the use-case into individuals of the type triggerItem, requestItem and responseItem (see Figure 4): In our example, it is sufficient to transfer the information model elements partProcessed:triggerItem and diameter:requestItem as well as distance:requestItem as individuals.

At t=1, the use-case should be extended by also sending the requestItem torque within the event to the MES systems. However, the domain-specific MES telegram defines the torque as speed. Hence, the system integrator defines speed as an equivalent class of torque and reuses the existing integration knowledge.

At t=n, already integrated interfaces for the communication style trigger-event can be automatically reused by matching the device specific type IDs as well as their information and service model elements with a syntactical matcher.

When discussing our Mapping-Assistant with domain experts from our research project, it quickly appeared that the introduced tool may still be too complex for integrating a new device intuitively. Hence, we designed an additional input application for formalizing integration knowledge. Here, we used a progressive web application framework (i.e. IONIC) that only allows to add individuals to an existing ontology without allowing for adding new structural elements (see Figure 6). A benefit of such a progressive web-application is that it can be easily compiled to native-code for all known mobile devices (e.g. running on a tablet or a laptop). Hence, during

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<root>
  <header eventId="{@eventId@}" eventName="partProcessed" vers
    <location lineNo="10" statNo="3" statIdx="1" fuNo="1" work
  </header>
  <event>
    <result returnCode="0">
    </result>
    <trace />
  </event>
  <body>
    <items>
      <item name="distance" value="300" dataType="8" />
      <item name="diameter" value="20" dataType="8" />
    </items>
    <structs />
  </body>
</root>
```

Figure 7. Message sent over Bus

the integration iterations within the KDAC process this tool can also be used to perform some integration work at the Shopfloor incrementally without changing any code. Practically, this means that the system integrator can deploy changes to the Integration Knowledge Repository. After all files are updated in this repository, the Mapping-Runtime parses the OWL-File and provides the formalized integration knowledge during runtime for other interested architectural components. To conclude our use-case, the message being sent to the MES after all request values have been retrieved or derived is illustrated in Figure 7.

Please note that the proposed integration process must be performed in addition to connecting the services on a technical and syntactical level (i.e. implementing an adapter). Please notice further, that the bus used does not provide any means neither for integrating new IoT services automatically nor the possibility to search for manually defined integration knowledge. This is also due to the fact that the ontology schema is highly driven by the use-case and not driven by any community

standard. A rational for doing so is the intuitive applicability of logic-based languages within the IoT domain.

3.2. Related Work

Future self-adaptive systems must deal with adaptation strategies in decentralized settings and with unanticipated changes. Here, the Internet-of-Things serves as a new application opportunity for this system class. However, dealing with unanticipated changes within such system combinations is still a challenge yet to be solved. Therefore, we position the proposed method as a necessary pre-stage to enrich self-adaptive systems with knowledge.

Other research communities have already produced valuable solutions for interface matching, automated integration technologies and adaptation mechanisms [9]. As we cannot provide a concluding literature overview, we will focus on sketching possible solutions for our motivating IoT-Interface example (c.f. Figure 1) from different research perspectives.

The MAPE-K control loop can be regarded as the state-of-the-art for performing adaptation plans. From an engineering perspective, there must be a human-in-the-loop to create and maintain suitable models [2]. Here, the proposed method can help to formalize atomic ¡condition, event¿ pairs already during device integration which then can be used to enrich shared knowledge bases utilized by MAPE-K control loops.

Within the knowledge engineering community, evolutionary extending knowledge-bases by using incremental formalization techniques is not new [10]. As early as in 1994, it was described that it is hard to transform informally described knowledge into machine-readable knowledge. Here, the focus was to transform the former into the latter one. In our example, this would result in a tool where one can insert use-case specific <condition, adaptation plans> with natural language. However, their application case was not the integration of IoT-devices.

IoT interfaces can conceptually be linked to component-based software development [11]. Like physical IoT devices, a software component does provide functionality realized by software and hardware through an interface. Hence, syntactic matching approaches [12] can be used to determine whether a required interface matches a provided interface. Another approach would be to search for the specific adapters needed to bridge between two endpoints [13]. Regarding our example, one can view the services and information models as a search specification that can be fed to a matching system. However, such systems usually only provide a probabilistic result which may result in undesired effects for IoT actors in the physical world [12].

SOAP-based systems [14] and (semantic) web service descriptions [8] do expose mechanism to coupled services during runtime. These systems require each service to be defined in an additional interface description file which is attached to each device. In our example, this would result in the definition of additional interface descriptions (e.g. using the SAWSDL language). However, such approaches need a special runtime environment and also inherit a high formalization effort for IoT- device manufacturers. As argued earlier, IoT device manufacturers do not have a clear incentive to do so. In the case of Web Services, annotating web services with tags would be a minimalistic way of providing an interface description unfortunately, such tags are only beneficial for search engines optimized for humans and contain no unambiguous semantics.

Last, the software architecture community has recently come up with a reference architecture for fast self-commissioning of for industrial IoT-systems in process automation [15]. Although commissioning times could be reduced to a few seconds across vendor products, they rely on the assumption that all vendors conform to standardized models such as PLCOpen on every layer (c.f. Figure 1). However, this approach may be feasible in the context of safety-critical systems such as the AUTOSAR platform.

Overall, the dichotomy between machine-readable formalization effort and the potential benefit of automated integration and adaptation strategies for self-adaptive systems must be tackled. As a potential solution, we will now describe potential impacts of using our method for self-adaptive service systems in the next section.

3.3. Perspectives on Self-Adaptive Service Systems

Nowadays, marketplaces for data and services already exist for various domains [16]. By applying the Knowledge-driven Architecture Composition approach on self-adaptive systems conceptually, we took the first technical step towards reusing integration knowledge within the IoT domain. As a key result, the overhanging incompleteness claim for formalizing integration knowledge bases from bottom-up is technically feasible for self-adaptive service systems.

Another perspective on self-adaptive IoT systems deals with proprietary data and service semantics. Regarding the integration of new devices, most platform operators require system integrators to conform with predefined platform adapters (e.g. redlink or Hub-of-all-Things). Unfortunately, these adapters do not offer practical possibilities to document the semantic relations between the proprietary service provider and the service requester. This is due to the circumstance that both, (sensor) data and (web) services only contain weak semantics (i.e. ambiguous and informal semantic descriptions). For example, IoT data typically resides on a low abstraction level meaning that the expressiveness of a single data point itself is rather limited [9][17]. One central observation from our case study was that information models and services offered by IoT-devices are highly interlinked. The assumption of stateless services, which does hold for most web-services, does not hold in our use case. Hence, the state of an IoT-device may be changed by setting a value of a model element or by calling a function. The former originates from embedded and the latter one from software system design. Here, other semantic relationships must be captured.

Shifting from a revolutionary way of defining adaptation rules towards an evolutionary way seems promising for self-adaptive service systems. Among other reasons, the decentralized development and availability of IoT-devices also challenges the way how adaptation plans for control loops are formalized. Regarding the sketched knowledge-base, such atomic ¡condition, event¿ rules can serve as a sound basis for more abstract adaptation scenarios.

Another central observation when applying the KDAC approach was that the SPARQL-query became the bottleneck for runtime adaptations. If a new structural element or relationship is inserted into the OWL-DL ontology, which is not a subclass or a formula conversion, the queries will not return all individuals. This is mainly because the reasoner cannot infer all individuals at runtime. Here, the system integrator must adapt the SPARQL-Queries within the Mapping-Assistant. This requires additional skills and may result in code changes as new types are returned by the query itself.

4. Next Steps and Future Challenges

Based on the presented architectural influences on future self-adaptive IoT service systems by using our incremental integration method, we are currently planning future technical as well as conceptual improvements towards enabling smart application development in the context of Smart Cities. In this section, we will elaborate on some of the next steps that we plan to take.

Challenge 1 is an extended empirical evaluation of our method within a self-adaptive service system. Therefore, we plan to extend the DAiSI platform [18] with the introduced architectural elements and demonstrate that the KDAC approach efficiently helps to build up a shared knowledge base without relying on predefined integration standards. Another objective of this evaluation will be the usage of formalized integration knowledge to fulfill user goals dynamically with control loops.

Challenge 2 will be to fan out the human within the proposed integration method. Especially in the beginning, the human must perform additional work other than implementing software adapters. Hence, as soon as enough integration cases are present, the human must again fan out of the process and automated integration should take over. It must be methodically ensured that this turning point is reached as early as possible.

Lastly, challenge 3 deals with the usability of our method when integration knowledge is missing for an integration case. Here, a suitable recommender system must point out the integration mappings needed between decentralized developed ontologies and integrated IoT-devices. However, there already exist promising approaches by the semantic web community that can be utilized in order to calculate (c.f. Sudoku metaphor) missing links between semantic models efficiently [19].

5. Conclusion

In this work we have introduced a conceptual self-adaptive system architecture for services and data sources in the context of the Internet-of-Things. Next, we discussed the applicability of an incremental integration method called knowledge driven-architecture composition (KDAC). In particular, we showed that the KDAC method can be utilized to fill knowledge-bases in an evolutionary instead of revolutionary way. The formally gathered integration knowledge explicitly allows for incomplete integration knowledge without loosing the ability to be automated in the long run. Therefore, we introduced two tools and exemplified the proposed integration approach in the context of a self-adaptive system and a case study. We conclude that self-adaptive service systems are a promising candidate for further research. A more elaborate empirical evaluation will potentially

prove that the KDAC approach can efficiently fill knowledge-bases utilized by feedback and control algorithms.

Acknowledgement

This work was supported by the BMVI project xDataToGo (http://www.bmvi.de/goto?id=359354) under the support code 19F2048D.

References

[1] N. F. Noy, "Semantic Integration: A Survey of Ontology-based Approaches," *SIGMOD Rec.*, vol. 33, pp. 65–70, Dec. 2004.

[2] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and Analyzing MAPE-K Feedback Loops for Self-adaptation," in *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '15, (Piscataway, NJ, USA), pp. 13–23, IEEE Press, 2015.

[3] A. Meddeb, "Internet of things standards: who stands out from the crowd?," *IEEE Communications Magazine*, vol. 54, pp. 40–47, July 2016.

[4] F. Burzlaff and C. Bartelt, "I4.0-Device Integration: A Qualitative Analysis of Methods and Technologies Utilized by System Integrators: Implications for Enginering Future Industrial Internet of Things System," in *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pp. 27–34, Apr. 2018.

[5] M. U. Iftikhar, G. S. Ramachandran, P. Bollanse, D. Weyns, and D. Hughes, "DeltaIoT: A Self-adaptive Internet of Things Exemplar," in *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '17, (Piscataway, NJ, USA), pp. 76–82, IEEE Press, 2017.

[6] I. Crnkovic, S. Sentilles, A. Vulgarakis, and M. R. V. Chaudron, "A Classification Framework for Software Component Models," *IEEE Transactions on Software Engineering*, vol. 37, pp. 593–615, Sept. 2011.

[7] F. Burzlaff, C. Bartelt, and u. L. Adler, "Towards automating Service Matching for Manufacturing Systems: Exemplifying Knowledge-Driven Architecture Composition," *Procedia CIRP*, vol. 72, pp. 707–713, Jan. 2018.

[8] M. Klusch, P. Kapahnke, and I. Zinnikus, "SAWSDL-MX2: A Machine-Learning Approach for Integrating Semantic Web Service Matchmaking Variants," in *2009 IEEE International Conference on Web Services*, pp. 335–342, July 2009.

[9] P. Anantharam, P. Barnaghi, and A. Sheth, "Data Processing and Semantics for Advanced Internet of Things (IoT) Applications: Modeling, Annotation, Integration, and Perception," in *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics*, WIMS '13, (New York, NY, USA), pp. 5:1–5:5, ACM, 2013.

[10] F. M. Shipman, III and R. McCall, "Supporting Knowledge-base Evolution with Incremental Formalization," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, (New York, NY, USA), pp. 285–291, ACM, 1994.

[11] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. d. M. Silveira Neto, Y. C. Cavalcanti, and S. R. d. L. Meira, "Twenty-eight years of component-based software engineering," *Journal of Systems and Software*, vol. 111, pp. 128–148, Jan. 2016.

[12] M. C. Platenius, *Fuzzy matching of comprehensive service specifications*. PhD thesis, Universittsbibliothek, Paderborn, 2016.

[13] W. Janjic, O. Hummel, and C. Atkinson, "Reuse-Oriented Code Recommendation Systems," in *Recommendation Systems in Software Engineering* (M. P. Robillard, W. Maalej, R. J. Walker, and T. Zimmermann, eds.), pp. 359–386, Springer Berlin Heidelberg, 2014.

[14] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Nielsen, S. Thatte, and D. Winer, *Simple object access protocol (SOAP) 1.1*. Jan. 2000.

[15] H. Koziolek, A. Burger, and J. Doppelhamer, "Self-Commissioning Industrial IoT-Systems in Process Automation: A Reference Architecture," in *2018 IEEE International Conference on Software Architecture (ICSA)*, pp. 196–19609, Apr. 2018.

[16] F. Stahl, F. Schomm, L. Vomfell, and G. Vossen, "Marketplaces for digital data: Quo vadis?," Working Paper 24, Working Papers, ERCIS - European Research Center for Information Systems, 2015.

[17] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos, "IoT-Based Big Data Storage Systems in Cloud Computing: Perspectives and Challenges," *IEEE Internet of Things Journal*, vol. 4, pp. 75–87, Feb. 2017.

[18] H. Klus, A. Rausch, and D. Herrling, "DAiSIDynamic Adaptive System Infrastructure: Component Model and Decentralized Configuration Mechanism," tech. rep., IARIA, 2014.

[19] C. Meilicke, "Alignment Incoherence in Ontology Matching," 2011.