Institute of Architecture of Application Systems

Applying IoT Patterns to Smart Factory Systems

Lukas Reinfurt[1,2] and Michael Falkenthal[1],
Uwe Breitenbücher[1], Frank Leymann[1]

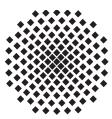
[1]Institute of Architecture of Application Systems,
University of Stuttgart, Germany,
[lastname]@iaas.uni-stuttgart.de

[2]Daimler AG,
Germany,
[firstname].[lastname]@daimler.com

BibTeX

```
@inproceedings{Reinfurt2017,
    author    = {Reinfurt, Lukas and Falkenthal, Michael and
                 Breitenb{\"{u}}cher, Uwe and Leymann, Frank},
    booktitle = {Proceedings of the 11\textsuperscript{th} Advanced Summer
                 School on Service Oriented Computing},
    pages     = {1--10},
    publisher = {IBM Research Division},
    title     = {{Applying IoT Patterns to Smart Factory Systems}},
    year      = {2017}
}
```

The full version of this publication has been presented as a poster at the Advanced Summer School on Service Oriented Computing (SummerSOC 2017).
http://www.summersoc.eu



Universität Stuttgart
Germany

Applying IoT Patterns to Smart Factory Systems

Lukas Reinfurt[1,2], Michael Falkenthal[1], Uwe Breitenbücher[1], and Frank Leymann[1]

[1] University of Stuttgart, Institute of Architecture of Application Systems
Universitätsstr. 38, 70569 Stuttgart, Germany
[firstname.lastname]@iaas.uni-stuttgart.de
[2] Daimler AG
Epplestraße 225, 70567 Stuttgart, Germany
lukas.reinfurt@daimler.com

Abstract. Creating Internet of Things systems is a complex challenge as it involves both software and hardware, and because it touches on constrained devices and networks, storage, analytics, automation, and many other topics. This is further complicated by the large number of available technologies and the variety of different protocols and standards. To help with the ensuing confusion, we presented Internet of Things Patterns in several categories, such as device communication and management, energy supply types, and operation modes. These patterns describe abstract solutions to common problems and can be used to understand and design Internet of Things systems. In this paper, we show that these patterns can be applied to Smart Factory systems, which is one of the many domains where the Internet of Things is applicable.

Keywords: Internet of Things, Architecture, Patterns, Industry 4.0, Smart Factory, Industrial Internet

1 Introduction

Building Internet of Things (IoT) systems is a complex endeavor. It requires successfully combining both software and hardware across various domains. Sensing and actuation capabilities have to be brought into all kinds of environments using constrained devices and networks. Moreover, collected data has to be communicated and turned into usable information, sometimes fast and sometimes in huge quantities. Data and information has to be stored, has to be made accessible to others, and is used as basis for automation. Remote sensing and control offers great possibilities, but also high risks when it comes to security, privacy, and safety. This situation gets more complicated by the current state of the IoT field. As it is still relatively new and growing, a wide variety of technologies and solutions pushed by vendors from different areas are fighting for attention [2,15,18]. There is also an abundance of standard, as these solutions often have been created in silos [23]. This makes it confusing for IoT developers and architects to find

appropriate technologies and solutions for their particular situation. To tackle these issues, we identified and collected *IoT Patterns*[3] in various categories that abstract from individual technologies [20,21,22]. During the design of IoT systems, architects can use IoT Patterns to solve specific problems they encounter. But applying the patterns is not limited to single problems. Rather, once one pattern has been applied, architects can now follow links to other related patterns. This enables them to build IoT systems step by step. As the IoT is applicable in many different domains, such as home automation, health care, logistics, and industrial fabrication, our IoT Patterns should also be applicable in these different domains. In this paper we will show that this is the case for industrial fabrication.

The remainder of this paper is structured as follows: In Section 2, we will present work related to IoT Patterns and their application. Section 3 introduces a motivating example from the domain of industrial fabrication which will be used in the remainder of the paper. Section 4 briefly describes the IoT Patterns which are relevant to this paper. Section 5 elaborates how these patterns can be used to describe and refine the system which was introduced in the motivating example. Finally, Section 6 ends the paper with a conclusion.

2 Related Work

We have published IoT Patterns for device energy supply and operation modes [22] and device communication and management [20,21]. Eloranta et al. presented patterns for building distributed control systems [5,6], which are concerned with reliability and fault-tolerance of large moving machines used for forrestry, mining, construction, etc. Qanbari et al. introduced four patterns for provisioning, deploying, orchestrating, and monitoring edge applications [19]. Another paper presents a pattern language for IoT applications [4], based largely on patterns from blog entries which are not comparable to our patterns in format or scope. Guth et al. compared several IoT Platform architectures to create an IoT reference architecture [16] to which our example system can be mapped. There are several publications that use patterns to design software architectures, such as [3,13]. But as the IoT includes a lot of physical things, our patterns are not only concerned with software, but also with the features of these things and how they shape and influence the IoT systems. There is also work by Falkenthal et al. which refines abstract patterns to patterns with more concrete, technology specific solution descriptions [9] or links them to concrete implementations [8,7]. This could be a next step to move our abstract IoT Patterns towards concrete solutions.

3 Motivating Example

One of the many domains where the IoT is applicable is industrial fabrication [10,11]. Here, as in many other domains, digitalization is advancing. This

[3] An up-to-date overview of all published IoT Patterns can be found on http://www.internetofthingspatterns.com

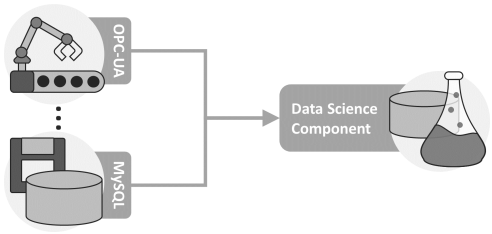


Fig. 1. Abstract high-level overview of the SePiA.Pro project.

has lead to the creating of various movement, such as Smart Factory, Industrial Internet, or Industry 4.0, which at their core have similar goals: to gain advanced insights and control of productions processes through wide spread digitalization and to further automate and optimize these processes.

Our example stems from the Industry 4.0 research project SePiA.Pro[4], which is situated in the domain of optimization [10]. Todays factories are comprised of many machines. By connecting several of these machines together, production lines are formed, which transform a part over several steps, for example by cutting, drilling, bending, welding, and painting. While single machines are often highly optimized, the overall optimization of production lines might be improved. The aim of SePiA.Pro is to build a self-service platform that enables analytics specialists to offer optimization services to large and small companies, which can instantiate these services on- or off-premise to optimize their existing machines and production lines with little or no technical expertise.

Figure 1 shows a very high level example of what this entails. On the left side, there are several data sources that provide the data which is used as input for the optimization services. These include data from the actual machines that should be optimized, but potentially also data from other sources, such as databases. The right-hand side of Figure 1 shows a *Data Science Component*, which takes this as input for analytical algorithms and produces a result that can be used to optimize machines and production lines. Of course there are several additional steps and obstacles between the left and right side that have to be looked at in more detail. For this we will use IoT Patterns.

4 IoT Patterns

The idea of patterns, which are abstract textual descriptions of proven solutions to reoccurring problems, goes back to Alexander's architecture patterns [1]. Since then, patterns have been published for all kinds of domains, also in the domain of IT [14,17]. In our previous work, we added to the already existing IT related patterns by publishing a collection of IoT Patterns [20,21,22]. Table 1 provides a brief overview of the IoT Patterns that are relevant in the context of this paper[3]. They include patterns concerned with device energy supply and device

[4] `http://projekt-sepiapro.de`

operation modes, as well as patterns concerned with communication and with data processing.

5 Applying IoT Patterns to Smart Factory Systems

Our motivating example shown in Figure 1 gives only a very general overview of what the SePiA.Pro project is trying to achieve. But by stepwise applying the existing IoT Patterns presented in Section 4, we can build this simple overview into a more full fledged architecture. We start on the left hand side of Figure 1 with the devices and other data sources.

5.1 Devices

The devices and potentially other data sources are the first point in the system where patterns can be applied. In some cases, these pattern can have a profound impact on the rest of the system architecture. There are two patterns that are applicable in this case, as shown in Figure 2.

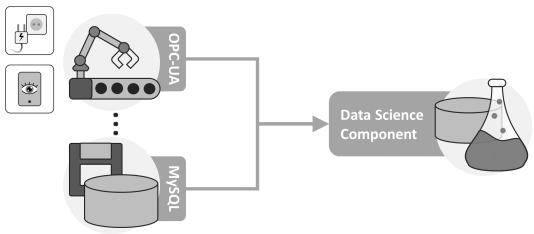
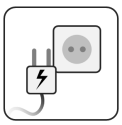
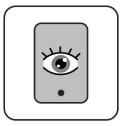
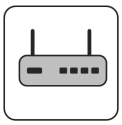
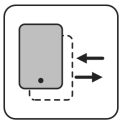
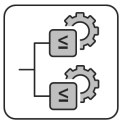


Fig. 2. Example with added Mains-Powered Device and Always-On Device.

One important question in IoT systems is how you provide energy to all the devices. In many systems you will have Period Energy-Limited Devices or Lifetime Energy-Limited Devices which are powered by batteries, or Energy-Harvesting Devices that gather their energy from their surroundings, for example with solar cells [22]. These kinds of devices often have a large impact on the overall system, as they are usually very constrained in their resources and only intermittently online [22]. In our case we are looking at large industrial machines that require a lot of power to operate and are therefore connected to the energy grid. Thus, they are Mains-Powered Devices (see Table 1) [22]. These kinds of devices have the advantage that they have all the power they need at their disposal and do not have to restrain themselves. On the other hand, they cannot be mobile and are depended on the power grid. In our case, this is not a problem and these devices do create no special problems regarding their energy supply.

Devices can use different operating modes to get the most out of the energy available to them. Device with limited energy are often Normally-Sleeping

Table 1. Short summary of the IoT Patterns used in this paper.

Icon	Description
	**Mains-Powered Device** Some devices have high energy requirements or are otherwise restricted so that powering them with batteries or energy harvesting is not an option. Connecting these devices to mains power provides them with plenty of energy [22].
	**Always-On Device** Some devices have to be constantly active and connected to fulfill their intended function or have virtually unlimited power available to them (e.g., Mains-Powered Devices). Leave these Always-On Devices connected and running at all times [22].
	**Device Gateway** Devices often differ in the communication technologies, protocols, or payload formats they use. Connect them to an existing network or system by using an intermediary Device Gateway that translates between the differrent communication methods [20].
	**Device Shadow** Devices go offline to save energy or because of network outages. Other components still want to interact with them. By storing a persistent virtual representation of devices and communicating through this copy only, other components can still work with offline devices [20].
	**Rules Engine** Throughout its operation a system receives a wide range of messages from devices and other components. A Rules Engine can react in different ways to these messages depending on their content, metadata, or additional external data sources. Each message is evaluated against a set of rules which trigger actions if they match [20].
	**Remote Processing** Some processing on the data produced by devices is very processing intensive, requires a lot of storage, or requires multiple data sources to be combined. Such processing steps may be too resource intensive for devices. Remote Processing runs the processing steps somewhere else, e.g., the Cloud, and returns the result to the originator.
	**Local Processing** Some situations require a fast reaction to sensor readings or other events. In such cases, first sending this data to Remote Processing components and then waiting for the answer may take too long. Local Processing integrates processing capabilities directly on or physically close to the devices where time-critical data is generated.

Devices which turn most of their components off for long periods of time in order to safe energy [22]. But in our case, as our industrial machines are used continuously to produce goods and are Mains-Powered Devices anyway, it makes little sense for them to sleep for long periods. Thus, they are Always-On Devices (see Table 1) [22] which, apart from high energy costs, bring only few disadvantages.

5.2 Communication

The patterns we applied so far had no impact on our overall system architecture. We continue with the middle part of our original overview in Figure 1, which is concerned with the communication between the data sources on the left and the *Data Science Component* on the right. Two patterns can be applied here as we described in the rest of this section.

As mentioned earlier, we can have all kinds of machines and other data sources which produce data in which we are interested for optimization purposes. But these data sources rarely use a single communication technology, protocol, or payload format. Industrial machines may use OPC-UA[5] or other industrial communication technologies, while other sources, for example databases, might be accessed via SQL[6]. This is the problem solved by the Device Gateway pattern (see Table 1) [20]. As shown in Figure 3, by adding a *Data Interface Unit*, which implements the Device Gateway pattern, we are able to translate different communication technologies so that they can be uniformly accessed by the *Data Science Component*.

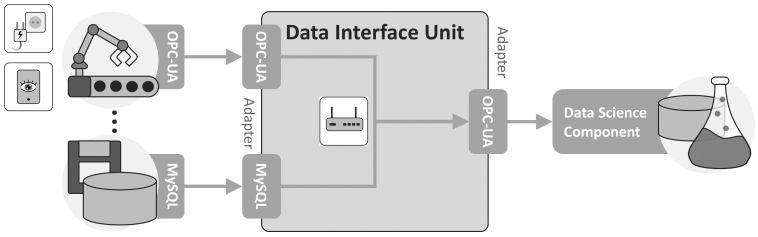


Fig. 3. Example with added Device Gateway pattern.

Although we have Mains-Powered Devices and Always-On Devices, there may be situations where they are unavailable, for example during maintenance or a power outage. Besides, some devices may not have a persistent storage for their data. In this case, a Device Shadow (see Table 1) helps as it stores the last known states and desired future states of all devices connected to it [20]. Thus, it is possible for other components, such as the *Data Science Component*

[5] https://opcfoundation.org/about/opc-technologies/opc-ua/
[6] https://www.iso.org/standard/63565.html

in our example, to access device data even if the device is currently not available. As can be seen in the middle of Figure 4, we added this functionality to the *Data Interface Unit* with the *Relay Service*.

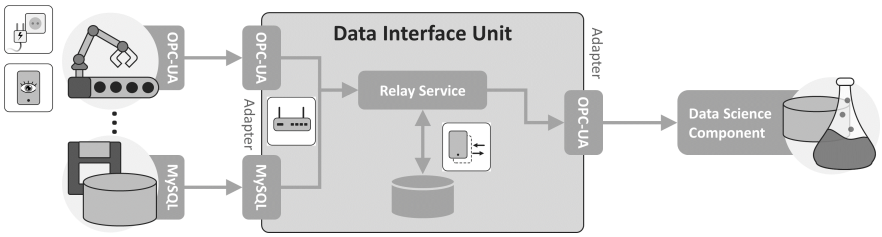


Fig. 4. Example with added Device Shadow pattern.

5.3 Processing

The communication patterns that we introduced in the last sections added new components to our example system which now allow us to communicate data from the different data sources to the *Data Science Component*. Now we turn our attention to the actual data processing. There are three patterns which can be applied in this area.

The algorithms and software needed for the analysis and optimization of machines and production lines vary depending on the use case. Thus, there is not only one *Data Science Component*, as shown in the previous figures. Instead, there are multiple different *Data Science Components* that can be selected. These are hosted on a common *Industrial Analytics Platform*, as shown in Figure 5. As an example, A Rules Engine (see Table 1) [20] could be one kind of *Data Science Component* or part of a *Data Science Component*.

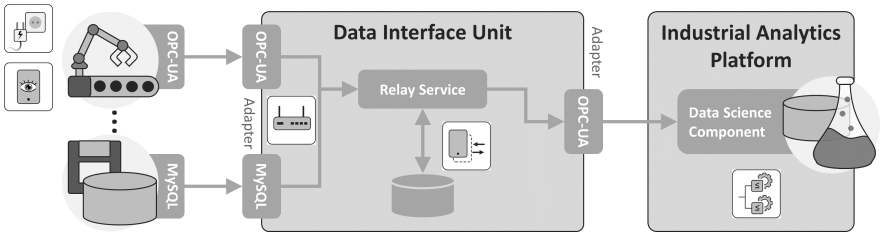


Fig. 5. Example with added Rules Engine pattern.

As mentioned earlier, the analytics capabilities provided by the SePiA.Pro platform should be usable for all kinds of small or large organizations. Some

of these do not have the required IT infrastructure or technical knowledge to run the software provided to them. For such cases, SePiA.Pro offers the option of Remote Processing (see Table 1), where the analytics software is hosted remotely in the cloud and the data produced by the machines and other data sources is transferred into the cloud for analysis. To support this scenario, the *Data Interface Unit* and the *Industrial Analytics Platform*, which includes a *Data Science Component*, are packaged as a Smart Service [11]. This Smart Service is then provisioned by the *Smart Service Provisioning Engine* into a remote environment, as shown in Figure 6.

There may also be organizations using the SePiA.Pro platform for which sending all their data to a remote cloud is not an option for security and privacy reasons. Besides, in cases where a lot of data is produced and should be analyzed, sending all this data to a remote location may not be practical because of bandwidth limitations, high latency, or high costs. For such cases, SePiA.Pro also implements the Local Processing patterns (see Table 1). Here, the *Smart Service Provisioning Engine* is used to provision the Smart Service at the local premises of the company, where they are in full control of their data and IT infrastructure.

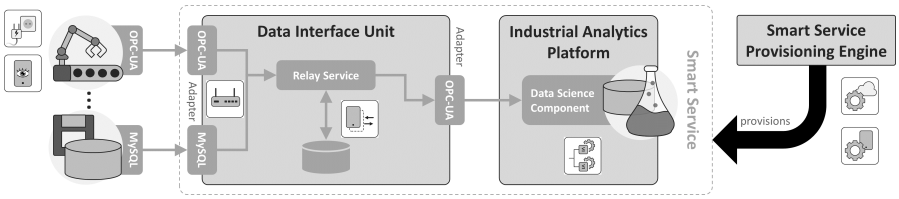


Fig. 6. The final example system after applying the Remote Processing and Local Processing patterns.

6 Conclusion

Industrial production and automation is one area where IoT will be relevant in the future. We have shown that our existing IoT Patterns can be applied in this area to better understand the consequences of choosing a particular solution, solve problems that appear in such systems, and provide an abstract architectural overview. In the future we work on adding more patterns to the already existing pattern catalog. One interesting area is security and privacy, where IoT Patterns would be of high relevance, especially in industrial scenarios such as the example described in this paper, where security and privacy is highly relevant. We are also planning to further refine the existing connections between the IoT Patterns into a pattern language, which gives IoT architects and designers more tools to find and apply the right patterns for their particular use case. Besides, we are working on methods that allow generic patterns to be refined to technology

specific patterns [9] which can be linked into solution languages [12]. This could provide IoT architects additional support when implementing IoT systems based on IoT Patterns.

Acknowledgments. This work was partially funded by the project SePiA.Pro (01MD16013F) of the BMWi program Smart Service World.

References

1. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (1977)
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer networks 54(15), 2787–2805 (2010)
3. Buschmann, F., Meunier, R., Rohnert, H., Sornmerlad, P., Stal, M.: Pattern-Oriented Software Architecture: A System of Patterns, vol. 1 (1996)
4. Chandra, G.S.: Pattern language for iot applications (2016)
5. Eloranta, V.P., Koskinen, J., Leppänen, M., Reijonen, V.: Designing distributed control systems: A pattern language approach. Wiley series in software design patterns, Wiley, Hoboken, NJ (2014)
6. Eloranta, V.P., Koskinen, J., Leppänen, M., Reijonen, V.: Patterns for the companion website (2014), `http://media.wiley.com/product_ancillary/55/11186941/DOWNLOAD/website_patterns.pdf`
7. Falkenthal, M., Barzen, J., Breitenbücher, U., Fehling, C., Leymann, F.: Efficient pattern application: Validating the concept of solution implementations in different domains. International Journal on Advances in Software 7(3&4), 710–726 (2014)
8. Falkenthal, M., Barzen, J., Breitenbücher, U., Fehling, C., Leymann, F.: From pattern languages to solution implementations. In: Proceedings of the Sixth International Conferences on Pervasive Patterns and Applications (PATTERNS 2014). pp. 12–21. IARIA, Wilmington, DE (2014)
9. Falkenthal, M., Barzen, J., Breitenbücher, U., Fehling, C., Leymann, F., Hadjakos, A., Hentschel, F., Schulze, H.: Leveraging pattern application via pattern refinement. In: Proceedings of the International Conference on Pursuit of Pattern Languages for Societal Change (PURPLSOC 2015)
10. Falkenthal, M., Breitenbücher, U., Christ, M., Endres, C., Kempa-Liehr, A.W., Leymann, F., Zimmermann, M.: Towards function and data shipping in manufacturing environments: How cloud technologies leverage the 4th industrial revolution. In: Proceedings of the 10th Advanced Summer School on Service Oriented Computing. pp. 16–25. IBM Research Report, IBM Research Report (2016)
11. Falkenthal, M., Breitenbücher, U., Képes, K., Leymann, F., Zimmermann, M., Christ, M., Neuffer, J., Braun, N., Kempa-Liehr, A.W.: Opentosca for the 4th industrial revolution: Automating the provisioning of analytics tools based on apache flink. In: Proceedings of the 6th International Conference on the Internet of Things. pp. 179–180. IoT'16, ACM (2016)
12. Falkenthal, M., Leymann, F.: Easing pattern application by means of solution languages. In: Proceedings of the Ninth International Conferences on Pervasive Patterns and Applications (PATTERNS) 2017. pp. 58–64. Xpert Publishing Services (2017)
13. Fernandez, E.B.: Security Patterns in Practice: Designing Secure Architectures Using Software Patterns. Wiley (2013)

14. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, Massachusetts (1995)
15. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (iot): A vision, architectural elements, and future directions. Future Generation Computer Systems 29(7), 1645–1660 (2013)
16. Guth, J., Breitenbücher, U., Falkenthal, M., Leymann, F., Reinfurt, L.: Comparison of iot platform architectures: A field study based on a reference architecture. In: Proceedings of the International Conference on Cloudification of the Internet of Things (CIoT). IEEE (2016)
17. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley, Boston, Massachusetts (2004)
18. Ishaq, I., Carels, D., Teklemariam, G., Hoebeke, J., Abeele, F., Poorter, E., Moerman, I., Demeester, P.: Ietf standardization in the field of the internet of things (iot): A survey. Journal of Sensor and Actuator Networks 2(2), 235–287 (2013)
19. Qanbari, S., Pezeshki, S., Raisi, R., Mahdizadeh, S., Rahimzadeh, R., Behinaein, N., Mahmoudi, F., Ayoubzadeh, S., Fazlali, P., Roshani, K., Yaghini, A., Amiri, M., Farivarmoheb, A., Zamani, A., Dustdar, S.: Iot design patterns: Computational constructs to design, build and engineer edge applications. In: Proceedings of the First International Conference on Internet-of-Things Design and Implementation (IoTDI). pp. 277–282. IEEE (2016)
20. Reinfurt, L., Breitenbücher, U., Falkenthal, M., Leymann, F., Riegg, A.: Internet of things patterns. In: Proceedings of the 21st European Conference on Pattern Languages of Programs (EuroPLoP). ACM (2016)
21. Reinfurt, L., Breitenbücher, U., Falkenthal, M., Leymann, F., Riegg, A.: Internet of things patterns for communication and management. LNCS Transactions on Pattern Languages of Programming (2017)
22. Reinfurt, L., Breitenbücher, U., Falkenthal, M., Leymann, F., Riegg, A.: Internet of things patterns for devices. In: Proceedings of the Ninth International Conferences on Pervasive Patterns and Applications (PATTERNS) 2017. pp. 117–126. Xpert Publishing Services (2017)
23. Singh, J., Pasquier, T., Bacon, J., Ko, H., Eyers, D.: Twenty security considerations for cloud-supported internet of things. IEEE Internet of Things Journal 3(3), 269–284 (2016)