SEPL: An IoT Platform for Value-added Services in the Energy Domain

Architectural Concept and Software Prototype

Theo Zschörnig[1], Robert Wehlitz[1], Ingo Rößner[1] and Bogdan Franczyk[2,3]
[1]*Institute for Applied Informatics (InfAI), Leipzig University, Hainstr. 11, 04109 Leipzig, Germany*
[2]*Information Systems Institute, Leipzig University, Grimmaische Str. 12, 04109 Leipzig, Germany*
[3]*Business Informatics Institute, Wrocław University of Economics, ul. Komandorska 118-120, 53-345 Wrocław, Poland*

Keywords: Internet of Things, Value-added Services, Platform Architecture.

Abstract: The Internet of Things (IoT) is based on ubiquitous smart devices equipped with sensors, actuators and tags which are connected to the Internet. Combining their sensing and actuation capabilities yields large possibilities for businesses to provide customers with value-added services in terms of energy management, entertainment, security and convenience. Businesses may use IoT data to develop innovative business models thus further increasing the value and usefulness of smart devices. Despite these potentials, providers of IoT platforms struggle to tackle some challenges which come with the design and operation of the platforms. In this paper, we propose an architectural concept which aims to bridge this gap and provides an integrated environment for smart device integration, data and analytics as well as IoT-aware processes. We also present the Smart Energy Platform (SEPL) as an instantiation of the proposed concept to evaluate it in terms of its functional feasibility and show that it addresses the issues current IoT platforms face.

1 INTRODUCTION

The Internet of Things (IoT) comprises smart devices equipped with sensors, actuators, and tags which are or attached to real-world objects and connected to the Internet (Han et al., 2016). The number of smart devices is estimated to reach 24 billion by the year 2020 (Greenough, 2016). This increase is accompanied by huge amounts of data which offer new possibilities for businesses to provide value-added services for their customers.

In the energy domain, current and future trends, such as smart metering, smart home, e-mobility and smart grid are summarized under the umbrella term smart energy (Aichele et al., 2013). In this context, smart devices, which provide sensing and actuation capabilities, are already used to gather energy data and respond to changing circumstances, e.g. in the case of demand-side management of energy consumption. IoT platforms are a means to interconnect distributed smart devices, use their exposed services to create more sophisticated services with added value and to build applications upon them. Against this background, previous research has found that current IoT platforms are lacking important features for device integration, data and analytics, as well as IoT-aware processes (Wehlitz et al., 2017).

In this light, the main contribution of this paper is an architectural concept for an IoT platform, which bridges this gap. We furthermore present a software prototype of a Smart Energy Platform (SEPL) as an instantiation of the proposed concept in order to evaluate its functional feasibility.

In this paper, we describe the motivation for conducting this research (Sect. 2). Following, we highlight the major challenges and state of the art for designing and operating IoT platforms (Sect. 3). In Sect. 4, we introduce our architectural concept, which is divided into the three technical layers: Integration, data and analytics, and IoT-aware processes. Afterwards, we present the SEPL software prototype to show in which way the components of each layer are implemented and how they address the identified challenges (Sect. 5). Finally, the paper concludes with a short summary and outlook (Sect. 6).

Zschörnig, T., Wehlitz, R., Rößner, I. and Franczyk, B.
SEPL: An IoT Platform for Value-added Services in the Energy Domain.
DOI: 10.5220/0006695205930600
In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018)*, pages 593-600
ISBN: 978-989-758-298-1

2 MOTIVATION

The growing number of smart devices offers many benefits for people in different areas of their everyday life, such as energy management, entertainment, security, and convenience (Zion Market Research, 2017). However, smart devices, in comparison to regular devices, do not offer a notable advantage in possibilities if the provided data and actuation capabilities are not utilized. Therefore, services which use the aforementioned capabilities are required in order to create customer value around products like smart meters, smart thermostats, smart lights, etc.

Especially in the energy domain, utilities have to develop new service offers and business models due to low energy prices and high dues. In this context, expert interviews revealed that utilities' margin for electricity supply in Germany is between 10 and 20 euros per household per year. Hence, with the comprehensive rollout of smart meters and the increasing adoption of smart devices, such businesses have the chance to provide services which, in addition to the pure energy supply, add value to their customers and enable new sources of income. Possible fields of application could be, but should not be limited to, the increase of energy consumption transparency, energy savings, and energy efficiency in households.

Against this background, we found that IoT platforms are an important building block to provide the infrastructure and software tools necessary to develop and run value-added services in the energy domain. They can be used to integrate smart devices of customers, e.g. smart meters, process and analyse sensing data, e.g. energy consumption values, and create more sophisticated services by service composition and individualisation, for instance, to control household appliances in an energy efficient manner.

3 STATE OF THE ART

Recent studies analyzing IoT platforms have identified multiple challenges for their design and operation. These are usually separated into a functional and non-functional dimension (Wehlitz et al., 2017).

On the side of the functional challenges, data management, service abstraction, marketplace functionality, and device and service discovery are most significant.

The area of non-functional challenges for IoT platforms contains interoperability, privacy and security, scalability, quality of service and context awareness.

With regard to these challenges, there are many existing approaches from businesses, the open source community as well as researchers which have different degrees of coverage (Wehlitz et al., 2017). Concerning the functional challenges, most platforms use service abstraction for providing a unified access to smart devices. Data management and device and service discovery are implemented only partly by some platforms. More importantly, a marketplace is missing in almost all of them.

On the side of the non-functional challenges, context awareness is implemented by almost all platforms. Yet, the integration of smart devices of different vendors with distinct service interfaces, i.e. interoperability, is supported only in part. Also, the bulk of the regarded platforms do not fully support scalability which seems to be a major issue with rising smart device numbers. Quality of service and privacy and security are also challenges which are only partly solved (Wehlitz et al., 2017).

It is noteworthy, that two platforms we have studied, FIWARE and ThingWorx, address most of the found challenges. However, both are extensive systems, exhibiting high complexity, a steep learning curve or they are proprietary. Hence, it seems plausible that they are not suitable to be used directly by customers.

4 PLATFORM ARCHITECTURE

Looking at the challenges and the state of the art as described in Sect. 3, we found the basic design of an IoT platform should comprise three layers:

- An integration layer providing bidirectional platform-to-device communication as well as tools to unify device and service access.
- A data and analytics layer providing smart device data management, but also means to analyze and filter data.
- An IoT-aware process layer to enable the composition and orchestration of IoT and analytics services into more useful services and applications.

Additionally, a marketplace should be offered to platform users allowing them to share created content in order to bring the platform to life. Another important design goal for us is to make the entire platform loosely coupled and easily extensible. This requires a unified, resilient way of communication

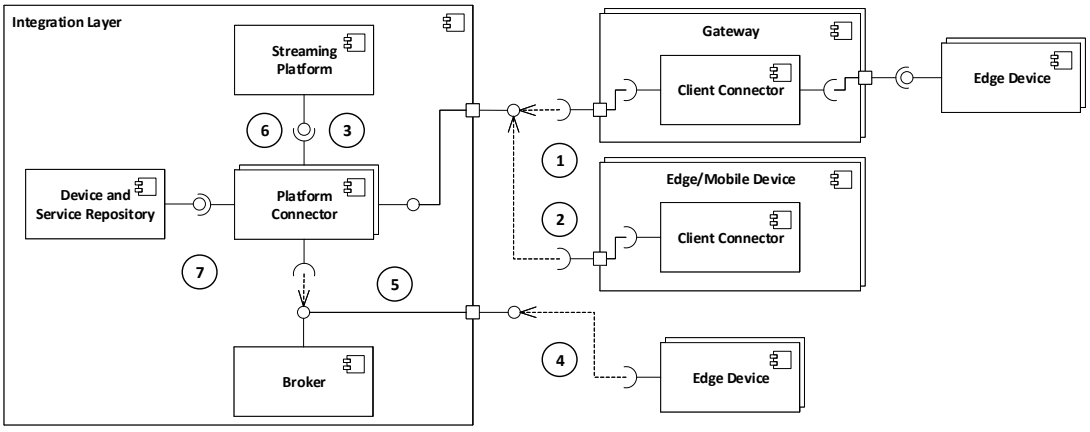

Figure 1: Components of the integration layer and their interactions.

between the different system components. The proposed concept employs a *streaming platform* to achieve this.

4.1 Integration Layer

The integration layer is the foundation of the proposed platform concept. It acts as the main point of integration for different kinds of smart devices. Therefore, the layer must provide interfaces for them to send and receive data.

In the context of smart home and smart building environments, smart devices are often connected to a local gateway (Risteska Stojkoska and Trivodaliev, 2017). In this regard, the platform operator needs to provide either best practices for end users on how to configure their gateway device or, more commonly, find ways to enable inbound network traffic automatically. In this regard, they have to provide software for different gateway types in order to register locally connected smart devices on IoT platforms.

As shown in Figure 1, we use *platform connectors* in conjunction with *client connectors* to achieve this. *Client connectors* are deployed either on gateways (1) or on stand-alone smart devices (2). Supporting various message protocols (e.g. HTTP, WebSocket, CoAP), they send data to their counterparts, i.e. *platform connectors*. These receive the data and push it to the central *streaming platform* (3) for further processing by other IoT platform services. Publish-subscribe communication (e.g. based on MQTT) (4) via an intermediary broker (5) is also possible. Besides the actual communication with smart devices, *platform connectors* gather historical data about connection states of *client connectors*, e.g. time of disconnect. They also handle the outgoing data, meaning service requests, to the *client connectors*. These requests are pushed to the *streaming platform* by IoT platform services and applications. The *platform connectors* interface the *streaming platform*, fetch new service requests (6), and proceed in sending the data to the destined *client connector*. In addition, *platform connectors* register, unregister and update device and service instance data in the *device and service repository* (7). It contains device and service type descriptions as well as device instances with their service instantiations.

Since the IoT domain has not yet undergone successful standardization efforts (Díaz et al., 2016), it is characterized by a large heterogeneity in terms of data types, semantics and protocols, which complicates access to their services (Perumal et al., 2015; Díaz et al., 2016). Therefore, it seems appropriate to transform different data types and structures as well as semantics to a unified basis. In our proposed concept, this is done by the *platform connectors*. They map data structures of smart devices to a generic data model, thus enabling platform wide understandability and reusability across different services and applications.
In this model, a device type is a collection of comparable devices (e.g. *Temperature Sensor*), which contains services (e.g. *getTemperature*). Transforming data structures of smart devices into the generic data model ensures that all data, which is queued in the *streaming platform*, is already unified. This approach also leads to reusability of value types, since they are used frequently to create services in the platform environment. Also, data structures using this data model are independent from specific serialization formats. Hence, the layers on top of the integration layer are able to create processes and applications with high reusability across smart devices.

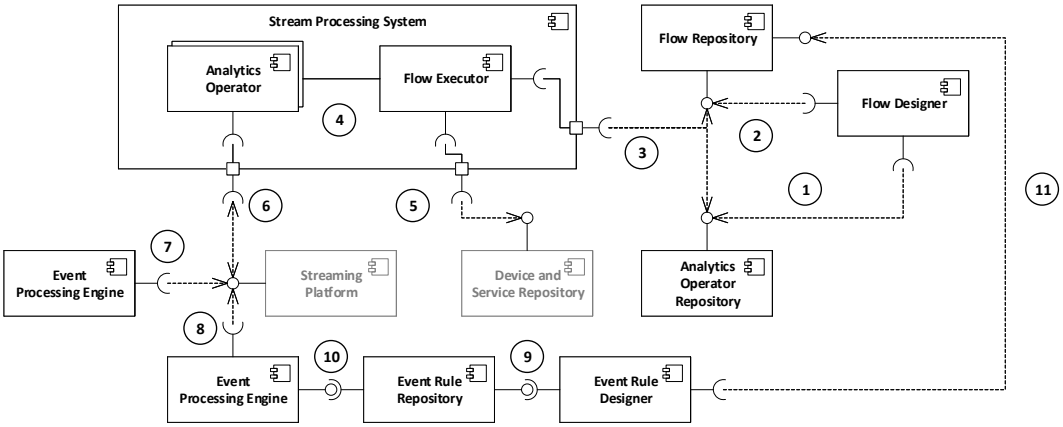

Figure 2: Components of the data & analytics layer and their interactions.

4.2 Data & Analytics Layer

The data and analytics layer is on top of the integration layer and provides sophisticated data manipulation, analytics, persistence and event processing functionalities. Smart device data usually arrives in data streams and is characterized as time series data, thus creating the need for real-time processing (Pawar and Attar, 2016). While the batch processing of historical data combined with the application of prediction models on this data is already established, real-time processing of smart device data enables the detection of causable relationships between the devices themselves and their environments. This may lead to the autonomous adaption to new situations (Stolpe et al., 2016). As a result, the analytics layer should be a lightweight *stream processing system* which is able to easily access the central *streaming platform* with the smart device data, as shown in Figure 2. Also, it is necessary to provide capabilities to design analytics flows for large amounts of different analytics problems.

In order to process data, we utilize *analytics operators* which perform analytics actions on data streams (e.g. *Temperature Unit Conversion*). These operators are lightweight and perform a single task. They may be composed into complex analytics flows, thus providing the needed flexibility in analytics design. *Analytics operator* metadata is stored in the *analytics operator repository* and comprises inputs (e.g. *Temperature #1*: float, *Unit*: string), outputs (e.g. *ConvertedTemperature*: float), and configuration parameters (e.g. *ConversionUnit*: string).

A *flow designer* is used to graphically design and orchestrate input streams and *analytics operators* into analytics flows, using operator metadata (1). Finished analytics flows (e.g. *Temperature Unit Conversion and Mean Value*) are saved into the *flow repository* (2) and may contain multiple *analytics operators* and data streams as input. A flow exposes inputs (e.g. *Temperature #1*: float, *Unit #1*: string, *Temperature #2*: float, *Unit #2*: string), outputs (e.g. *MeanTemperature*: float) and configuration items (e.g. *ConversionUnit*: string) for all *analytics operators* used. Analytics flows are started (3) and monitored (4) using a flow executor. In addition, the flows are linked to smart device services data (5) as input streams.

The *analytics operators* pull the data from the *streaming platform* (*input topic #1*), process it and push it back (*output topic #1*) (6). *Analytics operators* linked to them in an analytics flow, pull their input data from the data pushed back (*output topic #1*) and continue in the same manner as the previous *analytics operators.* In order to achieve data persistence, all data streams need to be pulled from the *streaming platform* into a *Data Lake* (7). Following the Data Lake paradigm, they are saved as they are without further processing. Once the saved data is needed by platform services, it is pushed back into the *streaming platform*.

Another key aspect of the data and analytics layer is event processing. An *event processing engine* is consuming data streams and triggers events if defined event rules are met (8). We describe events at type (e.g. *value > 22*) and instance level (e.g. *temperature value from Temperature Sensor #1 > 22 °C*). Events are defined using an *event rule designer* and event rules are stored in the *event rule repository* (9). The *event processing engine* listens for all event rules (10). These may be applied to all data streams present in the *streaming platform* (e.g. smart device data, analytics data, process data, etc.). In this regard, the *event rule designer* may also access the analytics flow *repository* to expose analytics flows to the user which may be used to listen on for event detection (11).

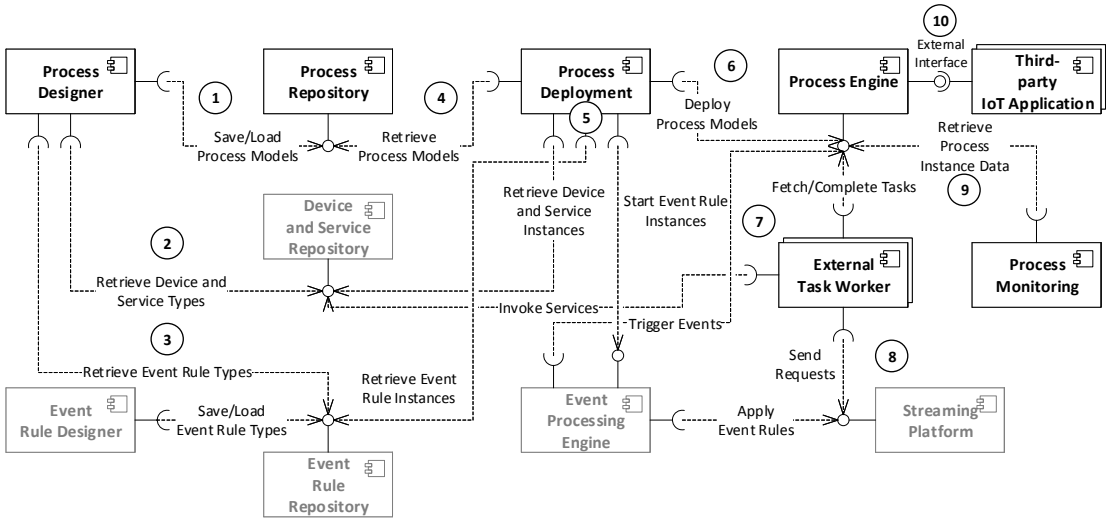

Figure 3: Components of the IoT-aware process layer and their interactions.

4.3 IoT-Aware Process Layer

The IoT-aware process layer bundles the functionalities of the integration as well as the data and analytics layer. It serves the orchestration of IoT services, analytics and event data along businesses, smart devices and customers. The layer is based on a centralized orchestration model and allows for the modelling, implementation and execution of IoT-aware processes.

As illustrated in Figure 3, processes are designed using a *process designer*, which includes a graphical user interface and stores created process models in the *process repository* (1). It has access to device and service types from the *device and service repository* (2) together with event rule types from the *event rule repository* (3). Hence, processes are designed using abstract device, service and event definitions. This allows for processes which are instance-independent and therefore easily reusable. In addition, the same model can be deployed for different smart devices multiple times. Executable process models are deployed by the *process deployment* component. It pulls model definitions from the *process repository* (4) and scans it for references to device/service and event types. If any are present, it retrieves compatible instances from the *device and service repository* (5). User input is required to select which concrete devices and services shall be used for a process deployment. Event types are handled in the same manner. After this, the process model is deployed to the *process engine*, which ensures proper execution (6). *External task workers* execute service tasks. Once the *process engine* encounters a service task (e.g. smart device actuation), it creates a new job for the *external task workers* to be pulled and executed. When they completed their jobs, they notify the *process engine* which proceeds to the next step (7). In case of an actuation service, an *external task worker* fetches the required service description from the *device and service repository* and sends a service request to the *streaming platform* (8), which in turn is accessed by *platform connectors*. A *platform connector* forwards the service request to the *client connector* which executes it or relays it to the actual smart device.

Sensing requests get relayed back from the device to the *client connector* to the *platform connector* which pushes the sensing data to the *streaming platform*. From there, events are, if applicable, applied by the *event processing engine* on the data stream. The *process engine* is able to process fired events and act according to the executed process model (e.g. to start a new or to affect the control flow of a running process instance). The *process engine* is supervised by the *process monitoring* component, which provides insights to historical and operational data of process instances (9). Finally, the execution of processes may be integrated in external applications using a REST API (e.g. a mobile application to trigger new process instances) (10).

4.4 Marketplace

The marketplace allows customers to share their data, modeled IoT-aware processes, analytics flows and operators with other platform users (businesses and customers). All content created by customers and businesses as well as data streams originating from smart devices may be accessed as defined in the

permission system. This system links user instances to processes, flows, operators, and device instances from the respective repositories. Customers and businesses are able to share their content and device data and publish them at the marketplace. The access to the shared entities may be limited to individual customers or businesses or groups of them. Also, a fee for using the entities may be applied. If other customers or businesses decide to subscribe to content or device instances, the required permissions are set in the permission system.

Granting permission to content allows the subscriber to use the flow, operator or process with their own smart devices. In case of a shared smart device, the subscriber may access emitted data from the device as set by the owner. In this context, using the actuation capabilities of a shared device is generally forbidden because of security concerns.

5 PROTOTYPE

In order to provide a proof of concept of our architecture presented in Sect. 4, we implemented all system components as an integrated cloud-based software prototype, the SEPL. Its purpose is to provide platform tools to customers, to integrate their smart devices, analyze their data and develop and use IoT-aware processes for energy management scenarios. Also, customers may share their data with businesses, which in turn are provided with tools to develop value-added services around this data, creating surplus value.

The platform architecture is based on the concept of Microservices, meaning that functionalities are encapsulated as single services by software containers, in our case using Docker (https://www.docker.com/). This provides easy scalability of services which undergo heavy usage and enables easy change or substitution of services with changed requirements. Because of this, we employed Apache Kafka as a central *streaming platform* (https://kafka.apache.org/). To ensure usability, we also developed a front-end application using AngularJS which provides access to important platform tools. User authentication and authorization is ensured using the API gateway Kong (https://getkong.org/) in conjunction with a *user repository* and *permission system*. In order to further strengthen privacy and security we followed the "Privacy by Design" and security principles as pointed out in scientific literature (Schaar, 2010; Dougherty et al., 2009).

Because of the complex structure and the Microservice approach of the platform architecture, it is necessary to employ a powerful container orchestration and scheduling service. We use Rancher (http://rancher.com/) as it is easy to use and offers many out-of-the box features for platform operation.

5.1 Integration Layer

The main focus when implementing the integration layer, was to provide interfaces to enable communication between smart devices and the SEPL, but also to unify device and service descriptions from different vendors. Although the layer components are designed to be easily extendable in terms of protocol and device integration, a lot of the smart devices we use in our laboratory environment for testing our approach support the wireless communication standards ZigBee and Z-Wave. Both are very common and well suited for home energy management scenarios (Zhao et al., 2016).

The *client connectors* in smart home and smart building environments are provided in two ways: As plugins for already existing IoT gateways and as software libraries to be used for single smart devices by more advanced users. When deployed at IoT gateways, it is possible to use existing resources of them to search for locally connected smart devices and send this information to the SEPL, as the foundation for their integration.

The *platform connector* is scalable in order to being able to communicate with huge numbers of smart devices. It uses the device and service descriptions from the *device and service repository* to transform data from *client connectors* to the generic data model as explained in Section 4.1.

The *device and service repository* is based on Virtuoso (https://virtuoso.openlinksw.com/) and utilizes RDF for schema description. Other services can access the data from the *device and service repository* using a REST interface. Since it also stores all registered device instances, it enables platform services and users to discover smart devices and their services. Internally, it queries the RDF database using SPARQL, therefore enabling powerful queries.

5.2 Data & Analytics Layer

The analytics concept we employ is based on the Kappa architecture, meaning all data is handled as a stream. Therefore, *analytics operators* use the Kafka Streams (https://kafka.apache.org/) library to integrate with the streaming platform and are written as individual Microservices, which are encapsulated

using Docker. Using JAVA or Python, it is possible to create powerful data analytics as well as machine learning algorithms for processing. This allows for flexible *analytics operators* which offer the same capabilities as stream processing engines like Apache Spark Streaming or Apache Storm. In addition, this approach helps in reducing the programming overhead caused by changing data models and analytics requirements, compared to a Lambda architecture (Zschörnig et al., 2017).

New analytics flows are created with the help of the *analytics designer*, which interfaces the *analytics operator repository*. In our prototype, the designer utilizes a flow chart visualization to enable users to design new analytics flows, which are saved in the *flow repository*.

The request to start an analytics flow is send to the *flow executor*, which checks it for errors. If the flow is valid, it starts the *analytics operator* Docker containers in the required order with the necessary input, output and configuration values. The *analytics operator* containers are deployed on the underlying orchestration and scheduling environment. Data persistence is achieved by pulling all data streams from Apache Kafka into the *Data Lake* which uses a HDFS environment. Platform services may request data from the *Data Lake*, which pushes it back into Apache Kafka.

Event rules are stored in the *event rule repository* and are created using a JSON document which contains information on the data stream to be monitored and the actual rule. A rule is written by combining service fields from the *device and service repository* and logical expressions using the *event rule designer*.

5.3 IoT-Aware Process Layer

The *process designer* component of our software prototype is based on bpmn.io, a BPMN 2.0 rendering toolkit and web modeler (https://bpmn.io), which was extended to integrate with the *device and service repository*. We selected BPMN 2.0 because it seems to be the best suited modeling language for mapping IoT concepts in processes (Meyer et al., 2011). Following Meyer et al., 2013, our solution intends that IoT-aware processes are modeled within BPMN pools. The lanes of a pool represent device types. Each service task that is placed on a lane is assigned to and handled by the respective device type.

In case of a new deployment, the *process deployment* component analyses the process definition and prompts the user to select concrete instances for every found event and/or device type. It then deploys the process model to the *process engine*, which is based on the open-source platform Camunda (https://camunda.org). This supports the external task pattern and, hence, provides more flexibility and scalability for executing service tasks. The implementation of *external task workers* is independent from a specific programing language and additional source code for service task execution has not to be deployed to and executed by the *process engine*. Additionally, in conjunction with the *event processing engine* and the *stream processing system*, the *process engine* enables context awareness. In this regard, the *stream processing system* provides new insights into smart device data and the *event processing engine* is able to define actions based on changing parameters. IoT-aware processes may include this information, thus allowing smart devices to be "aware" of their surroundings and adapt to changing environments.

5.4 Marketplace

The main objective of the marketplace is to allow users (customers and businesses) of the SEPL to easily share and subscribe to processes, analytics operators and flows. In this regard, users of the SEPL may define if a process, analytics flow or operator may be published at the marketplace or only be shared with a single user or a group of users. If it is published at the marketplace, a request is sent to its repository, which then lists it in the appropriate category. This categorization is based on the type of the shared entity (e.g. process vs. operator) and its characteristics (e.g. inputs, outputs, purpose, etc.).

In conclusion, the marketplace offers various capabilities to share key entities of the SEPL, thus enabling device integration through sharing, but most important it is the key component to enable businesses to apply new business models around smart devices in the energy domain.

6 CONCLUSION & OUTLOOK

In this paper, we presented an architectural concept for the design and operation of IoT platforms. Customers may use its tools to integrate and orchestrate their smart devices and gain insights into their data, whereas businesses use them to create new applications and value-added services on top of the customer-specific data.

The proposed concept takes into account functional and non-functional challenges which were identified by current research and are not met entirely

by existing solutions. Looking at these challenges, we found that an IoT platform should comprise technical layers for device and service integration and abstraction as well as their discovery, but also capabilities for data management and analytics and IoT-aware process modeling, implementation and execution. Furthermore, customers and businesses need to be able to share device data access and processes with other users of the platform through a marketplace. The composition of all these platform components leads to quality of service regarding the communication with smart devices and context awareness in terms of their environment.

Future research in this field needs to focus on the advanced use of semantic technologies for device integration as well as device and service discovery. Analytics architectures have to be developed to cater to a non-technical audience, allowing self-service analytics. In terms of IoT-aware processes, tools for adaptive case management should be investigated to improve context awareness and flexibility. Also, business models on how to operate this kind of platform need to be created and evaluated. Finally, the research concerning hybrid approaches on IoT platforms, e.g. in conjunction with Fog Computing, needs to be deepened.

ACKNOWLEDGEMENTS

The work presented in this paper is partly funded by the European Regional Development Fund (ERDF) and the Free State of Saxony (Sächsische Aufbaubank — SAB)

REFERENCES

Aichele, C., & Doleski, O. D. (Eds.). (2013). *Smart Meter Rollout - Praxisleitfaden zur Ausbringung intelligenter Zähler*. Wiesbaden: Springer Vieweg.

Díaz, M., Martín, C., & Rubio, B. (2016). State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. *Journal of Network and Computer Applications*, 67, (pp. 99–117).

Dougherty, C., Sayre, K., Seacord, R. C., Svoboda, D., & Togashi, K. (2009). Secure Design Patterns. Retrieved from http://www.dtic.mil/get-tr-doc/pdf?AD=ADA636498

Greenough, J. (2016). How the 'Internet of Things' will impact consumers, businesses, and governments in 2016 and beyond. Retrieved from http://www.businessinsider.com/how-the-internet-of-things-market-will-grow-2014-10?IR=

Han, S. N., Khan, I., Lee, G. M., Crespi, N., & Glitho, R. H. (2016). Service composition for IP smart object using realtime Web protocols: Concept and research challenges. *Computer Standards & Interfaces,* 43, (pp. 79–90).

Meyer, S., Ruppen, A., & Magerkurth, C. (2013). Internet of Things-Aware Process Modeling: Integrating IoT Devices as Business Process Resources. In D. Hutchison, T. Kanade, Ó. Pastor (Eds.), *Lecture Notes in Computer Science. Advanced Information Systems Engineering* (Vol. 7908, pp. 84–98).

Meyer, S., Sperner, K., Magerkurth, C., & Pasquier, J. (2011). Towards modeling real-world aware business processes. In D. Guinard, V. Trifa, & E. Wilde (Eds.), In: *The Second International Workshop on Web of Things* (p. 1).

Pawar, K., & Attar, V. (2016). A survey on Data Analytic Platforms for Internet of Things. In *CAST-2016. 19-21 December 2016* (pp. 605–610). Piscataway, NJ: IEEE.

Perumal, T., Datta, S. K., & Bonnet, C. (2015). IoT device management framework for smart home scenarios. In *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)* (pp. 54–55).

Risteska Stojkoska, B. L., & Trivodaliev, K. V. (2017). A review of Internet of Things for smart home: Challenges and solutions. *Journal of Cleaner Production, 140, Part 3*, (pp. 1454–1464).

Schaar, P. (2010). Privacy by Design. *Identity in the Information Society, 3(2)*, (pp. 267–274).

Stolpe, M. (2016). The Internet of Things: Opportunities and Challenges for Distributed Data Analysis. *ACM SIGKDD Explorations Newsletter*, 18(1), (pp. 15–34).

Wehlitz, R., Häberlein, D., Zschörnig, T., & Franczyk, B. (2017). A Smart Energy Platform for the Internet of Things-Motivation, Challenges, and Solution Proposal. In *Business Information Systems: 20th International Conference, BIS 2017, Poznan, Poland, June 28-30, 2017, Proceedings* (Vol. 288, p. 271).

Zhao, Z., Agbossou, K., & Cardenas, A. (2016). Connectivity for Home Energy Management applications. In *APPEEC 2016. 2016 IEEE PES Asia Pacific Power and Energy Engineering Conference: October 25-28, 2016, Xi'an, China* (pp. 2175–2180).

Zion Market Research. (2017). Global Smart Home Market is Set for a Rapid Growth and is Expected to Reach around USD 53.45 Billion by 2022. Retrieved from https://www.zionmarketresearch.com/news/smart-home-market

Zschörnig, T., Wehlitz, R., & Franczyk, B. (2017). A Personal Analytics Platform for the Internet of Things: Implementing Kappa Architecture with Microservice-based Stream Processing. In *Proceedings of the 19th International Conference on Enterprise Information Systems* (pp. 733–738).