An Approach for a Mutual Integration of the Web of Things with Business Processes

Andreas Ruppen[1], Sonja Meyer[1,2]

[1] Software Engineering Group, University of Fribourg, Switzerland
[2] SAP Research, Switzerland
{andreas.ruppen, sonja.meyer}@unifr.ch

Abstract. The vision of a multitude of heterogeneous, connected devices interacting with parts of its physical environment, known as the Internet of Things (IoT), has gained momentum over the last years. For traditional business information systems the integration of the IoT leads to the emergence of new value-added business processes that make use of its representative devices such as RFID, sensors and actuators, as entirely new resources. One promising approach in the IoT domain is a real-world service integration method based on Representational State Transfer (REST) principles expedited by the initiative Web of Things (WoT). The connection of classical Enterprise Resource Planning (ERP) systems commonly based on service oriented architectures coming with heavy-weight services and the resource oriented WoT coming with RESTful services for its limited devices is still a complex problem on both sides. In this paper, we propose a bi-directional integration approach of current Business Process Management (BPM)-based ERP systems and the WoT to provide a foundation to connect and analyze legacy systems as well as Future Internet applications. By following a three phase BPM-lifecycle, we investigate how the process metamodel of the industry standard Business Process Model and Notation (BPMN) can cover WoT specific components. Reversely, we examine how existing and useful business process information can be seamlessly included into the WoT respecting its essential architecture and applying a component-based approach.

Keywords: Web of Things, Business Process Modeling, Resources, REST, BPMN, BPM, Things, Sensors, Devices

1 Introduction and Motivation

The history of the Internet of Things goes back to RFID sensors, Tags and Sensor Networks. Tracking objects over different locations and the resulting increase of comfort in supply chain management was one of the initial ideas for the IoT. Yet, research and industry quickly discovered that the IoT is not limited to tracking objects. Rather, it can be adapted to all sorts of scenario where a physical property (be it an object like for the RFID scenarios or something else) has to be made available to be consumed by machines. For a long time, research and industry have tried to

connect together small devices with limited capabilities. However, such devices only became popular and available for the mass in the last decade. This advances were made possible through technological progress in the hardware. Nowadays these devices have enough computing power to solve a comprehensive range of problems and are still affordable, especially when using large quantities of these devices.

A prominent problem regarding the IoT is its heterogeneity. No standards are imposed to define interfaces to smart devices. The palette for interacting with such devices ranges from fully blown SOAP services [1] to some less common protocols (ZigBee, SunSPOTs). The Web of Things (WoT) tries to bridge this gap by bringing standards to this world. Endorsing factors of the WoT are RESTful Web services. REST takes the receipts and approaches, which made the web popular and brings it to the service world. Maybe the most important part is a common standard protocol, HTTP. HTTP allows browsing the entire web without any restrictions. Besides, it is simple to use. A browser is sufficient to flip through the World Wide Web. Fielding was the first to describe REST architectures in his PhD [2]. His approach found wide adoption in the WoT community. REST and RESTful Web services are based upon four architectural principles, Richardson and Ruby [4] sum them up as follows: (1) **Addressability.** Resources are anything that can be targeted by a hyperlink. Besides, every resource is uniquely identified by a hyperlink. (2) **Statelessness**. The server does not keep any state information about a client. He transmits the state in each request. This allows isolated requests and better error recovery. (3) **Connectedness**. Each resource contains hyperlinks to connected resources. A client discovers new resources through this mechanism. The web heavily uses the connectedness constraints by providing links to related content. (4) **Uniform Interface**. Requests to the service are made with one of the four HTTP methods (GET, POST, PUT and DELETE). Each of the four methods having its well defined semantic. Some of these methods are called save. They don't modify any resource. Others are called idempotent, they can be executed more than once without changing the outcome. In this classification, DELETE, used for deleting resources, is whether safe nor idempotent, whereas GET, used for retrieving a representation of a resource, and PUT, used for updating a resource are safe and idempotent. Guinard et al. showed the potential this architecture has for the WoT [3]. Richardson and Ruby [4] provide some examples and case studies to introduce best-practice for software engineering in this domain.

With the growth of the community, grows also the complexity of the involved scenarios. Where in the early years use-cases were restricted to simple smart devices and smart actuators they nowadays cover broader scenarios. They range from complex delayed services [5] to fully integrated business processes. Yet the implementation becomes far more complex. On the other hand, SOAP and BPMN are de facto standards for business process execution and modeling in an enterprise environment [6]. By that they are out of the scope of the WoT. Bridging the gap between the two worlds is important to leverage the power delivered by such systems to the WoT. Not only the WoT can benefit from accessing business process to start new use-cases from the scratch but more important, bridging the gap between business processes and the Web of Things gives access to plenty of already existing, stable and approved components to the latter.

This paper is structured in two parts. In the fist one, we motivate why business process have to take care of the WoT and show how this can be achieved. In the second part, we show how the WoT can benefit from an integration of business process into the WoT. The rest of the paper will be structured as follows. In Section 2 we give a short overview of the field of interest. We show existing approaches to the WoT and business process and briefly discuss them. In Section 3 we present a model for the integration of business processes and the WoT. Finally we present a short outlook on future research.

2 Background Information and Related Work

2.1 Web of Things – Definition and Terminology

With the widespread of cheap connected devices full of sensors like Arduino boards[1] appeared the need for architectures to make such connected devices talk to each other. Smart objects are the building block of the IoT [7]. It is foreseeable that in a near future there will be more devices taking part in the Internet as humans. For this reasons it is important to find ways to connect these devices together in an easily exploitable way. After the tsunami over Japan in 2011, the world could for the first time assist to the impact of connected smart objects. Projects like *Wind from Fukushima* and platforms like COSM rapidly spread on the web, giving access to a multitude of Geiger counter. It appeared that these measures were more accurate and more timely than the values provided by the government.

Such sensors boards, also called smart objects, have only limited resources regarding memory, CPU and battery. It is therefore important to find lightweight protocols. Several protocols were proposed over the past few years to make these smart devices connect to each other. Among them, an implementation of the IPv6 Stack for low-power wireless networks [8] and [9], and dedicated low-power radio communications [10]. Some even proposed to port the full WS-* stack on smart devices [1].

In parallel to these efforts a new way of connecting smart devices emerged. It is based on the REST architectural style, proposed by R. Fielding [2]. REST is a resource driven architecture. A resource is everything a client can communicate with, like a temperature sensor or a NFC reader. Therefore, everything important enough for a given scenario will be modeled as a resource. Such resources are accessed through a uniform interface defining the action executed on the resource. The most prominent realization of such a uniform interface is HTTP. By that it is possible to gather the temperature reading from a thermistor by issuing a GET request to the URI associated with this resource. Furthermore REST dictates principles like statelessness and addressability. Both are of great value in the domain of smart devices. The statelessness implies that a smart device does not need to track the current session. This makes its software lighter. Additionally, addressability allow the easy identification of a given resources and further allows sharing and bookmarking of

[1] http://www.arduino.cc/

them. Bringing the two worlds, the Internet of Things and the REST architectures, together, leads to the Web of Things [11].

The WoT is a web where smart objects are treated as first class citizens. A Thing, can be anything, ranging from RFID sensors, to door actuators. By that, a Thing is a smart object. It is an every day object augmented with some communication capabilities. Thus, a Thing is twofold: It has a physical manifestation, the sensor or actuator but also a virtual one. Interacting with the physical side leads to the same result as interacting with the virtual one, the resource. The physical part of a Thing is what we are used to interact with. For a thermistor, for example, this means reading the actual measured temperature on some scale or small monitor. On the other hand, the virtual part is accessed over a RESTful Web service. In the example above, this Web service would serve the actual temperature readings delivering by that the same information as the scale in the physical world. This interaction is done over resources. Each Thing can host one or more resources, each corresponding to one entity of interest to the client. Whereas resources can be clearly differentiated from the virtual side, they do not always from the physical one. A weather station, as they are available in most households, is a prominent example for this difference. Such a station informs the owner about the actual temperature, the humidity and some more values. They all show up on some small display. Generally the user sees the whole as one single object. On the virtual side however, we can clearly distinguish the available resources. There would be for example a resource for the temperature readings and another one for the humidity reading.

Whereas much research has been done on how to connect smart devices together, less research was done on how to integrate such Things with other services and into already existing processes. More recently researchers begun studying this domain and propose several approaches. In [12] the authors propose a high level abstraction from Things. They argue that the raw information produced by smart devices is just too huge to be processed by humans. They provide an easy way to integrate linked algorithms to distill the information. In [13] the authors describe the integration of smart devices in the WoT thus, allowing forming complex business processes out of smart devices.

For the past few years many ways for connecting smart devices were explored. As shown in [3] and other related papers we can state that RESTful web services are an accepted way for designing such devices today. Additionally, the shift from simple raw devices towards more complex scenarios can be clearly identified from the above discussion.

2.3 Business Process Modeling

Conventional enterprise systems support the automation of clearly planned business processes in a constant and limited enterprise environment. The WoT brings the big potential to complement this traditional domain by offering the functionality of connected smart devices based on REST principles in a web-like structure to flexibly perform parts of business processes in a quickly moving and permanent changing environment. Today's Business Process Management (BPM) solutions cover extensive lifecycles to "capture, execute, measure, document, monitor and control

automated and non-automated processes to reach certain goals" [14]. A central requirement before any process automation is the creation of a business process model. Since the beginning of the last decade leading BPM vendors provide solutions that compose business process models out of process tasks. These process tasks are implemented by exchangeable services [15] following a loosely-coupled approach based on the service-oriented architecture methodology. While the WoT envisions the REST service type, the BPM-based enterprise research world [16] still assumes comprehensive solutions using the WS* service type. The latter delivers its functional interfaces and non-functional properties in a description file using a dedicated description language. To exchange messages a protocol such as SOAP is used, which can be transported over HTTP. Just accessing a different service interface and providing a description mechanism for REST service types does not solve the problem of integrating the WoT to current BPM-based ERP systems, it rather starts there. The WoT comes already with new and different components and concepts that the BPM domain on top of the lower layers doesn't handle so far: Where are actually the WoT devices, the physical Things and the RESTful services in the business process model of conventional enterprise applications?
To facilitate the industrial acceptance and utilization, we base our work on the extension of the well-known and current modeling standard BPMN 2.0 [17]. The standard includes a matching graphical and machine readable process model representation. The XML-based machine-readable model serves as a clearly defined interface between process design and deployment phase and states the planned process execution flow for the enterprise systems engine. Depending on the available resources and services the actual resolution and the involved execution components vary time wise, even if the process model remains constant. To support this typical BPM resolution and execution approach, all possible WoT situations must be represented in the process model.

3 A Web of Things for a BPM Web of Things Integration

In this section we will investigate how the power of BPMN can be leveraged to the Web of Things. Having a BPMN modeling environment and an execution engine capable of taking WoT specificities into account is a necessary precondition. If BPMN processes are unable to handle WoT specific requirements, it decreases the interest of the hereby presented integration. In the next subsection we will give a short overview on the current state of WoT integration into the BPMN standard.

3.1 Including the Web of Things in a Business Process Model

To integrate the general components of the WoT to the Business Process Model we start our work based on [18]. Accordingly, we consider four main components where each of them is termed resource in the WoT vocabulary. Fig. 1 illustrates the relation between the components: a business process binding a RESTful API of a WoT service component that exposes functionality hosted on a WoT device attached to a Thing of process interest.

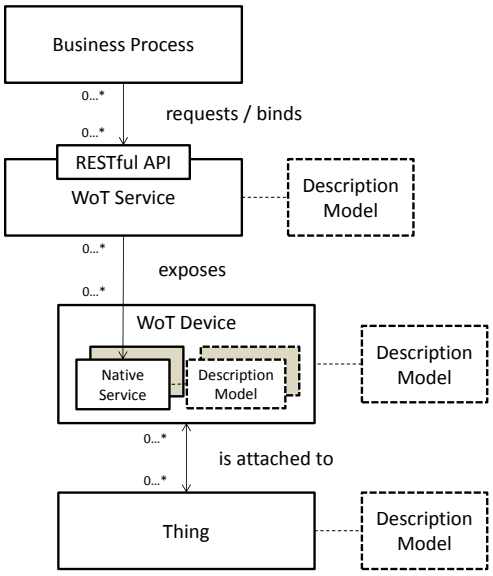

Fig. 1. Business process binding a RESTful API of a WoT service component

The RESTful s**ervice** of the WoT can be combined with the definition of an activity (action, function) as it exists in many process notations. An activity in the process model means a unit of work performed in the process. If this activity is performed by a resource, and the internal subpart is not describable by the process model, it is considered to be atomic or named "task" in [17]. A fully automated software component with a standardized interface as the WoT service could though be represented as a task. To address a separate functionality during the resolution of the process model specific to the WoT service, it is advised in [19] to establish a separate subclass. A **device** of the WoT as a technical artifact can offer computing resources to a process. It acts similarly to a human user as a linking artifact between the process and the real world. Analogously to a human process resource, the WoT device is responsible for the execution of activities and thus it adopts a direct role as a separate process performer in the model. From the perspective of the process model, a WoT device can be understood as a process resource so that contains sub-resources. Those software components with native interfaces are called **native services**. It is hence an indirectly used software component accessed through the well-defined service interface that already presents an atomic unit in the model and does not foresee any underlying unit. From a process model perspective, we understand a native service as a process resource hosted on an WoT device that performs the actual deployment responsibility and according to [20] has the relation „is exposed" to the process task. The characteristic **Thing** as a physical unit of the WoT becomes exclusively part of a business process if it is indirectly of central interest for the process. A process that can detect sensor-based data about a patient can be automated in a WoT only completely, if the patient himself becomes a passive part of the process model. Thus from a process perspective the Thing represents a passive process participant.

Fig. 2 shows the described WoT specific components and their relationships from a process model perspective. Each component can have its own semantic model, as it is provided by [20]. All shown WoT components might be relevant for the actual process resolution so that the appropriate and available services are bound to the process model for being executed by a compliant engine.

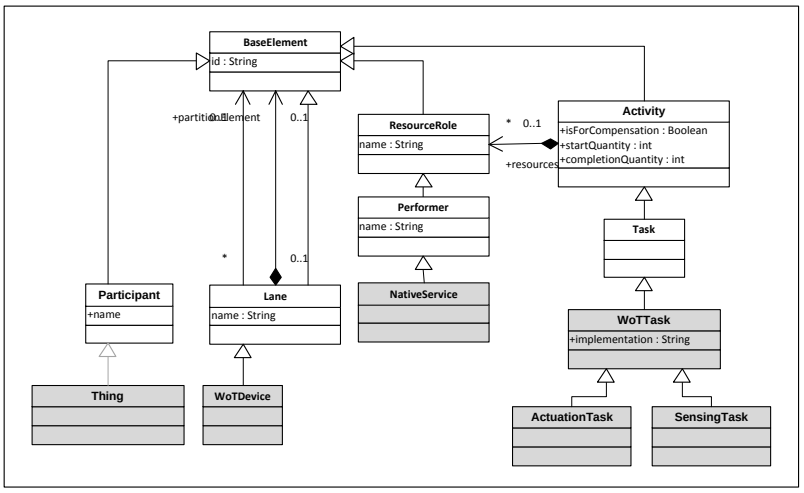

Fig. 2. Simplified BPMN metamodel of all main WoT extensions

The BPMN 2.0 notation basically doesn't support the presented WoT components, but building on [20] we have further created a BPMN-standard extension to integrate the four components WoT service, WoT device, its native services and the attached Things as own concepts. Fig. 2 presents a simplified version of the CMOF class extensions to the related BPMN 2.0 metamodel. The light-shaded areas show the BPMN standard classes, while the gray-shaded areas show the classes of WoT extensions. The class `Thing` was introduced as a subclass to `Participant` represented in the graphical model as a Pool or Collapsed Pool. Thus, the nature of Thing as passive participant can be kept. Device is introduced in the metamodel as a subclass to Lane, the swim lane subdivision of a Pool in the graphical model, which may include potential resources of type `WoTDevice`. As a second process resource type the class `NativeService` is introduced, a subclass of the executable resource role Performer. The illustration of BPMN resource roles in the graphical model depends on the applied modeling tool. They are commonly represented by the assignment as attributes to a process activity without having an own symbolism and through the assignment of an activity to a concrete Lane the structure as sub-resource to the device can be achieved. The class `WoTTask` as a special activity type contains the two subclasses `ActuationTask` and `SensingTask`. Accordingly, a specific RESTful service can be assigned and bound to the `WoTTask` during the process resolution.

This WoT metamodel extension is a first step to present the mentioned components directly in the process model, both in the graphical and machine-readable version.

The metamodel remains invisible for the standard modeling user of the actual BPMN 2.0 editor implementation like [21]. While the CMOF metamodel is not supported by most manufacturers, the standard comes with a matching XSD version of the metamodel. For each process model instantiation a machine-readable XML version is created based on the metamodel stored in the XSD file.

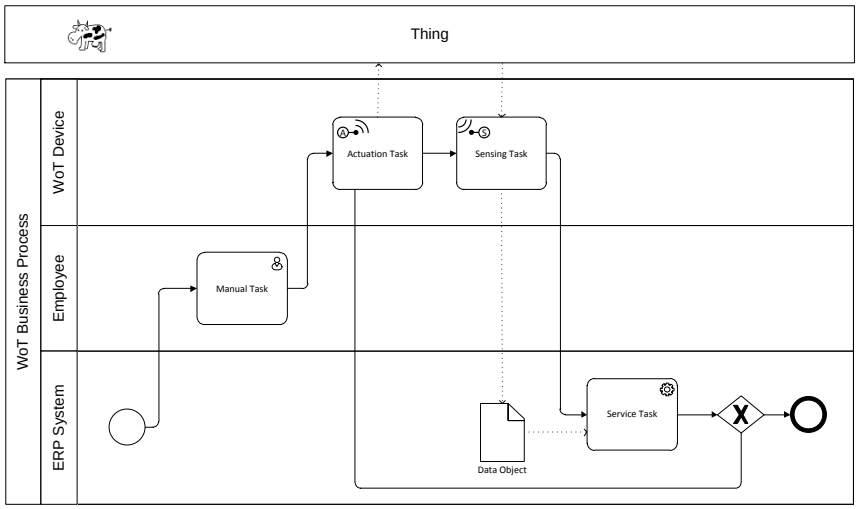

Fig. 3. Graphical BPMN model of business process with WoT extensions

Fig. 3 shows the graphical BPMN model example of an abstract process, which includes the proposed extensions for all WoT components. The shown WoT process comprises the BPMN standard resources “ERP system” and “Employee” and a resource “WoT Device” of type `WoTDevice`. This “WoT Device” hosts several native services. Two of these native services shall be bound to the business process through the `WoTTasks` “Actuation Task” and “Sensing Task” using standard RESTful interfaces. Once the process was started by the “ERP system” automatically, an “Employee” executes his manual task. Following the sequential process flow an allocated “WoT Device” performs an “Actuation Task” on the assigned “Thing”. Another native service of the same “WoT Device” receives the task again over the standardized interface of a RESTful service to measure the just completed sate performed on the Thing via its sensoring capabilities. This functional service stores the measured values in a data object of the “ERP system”. A conventional service component of this closed system checks if the desired change in state of the attached Thing was successful. Dependent on the achievement, the process is either terminated or the WoT tasks are performed again until the change of state is determined as successful.

Depending on the applied BPM methodology different BPM phases are differentiated. We concentrate on process automation by a conventional ERP system, and envision the three main phases: design, resolution and execution (cf. Fig. 4). The central construct of each presented phase either as input or as outcome is the WoT-aware process model. The design phase is supported by a Process Modeling

Environment such as [21] and refers to the creation activity of a graphical and machine-readable process model. Although the created process model may already contain fixed service bindings directly executable, it is hard to know at design time the availability and accessing details of all involved WoT capacities exactly. Therefore, we envision the usage of a supportive and automated resolution phase that finally resolves a professionally specified process model at run time depending on the dynamically available WoT resources. This phase is realized through a Resolution Infrastructure such as [22] which understands the specified components of the professional process model and targets to provide an executable process model by complementing it technically. The actual execution in classical ERP solutions is handled through a central engine such as [23] always having the complete process execution overview. In comparison to the further interfaces presented in Fig. 4 the execution interface to the model is one directional and doesn't change the process model.

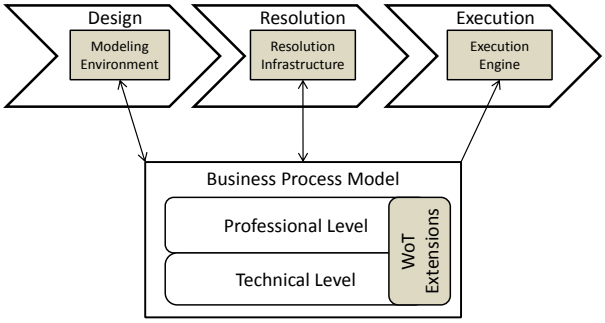

Fig. 4. Phase model for business process automation

3.2 Including Business Process Models in the Web of Things

In this subsection we will discuss our approach to seamlessly embed business processes in the WoT. In the Subsection 3.1 we showed that business process could take care of WoT related requirements. By the proposed extensions it becomes possible to model business process relying on sensors, actuators and using smart objects to achieve their goal. However, until now there was no feedback from the business process to the WoT. While a business process can use smart objects from the WoT, the latter is unable to address business processes. While, it might not be interesting to gain insight into the details of what a business process is actually executing, it would be valuable if WoT application could launch business processes, check their state or abort them. Once can to leverage business processes to the WoT we are done. From the previous subsection we already know how to integrate WoT smart objects into business processes, what is still open is the other way around: opening business process to the WoT.

Since the foundation of the WoT are RESTful architectures, which have been studied for some time now, we state that it is possible to embed existing business process and their execution engine in the WoT. Focusing only on RESTful

architectures, allow us to show up a clean solution embracing all parts of such an interaction. In Chapter 2, we saw that the WoT is mainly about connecting smart objects together to form mashup applications. With the increasing complexity of the desired scenarios, the clean integration of computational services, might it be marketplace-like services or more complex ones, becomes important. Thus, we face two types of interactions when speaking of the WoT. The first ones are classic scenario implying smart objects either delivering sensor readings or offering actuators over a RESTful interface. The second are computational ones.

The latter are not tied to some specific smart object but rather enhance what smart objects deliver and augment their capabilities. An example is a simple GPS sensor. It can tell where on the world it actually sits. However, it is only with a service like Google directions where this particular smart object gets augmented powers. By combining a general routing service (like the one provided by Google) and a GPS sensor offering position readings we can build a smart routing application. Separating the routing from the smart objects make sense. First, several GPS smart objects can rely on the same routing service. This reduces the complexity of every device. Second, business logic is only build once, and has to be maintained only once. Should there be a bug in the routing service, it is sufficient to fix this bug once and all mashup applications combining a smart object together with this service automatically benefit from the bug fix.

RESTful services can be divided into following parts: (1) A clearly defined interface to the outer world. This interface is the composition of the exposed resources together with the uniform interface to interact with them. (2) Clients and servers exchange messages to communicate. These messages are resources representations. Most time they take the form of XML, JSON or HTML files. But, depending on the situation other formats are possible, as long as they respect the IANA list of known mime types[2]. (3) Some sort of business logic. This part is executed when a resource is requested. Business logic range from simple database interaction, over sensors queries, to complicated processes. From a client's perspective, the business logic is a black box. Only the first two parts are important for them; which resources are available and how the interactions with them looks like. Let's take for example Google's routing service, which also proposes a RESTful interface. For a client the interesting parts are the resources themselves and the functionality or information available on these resources. However, where this information comes from, for Google's routing service, some shortest path over a map, is only secondary.

Past research demonstrates that, instead of starting from scratch, it can be easier integrating into the WoT what already exists. The research on the integration of the fosstrack data into the WoT is a good example for this approach [24]. Instead of creating a new fosstrack, the deployed version was enriched with a RESTful façade, seamlessly embedding it into the WoT. The main advantage of this approach is to build upon what is already proven to work. Regarding business process, today's standard is BPMN. It is successfully used all over the world and the applications building on top of these business processes will not disappear in a near future. Furthermore, this approach and the connected tools have shown its robustness.

[2] http://www.iana.org/assignments/media-types/index.html

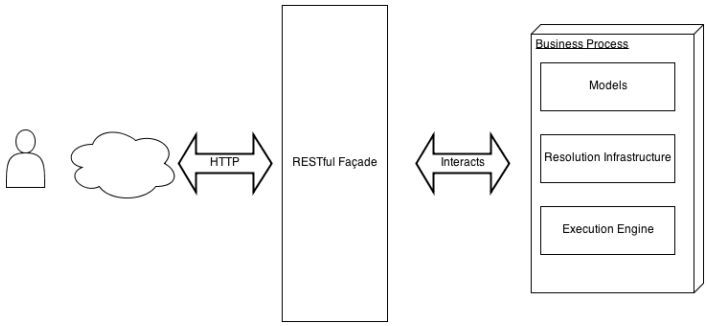

Fig. 5. Participants of the system

Instead of proposing a totally new approach on how business processes should be modeled and executed, we take what already exist and adapt it to the WoT. Complete new approaches on how to design and execute business process for RESTful architecture have already been explored in the past [25]. However, they lack the acceptance of big companies. Such companies have, over the past decade, invested money in adopting SOA architectures and standards, by choosing BPMN approaches. Rather than trying to convince them to start again, our approach allows the coexistence of both approaches.

We can apply this same approach to business processes by building a RESTful façade allowing a seamless interaction with the former. As long as the client knows what resources are available over this interface and how to use them, he will not ask how the execution is actually done. Therefore, the proposed solution is made of several parts shown on Fig. 5. On the leftmost side sits the client or the client application. It uses the RESTful API provided by the RESTful façade sitting between the client and the business processes on the right side. The business processes related parts are all grouped together. The client will never talk or use them directly. Instead, she will use the RESTful façade. From Subsection 3.1, we already know that business processes are composed of: (1) models, (2) some resolution infrastructure and (3) the execution engine. The RESTful façade will interact only with a subset of these parts. A client can retrieve or create models. However, the modeling is out of the scope of the interaction. Additionally a client can select a model and have it executed. The execution may use the resolution infrastructure if necessary and then let the execution engine handle the real execution of the business process.

To make the system work, we need at least the following components: (1) a resource leveraging the business processes to the WoT, and (2) some execution engine. In Subsection 3.1 we demonstrated how smart devices could be integrated into BPMN and we presented the necessary extensions to be able to model such processes. These extensions still have to be ported on the resolution and execution layer. To keep Things simple, we decided to implement the necessary resources, but to use a mock BPMN engine. Replacing the BPMN engine by a mock allows us to focus on the WoT specific aspects. This approach is sufficient to test the viability of the approach and draw some conclusions.

Our system exposes business processes as resources on the WoT. A client can get a list of running business processes or the details of one process. Furthermore, she can

launch new instances of processes or cancel running instances. Fig. 6. shows how this integration looks like for the case of the instantiation of a new business process. Each cloud represents one resource type, some of them map to smart objects, and others do not.

Let us take a hospital as our use-case. The daily business in hospital environments includes fixed sequences for given tasks. As such, they can be modeled in BPMN and executed. With the introduction of connected smart objects however, it is even possible to push further the automation of such business processes. As such, we can imagine the following scenario: "For a given patient, measure each 30 minutes and during 10 minutes the patient's temperature. If it exceeds 39 degrees Celsius alert whoever is responsible and available for this patient. Further, log the results in a new analysis sheet associated to this patient."

Table 1. Overview of the available Resources.

URI	Methods	Meaning
http://.../tasks	• GET • POST	• Returns the list of tasks • Creates a new task.
http:/**Table 2.** /.../tasks/{id}	• GET • PUT • DELETE	• Returns one task. • Modifies one task. • Deletes one task.

At the beginning the client POSTs a BPMN model to the TaskExecutor Service. By issuing a POST request, she creates a new task on the service. This action implies the creation of a new resource on the TaskExecutor Service. Furthermore, the model, encoded in the POST request payload is transmitted to the Business Process Engine where the resolution and execution is done. In our case, a predefined mock process is instantiated. The newly created resource stands for the created instance of the process. Over this resource, a caregiver can ask for the state of process or stop and delete the execution of the latter. This interaction happens over the RESTful interface exposed by the TaskExecutor Service. Table 1. gives an overview of the available resource on the TaskExecutor Service and the associated REST methods. There are mainly two types of resources: (1) the tasks resource, grouping all tasks on the service. Such tasks can either be running, aborted or finished. Through RESTful principles it is possible to filter this list by using URI query paramters. Thus, it is possible to return only running tasks on the system. (2) Individual taks. A task is a one-to-one mapping to a business process. It reflects the state of the associated business process as well as the model, which served to instantiate the process. Depending on the engine it could also reflect outcomes of the process. The proposed façade is restricted to the minimum requirements. This comes from the requirement that we do not make any assumptions on the underlying business process engine used. In a truly RESTful world it might be interesting to further break down a task resource into sub-tasks. However, this break-down puts some strong criteria on the business process engine. On the other hand, the use-cases assume that some standard execution engine (extended with the necessary WoT bindings as explained above) is used. In our case the instantiated mock process, will look up for a thermistor device and the patient to which this thermistor is linked. The results form the measures are then stored in a newly created Analysis associated

to the patient the thermistor belongs to. For the mock-implementation, the process stops itself after a few seconds, proving that it has successfully read from the smart device, and saved the information to a new Analysis associated to the patient. This Analysis and the associated patient are also available over a RESTful API. Thus, the system becomes a whole and is browsable with any modern webbrowser. Coupled to a real execution engine, this process could do more complex scenario. As such, it could examine the thermistor values and raise an alert when a given temperature is exceeded. With the current implementation we have shown that it is possible to integrate BPM Processes seamlessly into the WoT, leveraging the power of BPM to the WoT and vice-versa without changing the principles of one or the other world.

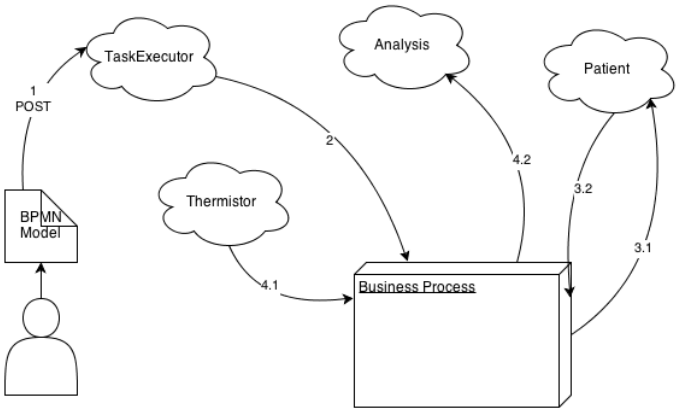

Fig. 6. Creation of a new instance of a given business process

5 Conclusion and Outlook

Integrating the architectures and its related principles of traditional BPM-based ERP systems and the WoT with one another is an important challenge that needs to be overcome to see wider usage of WoT-enabled business processes in the future. Thus, this integration provides many benefits and opens the door for mutual use of both worlds without changing the individual approaches in its roots. With this paper we have introduced our bidirectional integration approach: First, we followed a classical BPM-lifecycle for business process automation and investigated how the CMOF metamodel of the standard BPMN 2.0 could be extended to express WoT components. Therefore we presented the standard-conform extensions as part of the graphical as well as the machine-readable model. We identified three main phases with its responsible software components. Second, we examined how the existing information occurring during the three BPM-phases can be seamlessly integrated as resources into the WoT. In doing so, we followed its REST principles that make use of the information provided by a conventional business system through new standardized interfaces. Finally, for a use case we have shown how this component-

based approach allows creating new applications that combine the physical and the business world.

The provided implementation proves the feasibility of this approach. The integration of a real business process engine with the necessary extensions is currently ongoing work. Next we will replace the mock engine by our real resolution and execution infrastructure. Our future work will deal with the creation of a WoT Reference Architecture. Further, we will deal with the stepwise implementation of the presented BPMN metamodel extensions in a web-based editor tool such as [21].

Acknowledgments. The authors would like to thankfully acknowledge the support for this work provided by Prof. Jacques Pasquier-Rocha. The author Sonja Meyer would like to thank Carsten Magerkurth for his valuable suggestions and discussions. Parts of this research work are supported by the European Commission within the FP7 project IoT-A, contract number: 257521.

References

1. Priyantha, N.B., Kansal, A., Goraczko, M., Zhao, F.: Tiny web services: design and implementation of interoperable and evolvable sensor networks. In: Proceedings of the 6th ACM conference on Embedded network sensor systems. SenSys '08, New York, NY, USA, ACM (2008) 253–266
2. Fielding, R.T.: Architectural styles and the design of network-based software architectures. PhD thesis (2000)
3. Guinard, D., Trifa, V., Wilde, E.: A Resource Oriented Architecture for the Web of Things. In: Proceedings of Internet of Things 2010 International Conference (IoT 2010), Tokyo, Japan (2010)
4. Richardson, L., Ruby, S.: RESTful web services. 1 edn. O'Reilly Media, Inc. (May 2007)
5. Ruppen, A., Pasquier, J., Hürlimann, T.: A RESTful architecture for integrating decomposable delayed services within the web of things. Int. J. Internet Protoc. Technol. 6(4) (June 2011) 247–259
6. Freund, J., Rücker, B., Henninger, T.: Praxishandbuch BPMN. Hanser (2010)
7. Kortuem, G., Kawsar, F., Fitton, D., Sundramoorthy, V.: Smart objects as building blocks for the Internet of things. Internet Computing, IEEE 14(1) (jan.-feb. 2010) 44 –51
8. Hui, J., Culler, D.: Extending IP to Low-Power, Wireless Personal Area Networks. Internet Computing, IEEE 12(4) (july-aug. 2008) 37 –45
9. Ostermaier, B., Kovatsch, M., Santini, S.: Connecting things to the web using programmable low-power WiFi modules. In: Proceedings of the Second International Workshop on Web of Things. WoT '11, New York, NY, USA, ACM (2011) 2:1–2:6
10. Shelby, Z., Hartke, K., Bormann, C., Frank, B.: Constrained Application Protocol (CoAP). Technical Report draft-ietf-core-coap-07.txt, IETF Secretariat, Fremont, CA, USA (July 2011)
11. Gupta, V., Goldman, R., Udupi, P.: A network architecture for the Web of Things. In: Proceedings of the Second International Workshop on Web of Things. WoT '11, New York, NY, USA, ACM (2011) 3:1–3:6
12. Mayer, S., Karam, D.S.: A computational space for the web of things. In: Proceedings of the Third International Workshop on the Web of Things. WOT '12, New York, NY, USA, ACM (2012) 8:1–8:6

13. Pautasso, C.: BPMN for REST. In: 3rd International Workshop on BPMN, Luzern, Switzerland, Springer (November 2011)
14. ABPMP: Business Process Management Common Body of Knowledge - BPM CBOK. Leitfaden für das Prozessmanagement. Verlag Dr. Götz Schmidt (2009)
15. Leymann, F., Roller, D., Schmidt, M.: Web services and business process management. IBM systems Journal 41(2), 198–211 (2002)
16. Van der Aalst, W., ter Hofstede, A., Weske, M.: Business process management: A survey. Business Process Management, 1019-1019 (2003)
17. Business Process Model And Notation (BPMN). OMG Specification. Object Management Group (2011)
18. Magerkurth, C.: Converged architectural reference model for the IoT. EC FP7 IoT-A Deliverable 1.4 (2013)
19. Meyer, S.: Concepts for modeling IoT-aware processes. EC FP7 IoT-A Deliverable 2.2 (2012)
20. De, S., Barnaghi, P., Bauer, M., Meissner, S.: Service modelling for the Internet of Things. In: Computer Science and Information Systems (FedCSIS), Federated Conference on IEEE (2011)
21. Signavio Core Components. Signavio GmbH (2012)
22. De, S.: Concepts and solutions for entity-based discovery of IoT resources and managing their dynamic associations. EC FP7 IoT-A Deliverable 4.3 (2012)
23. Activiti BPM Platform. Activiti (2012)
24. Guinard, D., Mueller, M., Pasquier, J.: Giving RFID a REST: Building a Web- Enabled EPCIS. In: Proceedings of Internet of Things 2010 International Conference (IoT 2010), Tokyo, Japan (2010)
25. Pautasso, C.: RESTful Web service composition with BPEL for REST. Data and Knowledge Engineering 68 (2009) 851–866