Publication IV

Wu, Z., Itälä, T., Tang, T., Zhang, C., Ji, Y., Hämäläinen, M. and Liu, Y.. A Web-based two-layered Integration Framework for Smart Devices. ***EURASIP Journal on Wireless Communications and Networking*****, VOL. 2012, NO. 1, pages 1 – 12, April 2012.**

EURASIP Journal on
Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

A web-based two-layered integration framework for smart devices

Zhenyu Wu[1*], Timo Itälä[2], Tingan Tang[2], Chunhong Zhang[1], Yang Ji[1], Matti Hämäläinen[2] and Yunjie Liu[1,3]

Abstract

The explosions of Internet of Things industry have been bringing more and more smart devices (SDs) into business and people's daily life. This creates new opportunities to build applications that better integrate real-time state of the physical world and requires agility for the software to accommodate customers' requirements. Nevertheless, devices are usually provided by different manufacturers, and applications are independently constructed based on their own infrastructures with little interoperability. Web of Things concept has enabled the interoperability between devices by RESTful web service in a light-weight way; however, it make less efforts to discuss how to integrate devices into complex business environment. Service-Oriented Architecture and Business Process Management approach are becoming applicable to embedded real-world devices and provide flexible service composition. However, it is based on WS-* web service specification which is too heavy and complex for devices and not compatible to RESTful style. In such situation, integrating device into business application with simplicity and providing agility composition of service based on device are significant challenges. We propose a web-based two-layered integration framework that enables SD to integrate with each other via light-weight interface and other back-end applications into agile business process. A real-life use case on elderly care is studied in detail based on the framework.

Keywords: integration, smart device, Web of Things, REST, SOAP, SOA, service composition, business process management.

1. Introduction

In the current Internet of Things (IoT) industry, devices are usually provided by different manufacturers, and applications are independently constructed based on their own infrastructures. It means that these devices and applications are fragmentally distributed. This situation makes the sharing of data and collaboration between companies quite difficult. Even though Web of Things (WoT) [1] has initialized an insight of connecting devices to the Web via Internet protocols–Representative State Transfer (REST) [2] web service that physical objects could be acted as building blocks of Web applications which facilitates developing applications based devices, the building block for a device still remains simple and fragmental with each other, as well as with other back-end applications. It is still a fine-grained block from business application's point of view. The situation means that integration from device level is still not able to effectively bring devices into market for business usage. Moreover, Web 2.0 mashup of these building blocks is still development-specific for some static requirements of simple scenario; however, the speed of business environment changes is rapidly increasing, so these static, unstructured, and unrepeatable mashup applications will not accommodate their customers' demand.

In the context of this background, the research questions come out that: *how to integrate the fragmental devices with each other, as well as with other back-end applications for a more large-grained composition for business usage*, and *how to also guarantee business agility via business process?*

Within the scope of WoT concept, since many devices usually offer rather simple and atomic functionalities (e.g., reading sensor values), modeling them using REST is often straightforward. While REST services are well

* Correspondence: shower0512@bupt.edu.cn
[1]School of Information and Telecommunication Engineering, Beijing University of Posts and Telecommunications, Beijing, China
Full list of author information is available at the end of the article

adapted for rather atomic services, thus covering a fair part of the basic services offered by embedded devices, they have limitations when modeling services which require complex input and/or deliver complex outputs. The Service-Oriented Architecture (SOA) [3] approach promotes the creation of highly accessible, loosely coupled, business-oriented services, as well as end-to-end, fully orchestrated services using such standards as the Business Process Execution Language (BPEL) [4]. As such, an SOA can provide exactly the types of enterprise artifacts that Business Process Management (BPM) [5] is designed to consume and integrate, for the broader purpose of streamlining diverse business processes. In addition to leveraging the SOA, BPM can in turn serve as both a backbone and a platform for SOA constructs, and BPM design methodologies can extremely be useful in helping to define and design target business services and their orchestration. Thus, the integration of devices into the business IT-landscape through SOA is a promising approach to connect physical objects and to make them available to IT-systems. Nevertheless, the SOA/BPM is traditional integration patterns based on WS-* Web Services, meaning that the specifications are on the basis of Simple Object Access Protocol (SOAP) and Web Service Description Language (WSDL) which could not be applied to REST service.

So, in this article, we introduce a Web-based Two-layered Integration Framework (WTIF) for smart devices (SDs) based on WoT and SOA/BPM approaches. The objectives of the framework are:

Objective 1: provide interface standardization for device-layer integration according to WoT solution;

Objective 2: provide orchestration for device resources and other back-end services among different organizations for service-layer integration according to BPM solution;

Objective 3: provide structured, repeatable, and generic business processes for business agility.

The research method is based on a case study that we will study on the elderly care solution of Finland to analyze the detailed requirements for the integrated solution. Then, we will introduce the integration framework for SDs based on the requirements and finally a qualitative approach has been taken to evaluate the solution based on some implementation efforts of the specific use case.

The article makes the following contributions: First, we elaborated and structured a set of requirements and design paradigms for the integration problem based on the specific real-life elderly care case. Second, we are proposing a two-layered integration approach for the SD which realized the required paradigms. Our third contribution is a use case study with service modeling, reference implementation, as well as evaluations. In the use case study, we have modeled a packaged elderly caring service based on WTIF, and we have demonstrated the integration of assisted devices into the packaged service (PS) with other back-end services. We showed how SDs are integrated into business application by the WTIF with lists of benefits.

The rest of the article is organized as follows. Section 2 reviews related study about the integration of device into business applications. Section 3 describes the real-life elderly care case in Finland. Requirements and required paradigms of the integration work are analyzed in Section 4. Section 5 describes our integration approach in detail. In Section 6, we study on the use case and give a qualitative evaluation of our framework. Section 7 concludes the article.

2. State-of-Art

Integration for devices and enterprise services is not a totally new idea. Several efforts have explored the integration of real-world and enterprise services.

2.1. Web of things

In recent years, the so-called WoT attracted much attention and now begins to gain widespread acceptance as an approach to treat real-world objects and devices as emancipated entities participating in networked applications [6,7]. Heterogeneous interconnected objects have acquired the ability to sense their physical status and environment, to function as actuators, and to communicate not only with other objects/nodes, but to reach out to a plethora of other applications via Internet protocols [8]. It proposes to enable the device capability directly as a REST web service by following RESTful architectural style to facilitate the Web2.0 mashup based on SD. Two basic ways of integrations are provided. One is through direct API access from devices, while the other is through indirect API access on Smart Gateways. Initial concepts of WoT architectures already facilitate the integration of WoT systems with the Future Internet and shall form a set of building blocks of the future WoT [9,10]. While the WoT community has clearly identified the need for interoperability with other Internet applications, the domain of enterprise business process modeling so far seems to be more reluctant to address and model aspects of the real world.

SmartBUPT is a WoT platform in BUPT (Beijing University of Posts and Telecommunications) campus by MINELab, it aims to create an open campus innovation platform to facilitate users to create useful and intelligent services in their daily campus lives by following the WoT concept [11,12]. The platform provides heterogeneous sensors and actuators uniform web APIs access through Smart Gateway to reduce the barrier of Web2.0 application development based on atomic device

functionalities. However, it does not aim at building enterprise application so that the device service could not directly be used for business services with complex logics. Without the integration with business process, the device APIs are difficult to be commercially viable.

2.2. SOA-based integration for smart things

Pintus et al. [13] has considered connecting smart things via Web Service orchestration. In similar way, the projects SOCRADES [14] applies SOA approach for embedded network. SOCRADES is a concrete integration architecture focusing on leveraging the benefits of existing technologies and taking them to a next level of integration through the use of Device Profile for Web Services (DPWS) and the SOCRADES middleware.

In SOCRADES framework, it uses DPWS [15] to provide the functionality of embedded devices as service. DPWS is a subset of Web service standards (such as WSDL and SOAP) that allow minimal interaction with Web services running on embedded devices. Moreover, SOCRADES provides service bus and business process middleware upon which sophisticated production processes can be modeled and support connecting SDs, i.e., intelligent production machines from manufacturing shop floors, to high-level back-end systems such as an Enterprise Resource Planning (ERP) system.

Embedding SOA concepts at device level initially seems a good idea. However, we have to keep in mind that SOA standards were designed primarily for connecting, complex, and rather static enterprise services. Thus, implementing WS-* standards for enabling device functionalities is not always straightforward with redundant messages. Unlike enterprise services, real-world services are deployed on resource-constrained devices (even through an indirect Smart Gateway which is commonly also embedded environment), e.g., with limited computing, communication, and storage capabilities. This requires significantly simple, more light-weight protocols and uniform interfaces.

2.3. RESTful BPM for integrating REST web service

With the goal of attracting a larger user community, more and more service providers are switching to REST in order to make it easy for clients to consume their Web service APIs. Thus, Pautasso has proposed the BPM for REST [16] which has extended the WS-BPEL language to support integration of RESTful web service into a BPM system and also enable the process as a RESTful service. It possibly facilitates integrating Web2.0 applications with enterprise applications.

The new extension to the BPEL standard provides native support for the composition of RESTful web services. This approach turns the notion of "resource" and hides the resource-oriented interaction primitives (GET, POST, PUT, and DELETE) of REST inside the service-oriented abstractions provided by WSDL. With these, a BPEL process can directly interact and manipulate the state of external resources and declaratively publish parts of its state through a RESTful web service API. However, this approach focuses mainly on integrating REST service and managing the business process in a REST style but ignores the BPM with both WS-* service and REST service.

3. Use case

In Finland, elderly people prefer living in their homes instead of hospitals due to the increasing cost. So, some assisted technologies are being utilized into the elderly home care, such as wireless health monitoring systems, mobile robotic assistants, social alarms, and fall detection. New assistive devices should be introduced based on the analysis of each person's needs and capabilities of using the devices. In many cases, multiple devices are used at the same time. In addition, the devices are connected to different information systems as part of health care processes, and manufacturers have noticed that offering only assistive devices is insufficient–a service package that includes types of assisted devices and attached services is more lucrative alternative.

Unfortunately, the devices and services offered by the different providers are often incompatible with each other. Usually, each device has its own online service. Thus, purchasing using and managing the devices and services separately are difficult. Moreover, the core applications from most elderly care organizations and healthcare service providers are Patient Administration Systems (PAS) and Personal Health Record systems (PHR). So, a requirement for the manufacturers is that they should not only integrate their products with each other, but also with these legacy systems for large-grained business applications. However, today most of these manufacturers system are not well integrated into the PAS or PHR, resulting in a need to re-enter the data.

Another challenge in this case is that the caring plan ordered by customers may change during the execution period, which means that the requirements of the PSs may vary according to the customers' needs. So, any static and inflexible caring service will not follow the rapid change of requirements with less effort.

As a consequence, some integration is needed from different levels in this elderly care case to guarantee that the SDs and services could be integrated into one PSs with enough flexibility and agility.

As Table 1 shows, some small- and medium-sized enterprise (SME) companies provide assisted devices as well as a service provider (SP) who provides PHR services and a healthcare organization who possesses PAS.

Table 1 Products and services provided by different partners

SMEs & SP & organization	Products(P) & services(S)
Addoz	P: Med-O-Wheel devices S: Med-O-Wheel reminds the patient to take the medication on time and regularly http://www.addoz.com
Zephyr	P: Zephyr wearable belt S: Zephyr measures heart rate http://www.zephyr-technology.com
Elsi	P: Elsi safety floor S: Elsi increases safety of the elderly by reliably recognizing if the fall occurs and generates alerts http://www.elsitechnologies.com
Everon	P: Everon Safety bracelet S: Everon tracks the location of elderly and delivers automatic alarms http://www.everon.fi
Playground	S: PHR service to collect and store personal activity and health records http://www.anyplayground.com
Healthcare Center	S: PAS service to manage patient profile

4. Requirements and required paradigm

In this section, we present the requirements and design paradigms of the integration framework for SDs based on the real-life case in Finland. To better understand the requirements, we will give definitions to some important objects at first as shown in Table 2, and then we will give out the specific requirements from different role's points of view, as well as the design paradigms.

From Customer's point of view

The integration should guarantee that the solution is a PS with different types of composition choices and serves both elderly patients and caregivers. It means that the customer could choose different products to create, extend, and modify their service package dynamically by their own needs.

From Caring Service Provider's (CSP) point of view

CSP should be able to compose reusable services with transparency of the detailed specifications of each Product Vendor (PV) and Third-party Service Provider (TSP) via Integration Platform (IP) to deliver their own PS to the customers. In addition, CSP could support changing the business process on the fly with a real-time reaction to the unforeseen changes of customer's requirements via IP.

From Integrator's point of view

Integrator should provide IP where CSP could compose their own PSs; Integrator should guarantee that IP could provide BPM and execution environment for integrating external web services; Integrator should guarantee that IP could integrate different types of external web service regardless of the style and data formats (SOAP-based or REST, XML or JSON); Integrator should guarantee that composite business process could be reusable.

From PV's point of view

PVs should provide a standardized light-weight and easy-to-developed Web interface for their smart device service (SDS) to guarantee the interoperability between different devices, as well as with other back-end

Table 2 Definition of objects

Object	Definition
Customer	Elderly patients who need assistance from caregivers via packaged caring services Caregivers who are in charge of the elderly patients via the packaged caring services, such as nurse or relatives
CSP	The role that delivers packaged caring services to the customers
Integrator	The role that provides IP and service to compose other organizational applications
PV	The role that provides assisted caring devices
TSP	The role that provides legacy back-end system which could be integrated by CSP, such as PAS and PHR
PS	A composite service that integrates device products of each manufacturer and other back-end services
IP	An entity that provides service composition middleware
LA	The back-end system which is provided by TSP, such as the PAS and PHR. These systems are usually enabled via ESB or RESTful web service by default in the case
SD	An entity that is used for sensing, monitoring and assisting elderly patients

applications. Additionally, the Web interface should also be adapted to IP.

From TSP's point of view

The TSP usually enables Legacy Application (LA) via Enterprise Service Bus (ESB) or RESTful web service. Thus, the interface of TSP should be adapted to IP.

In general, the design paradigms for the SD integration framework could be summarized in different facets:
-*Interoperability*: at the device level, the SD should share a uniform interface pattern in regardless of the heterogeneity of underlying specifications, which means a high interoperability of different SDs.

-*Complexity*: the interface for SD should be simple and light-weight to fit into the constrained environment of embedded system. It means a low complexity of device interface.

-*Flexibility*: the granularity of service could be either coarse grained or fine grained according to particular requirements. It means a high flexibility of choices for integration blocks.

-*Compatibility*: a process could integrate both SOAP-based and RESTful web services with different data formats. It means high compatibility between different interface patterns.

-*Agility*: common business process could be repeatable to other processes and services. It means a good agility to adapt to the variation of business process with less modification efforts.

5. Web-based two-layerd integration framework

In this section, we present a concrete two-layered integration architecture focusing on leveraging the benefits of existing Web technologies and taking them to a next level of integration. The concept of "two-layered" in this article means the *Device Layer* and *Process Layer*, as shown in Figure 1.

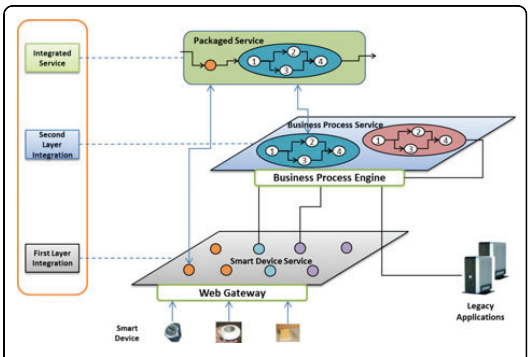

Figure 1 Integrating lagacy device into business via two layer approach: device and process layers.

At the *Device Layer*, the Web gateway is the core that it will bridge different types of sensors, actuators, and other SDs into the Internet, and all the data and capabilities of the devices could be abstracted as Web resources and enabled as web service APIs. We assume that the devices are provided and shared by third-party vendors. It means that their products are legacy infrastructures with different proprietary specifications and they could not directly be connected to the Web; thus these devices should connect to the Web via Web gateway. The first layer integration focuses mainly on the boundary between physical and digital realm, and the *OUTPUT* of it is the fine-grained building block for Web application based on SD which could be called SDS. At this layer, the SDs will have a uniform Web interface that could diminish the underlying heterogeneity of devices.

At the *Process Layer*, the Business Process middleware is the core that it will manage the business process modeling and execution, so it will have physical information integrated into some business process among companies with other back-end services by the business process engine. The second layer integration focuses mainly on the boundary between fragile digital resource and business services, and the *OUTPUT* is the large-grained web-based building blocks with more complex business process which could be called business process service (BPS). At this layer, the SDs will be integrated with each other and other legacy business applications into process and the integrated process will be reused as a new service to other processes. So, BPS blocks are built to handle common complex task for business usage and process could be added, replaced, or removed dynamically as modular blocks.

5.1. Device layer integration: RESTful WoT gateway

The RESTful WoT Gateway bridges the non-Internet access physical devices to the Internet and abstracts the data and capability of these devices into programmable Web Service API. So, the gateway could directly provide atomic web services to other service consumers with transparency of device's underlying technologies.

In the device layer integration, RESTful architectural-style should be followed because of its light-weight, loosely coupled, and standardized features. Since many such devices usually offer rather simple and atomic functionalities (e.g., reading sensor values), modeling them using REST is often straightforward. The RESTful Web Service have some basic regulations: (1) URI as the identification of the resource; (2) HTTP as the application protocol and HTTP method (GET, PUT, POST, and DELETE) as uniform interface with common semantics for service invocation; (3) multiple representations (data formats and media types) for the resources;

(4) hypermedia as the engine of application state that resource relationships can explicitly be rendered as hyperlinks (i.e., pointers to URIs). Therefore, we argue that a light-weight web server for the HTTP connectivity is necessary as the communication pattern of RESTful interaction. Besides, there exist different kinds of devices provided by different manufacturers and the standards for the data formats are diverse. Hence, there should be a uniform data model for information exchange at the device level, and the interface of the WoT Gateway should carefully be designed by following the data model.

As Figure 2 shows, the internal components of gateway contain (1) plugin drivers that communicate with devices in heterogeneous protocols; (2) embedded database that caches data from the devices; (3) embedded web server which provides HTTP connection and service container; (4) RESTful middleware that provide RESTful web services and manipulate REST interactions. The RESTful web services could directly be invoked both by end-user for Web2.0 mashup and service composer for business process.

5.2. Process layer integration: service composition and BPM middleware

Business process is described and modeled in BPEL and stored as a XML documents. A BPEL process describes a flow of interactions between the process and services. Each interaction describes what role the process and services play at that step in the flow and what data can be manipulated by the parties in those roles. The BPEL engine could generate instance of the process according to a certain BPEL document and execute the workflows when it is initialized.

At this layer, two basic tasks should be handled via BPEL engine: (1) binding and invoking RESTful web service, along with SOAP-based web service; (2) enabling the common business process as a service.

5.2.1. Web service adaptor

Traditional BPMs currently are all based on the specification of WS-*, such as WS-BPEL and WSDL 1.1 [4]; while device's capabilities are exposed in RESTful web service at the first-layer integration, and a RESTful web service could not be well described by WSDL 1.1 (though RESTful web service could be described by WSDL 2.0 [17], the WS-BPEL could not be compatible with WSDL 2.0). Therefore, current BPEL engine could not bind and invoke a RESTful web service unless an extension to map RESTful web service to WS-* web service via an adaptor.

Web Service Adaptor (WS-Adaptor) is the HTTP interface of business process engine, and the platform could consume RESTful web services with an internal SOAP-based invoking via the adaptor. The primary functions of the WS-Adaptor are (1) Invoked by business process engine in SOAP message to have WSDL 1.1 bindings; (2) Translate HTTP message into SOAP message and vice versa by HTTP connector; (3) Representation transfer from other formats (JSON, ATOM) into Plain Old XML (POX).

For the operations of RESTful, web services are based on four HTTP methods; so, a group of basic SOAP operation handlers are provided: *onGET()*, *onPOST()*, *onPUT()*, and *onDELETE()* to map the SOAP message to HTTP message as shown in Figure 3. For each kind of SOAP operations, there is a WSDL template to describe the SOAP handlers. The *Message* is service-specific which means the *URI* specifies the address of the REST interface, the *MediaType* specifies the data formats of resource representations, and *Param* represents the parameters for a request. When the RESTful web service is published, a WSDL instance for the interface is generated based on the template with the URI, MediaType, and necessary request parameters of a RESTful

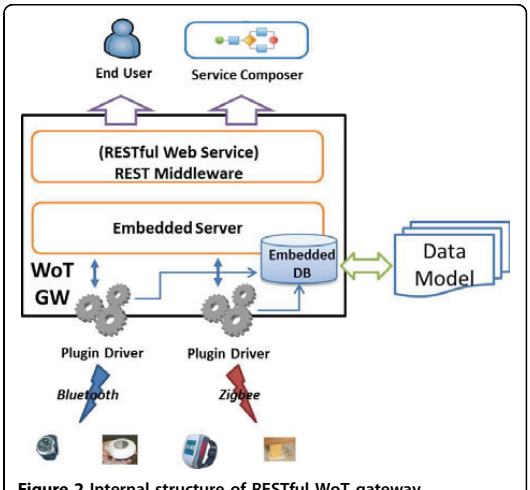

Figure 2 Internal structure of RESTful WoT gateway.

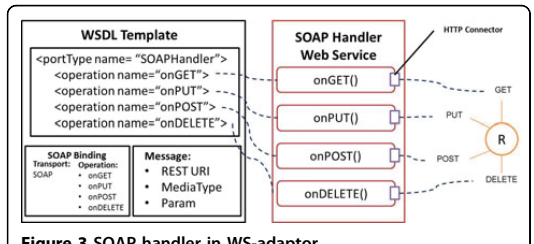

Figure 3 SOAP handler in WS-adaptor.

interface. The handlers and WSDLs will be used during the binding and invoking of business process execution.

5.2.2. Service binding and invoking for device service

Within the framework's scope, the devices are exposed as RESTful web service via gateways. To guarantee that a REST web service could also be binded and invoked in the business process engine, an HTTP binding is processed by the WS-Adaptor and BPEL engine. Figure 4 illustrates the procedure of a service binding and invoking.

If the BPEL engine prepares to invoke a RESTful web service inside its workflow, it will bind to the specified WSDL file to check which SOAP web service handler to invoke with URI, representation, and other attribute parameters. Then, the engine will invoke the handler in SOAP message, and the handler will invoke the RESTful web service on the gateway via HTTP connector by binding the request to an HTTP request with URI, method, content-type, and the parameters. The gateway will response the adaptor in HTTP with representations, and the adaptor will transfer the data into Plain XML and response to the BPEL engine with SOAP message.

5.2.3. Creating a BPEL process as a service for another BPEL process

To make a process reusable to another process, the reused process should define its interface and binding relationship; so that the main process could invoke it as a web service. We assume that a process could be abstracted as either RESTful web service or SOAP-based web service. If the process is abstracted as a RESTful interface, it will be described by the WSDL template which is defined in the previous section. The manipulation of the process will be based on four kinds of operations (GET/PUT/POST/DELETE).

BPEL defines constructs to identify roles and relationships used in interactions. The constructs are *partner link* and *partner link type*. A *partner link* describes the *roles* that a process and service play as well as what data they can manipulate in that *role*. A *partner link* is defined by its *partner link type* which describes the kind of message exchange that two WSDL services intend to carry out. A *partner link type* characterizes this exchange by defining the roles played by each service and by specifying the *port type* provided by the service to receive messages appropriate to the exchange.

As Figure 5 shown, when you define a sub-process as RESTful web service, you can select it as a static endpoint reference for a *partner role* of the main process. In the WSDL descriptor file of the sub-process service, the *partner link type* could be defined as a binding relationship between sub-process and main process. A *partner link type* can include one *role* or two *roles*. In the case where a *partner link type* contains only one *role*, there is no restriction placed on the calling web service regarding *roles*. The *partner link* in main process BPEL and sub-process BPEL could both use the *partner link type* with its *role* defined in the sub-process descriptor. However, the sub-process will use *myRole* in its *partner link* definition to specify itself to provide service to other process; while the main process will use *partnerRole* in its *partner link* definition to specify the sub-process as the external service it will invoke during process execution. Therefore, when the main BPEL process is running, it invokes the sub-process according to the predefined *partner link, portType*, and *operation* parameters. For a RESTful sub-process invocation, the main process will specify *portType* as "*SOAPHandler*" and *operation* as "*onPUT()*" which are predefined in the WS-Adaptor.

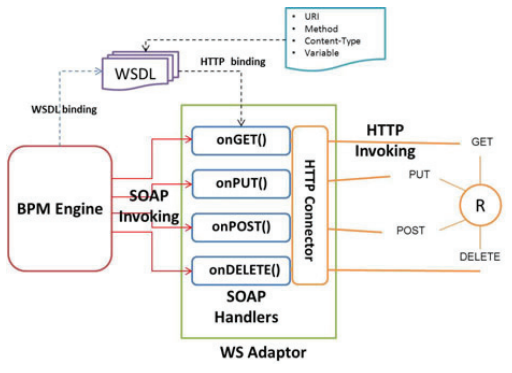

Figure 4 RESTful device service binding and invoking.

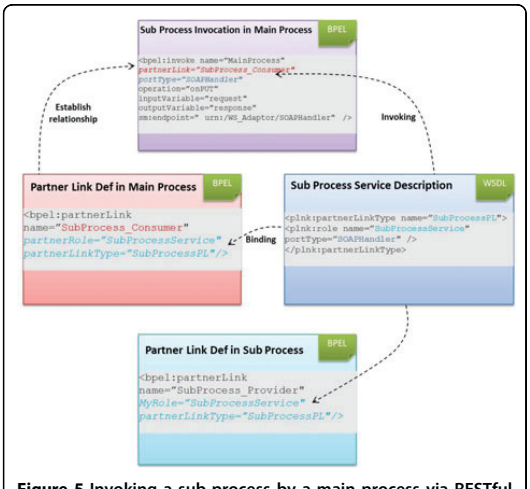

Figure 5 Invoking a sub-process by a main process via RESTful web service.

6. Use case study

To prove the feasibility and reasonability of the framework, we study on the use case described in Section 3. Most efforts of the case are based on the Active Life Home (ALH) project [18,19], SIDE project [20], and Valpas system [21].

6.1. Service and process modeling

In our elderly care case, the old person usually suffers from chronicle heart disease that the heart condition should be measured and reported. Additionally, they should also be reminded of taking medicines on time. Therefore, we model a packaged caring service based on the requirements.

The service in this framework can be a process, sub-process, or single function. The aim was to model and decompose the processes and services down until such a level of granularity that the services can be implemented as web services. Using this top-down approach, the process depicted in Figure 6 was modeled.

In the model, the Elderly Caring process is the most central and highest level process. The customer of the process is the elderly patient. The owner of the process is the caregivers in charge of the unit. The instance of the process is triggered by an event which is the arrival of an elderly patient for elderly care. In the main process, three main tasks are executed in a sequence, including caring plan, execution, and assessment. Since the caring execution is a common task for most caring scenario including several sub-tasks, such as initializing caring plan, monitoring patients, recording physical measurements, and sending events to the caregivers, we model it as a sub-process for the main process to invoke.

The Caring Execution sub-process is the second-layer service which consists of process steps executed according to a predefined logic and which may invoke other coarse-grained services. In the process sequence, it will initialize the caring plan by reading patient profile from PAS service, monitor and report patient heart rate measurements, and remind patient to take pills. Because the monitoring, reporting, and reminding tasks are also common activities for other caring packages, we compose them together as another common sub-process named Monitoring sub-process. The Monitoring sub-process also belongs to the second-layer service. It is a repeating activity which is invoked both by Monitor Heart Rate and Medicine Remind tasks in the Caring Execution sub-process.

The first-layer service in the model is the Device service which is provided by the SDs in a RESTful web service via WoT gateway. Two kinds of Device services are invoked by the Monitoring sub-process. One is from Zephyr which is invoked by Monitor Heart Rate task to measure patient' heart rate; the other is from Addoz Dispenser which is invoked by Medicine Remind task to remind patient of taking medicine. Additionally, in the Monitoring sub-process, the raw data from devices will sync to the playground's server by invoking PHR service, and sending back notification messages to both caregivers via the Monitoring sub-process an elderly person via Reminding Service directly on the Addoz device.

6.2. Reference implementation

We give a reference implementation based on the existing work and system as shown in Figure 7. It shows how the WTIF could practically be constructed.

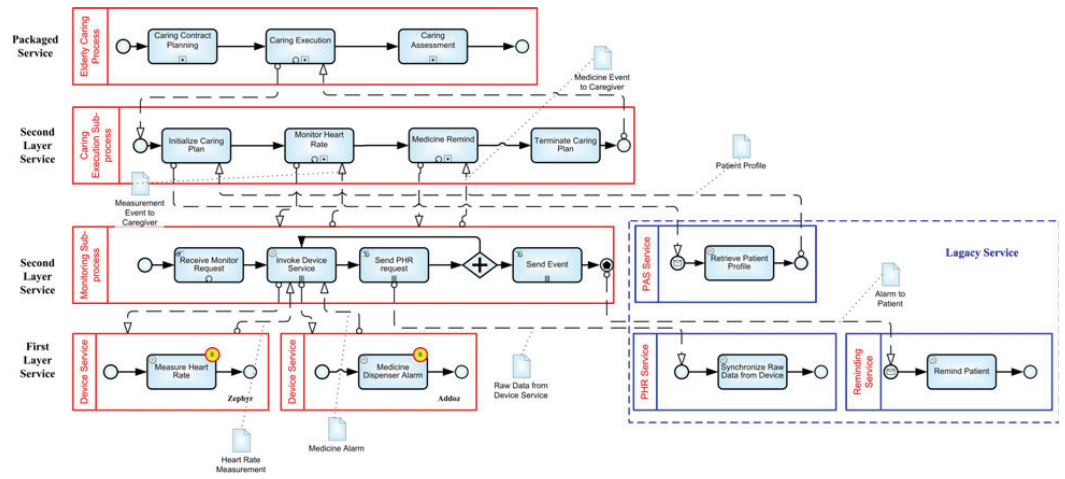

Figure 6 Elderly caring service and process modeling in a layered model.

Figure 7 Referencen implementation of the elderly caring solution based on WTIF.

6.2.1. VALPAS Home WoT gateway

In our case, the Valpas Home WoT gateway is used to connect Addoz medicine dispenser and provide REST APIs for the connected device. The Valpas Home WoT gateway system is developed based on the Valpas system [21] designed by Automation Department of Aalto University to be an easy to use home integration, safety monitoring, and alarming platform.

The Valpas system (shown in Figure 8) runs on OpenWRT-based Linux gateway called ThereGate [22]. The device has WiFi, four USB ports, and integrated Z-Wave controller for wireless sensors. USB-ports are used for connecting to ZigBee devices like, e.g., Addoz medicine dispenser. Internet connection is available either using the built-in Ethernet port or optional 3G modem.

In the earlier work, we have developed Open Building Information Exchange (oBIX) bridge for the ThereGate.

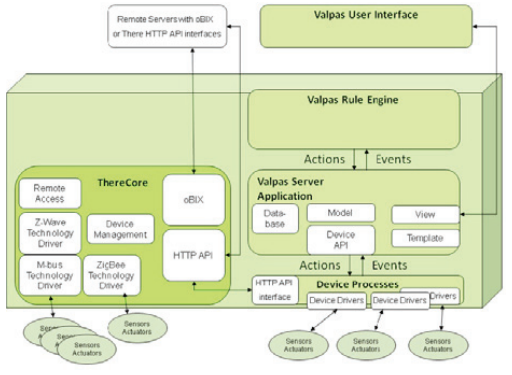

Figure 8 Valpas WoT gateway system consists of three main processes: ThereCore, Valpas Server, and Valpas Rule Engine in addition to device processes.

The oBIX bridge provides a REST interface for the ThereGate. Thus, the alarm notification messages from Addoz device could be sent from the gateway via RESTful interface and the gateway can easily be integrated to other information systems.

6.2.2. Android Mobile WoT gateway

Android Mobile WoT Gateway (shown in Figure 9) is used to connect Zephyr wearable belt to measure and record heart rate. The Android Mobile WoT Gateway is developed based on SIDE tool [20] which is an open source tool using android phone to collect data from various medical and well-being devices by connecting them via Bluetooth Health Device Profile (HDP). We have implemented a plug-in application to parse Bluetooth HDP protocols and store the data into the SQLite database [23].

To provide mobile phone a web-based environment and capability to handle HTTP messages, a simple web server, which will mainly handle HTTP request from client and sends response back, should be deployed on it. We argue to choose the iJetty [24] which is an open source web server on android phone as the web-based environment in our case. Additionally, we argue to use Restlet framework for Android [25] as the REST middleware because it provides HTTP connectors, URI routers and XML/JSON/ATOM formats parser, and it could also be deployed in iJetty servlet container. As a consequence, the heart rate measurements from Zephyr device could be exposed as REST interface.

6.2.3. Process middleware

We have investigated several optional BPM engines, such as Apache ODE [26], Active BPEL [27], and JOpera [28]. These engines are developed based on the WS-BPEL.

In this article, we focus on the new adapters for invoking external RESTful services from the WoT gateway using the HTTP protocol, as well as on "glue"

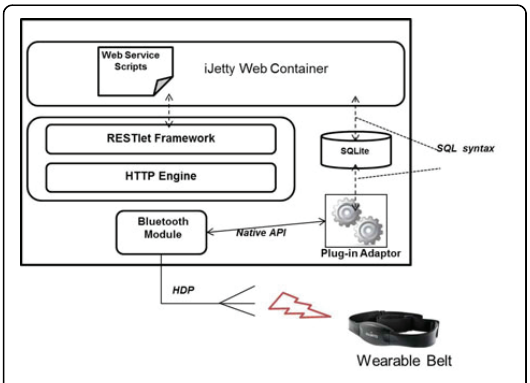

Figure 9 Android WoT gateway based on SIDE tools.

adapters to perform local computations used mainly for data transformation. A large collection of adapters (including support for traditional WS-* services) is available and more can easily be provided with a plug-in-based extensibility mechanism that does not affect the basic composition language.

The JOpera for Eclipse rapid composition environment provides an integrated development tool supporting the entire lifecycle of a service composition. It features a design perspective, with tools for managing a library of reusable services, a visual, drag&drop and connect, environment for composing them into workflows. You can add your own service invocation adapters and package them as Eclipse plugins, so JOpera helps you to deal with invoking SOAP and RESTful Web services, etc. Workflows are compiled to Java bytecode for efficient execution. Once workflows are completed, they can be deployed on a remote execution engine to be published as a reusable service (both accessible using REST and WS-* interfaces).

Thus, we argue to use this JOpera to compose Device services with the PHR service and Reminding service into a Monitoring sub-process, and we could publish this Monitoring sub-process as a RESTful web service, then compose it with PAS service into a Caring Execution process as the model in Section 6.1.

6.3. Results and evaluation

6.3.1. Answers to research questions

• *How to integrate the fragmental devices with each other, as well as with other back-end applications into a more large-grained composition for business usage?*

In the real-life elderly care use case of Finland, the heart rate monitor and medicine dispenser are integrated into the Web via Home and Mobile WoT Gateway at the device layer integration. During this integration, the devices are all abstracted as uniform REST API. The programming of application based on SDs is not platform dependent any more, instead of the same RESTful interface pattern.

Then, we use a BPEL-based process engine–JOpera with REST extension to orchestrate the Device service, PHR service, and PAS service into a Monitoring process which handles a more complex service. This integration guarantees the compatibility of composing both RESTful service and SOAP-based service into one business process

• *How to guarantee business agility via business process?*

The key characteristic of agile business is repeatability of process instance which could provide support to changed requirements faster and with less effort. According to this paradigm, all the Device and Process services are reusable as a new service to other process in our case.

Since elderly caring services have something highly related to the monitoring the elderly persons, in our case, we have modeled a common business process for different monitoring tasks, not only for heart rate and medicine alarm, but also could apply to detecting falling events and tracking location via the assisted devices in Table 1. When the elderly patients would like to extend the caring plan with falling detection and location tracking into the service package, the caring service provider (CSP) should only reuse the monitoring process by invoking falling detection and location tracking tasks, but not to redesign and redevelop new tasks for detecting falling events and tracking locations. Moreover, the modification of Monitoring process will not affect the logic of the Caring Execution process who invokes it.

6.3.2. Comparisons

According to the design paradigms for SD integration in different facets, we will evaluate our efforts by comparing our framework with other related study stated in Section 2, and the results are shown in Table 3.

Interoperability WTIF provides a RESTful web interface for SD at the first layer integration. While WoT solution also proposes to use REST to build a "universal" API for embedded devices. Thus, both two approaches provide uniform communication channel (HTTP protocol), operation interface (HTTP method), and identification (URI) for manipulation on the resource of devices. SOCRADES leverages DPWS to enable device connecting to high-level middleware with WS-* standardization. It also shields the underlying heterogeneity of devices. RESTful BPM solution is not specifically designed for integrating device, but integrating general RESTful web service into business process.

Complexity WTIF, WoT solution, and RESTful BPM solution are all using REST to abstract service. While REST services are well adapted for rather atomic services, thus cover a fair part of the basic services offered by embedded devices. Thus, for light-weight and more end-user-oriented applications, the RESTful approach offers significant advantages such as simplicity, direct Web integration, and loose-coupling. The SOCRADE is

Table 3 Camparions between WTIF, WoT, SOCRADES and RESTful BPM in different facets

Integration facets	WTIF	WoT solution	SOCRADES	RESTful BPM
Interoperability	√	√	√	-
Complexity	Light	Light	Heavy	Light
Flexibility	√	✗	√	√
Compatibility	√	✗	✗	✗
Agility	√	✗	√	√

based on WS-* service. According to our own experience and of others [26], in traditional integration patterns based on WS-* Web Services, we suggest that WS-* services are to be preferred for highly complex real-world integration and rather static use-cases, such as those involving complex business processes or those requiring high reliability or security.

Flexibility WTIF designs for an integration approach not only at the device level, but also at the process level. The approach guarantees that the services could be based on the atomic functionality of device or involve with complex business process. SOCRADES is aimed at connecting device to the legacy enterprise applications, such as ERP by SOA approach. It also provides composition of atomic device service into a complex business process. RESTful BPM provides mechanism of abstracting a business process as a RESTful web service. It means both a fine-grained function and a coarse-grained composed business process could be a RESTful web service reused by others. While WoT solution only proposes to abstract device function as an atomic service without further composition.

Compatibility WTIF is designed based on a REST adaptor with the extension of traditional BPEL engine. It guarantees that it could compose both RESTful web service and WS-* web service into one business process, thus the devices could be integrated with other enterprise applications. SOCRADES mainly focuses on WS-* web service composition; while RESTful BPM solution mainly focuses on RESTful web service composition. WoT solution does not talk too much about composition of services based on device in its concept.

Agility WTIF, SORCRADES, and RESTful BPM are built based on BPMS. It provides modeling repeatable process service to other process, and some common process with complex logics could be abstracted. Thus, it is not necessary to build all the service from device level to the higher level once the requirements change. WoT solution is more suitable for a static and simple scenario. Building highly complex real-world integration based on WoT solution needs reorganizing and reconfiguring the atomic device service case-by-case. It is not agile enough for the shifts of requirements.

6.3.3. Key findings

On the basis of the use case study, we have found several benefits of WTIF within and beyond technical merits that:

- WTIF is a truly flexible integration framework for SD to build intelligent enterprise applications but not only for end-user oriented mashup applications. It provides uniform light-weight web interface for device, supports for service composition of device services (REST) and other web services (WS-*), and makes the common process as reusable service to device and business applications.
- WTIF is an agile integration framework for SDs which could adjust the quick variation of the business requirements. By modeling repeatable generic process, the service providers could reuse the common functions to implement their own services based on devices with little efforts.
- Based on WTIF, the decision could be made loosely coupled with the running process at the process level for automatic processes and intelligent services. Based on a BPMS, process could execute automatically according to the predefined rule which facilitates the intelligence of service.
- WTIF saves the integration and maintenance cost for IoT SMEs. Development of intelligent applications is eased by web service technology, and the service composition middleware facilitates the mashup of physical resources and other back-end services. IoT manufacturer and IoT service providers need only focus on the business process modeling.
- WTIF provides new possible business model. In this framework, there are several customer segments, such as device manufacturers, IP provider, service providers, developers, process modeler, and end users, who will organize an IoT industry ecosystem. And the value proposition could be based on the exact services delivered to the end users, the technical provider devices and software, reused services, and the service IPs.

7. Conclusion and future study

In this article, we have introduced the WTIF, a Web-based Two-layered approach for integrating SDs. The framework is targeted for integrating heterogeneous SDs with each other and other back-end applications for agile business application. Our integration strategy is to use RESTful web services as the main connector technology for device. Based on it, we use a compatible BPM middleware to compose the RESTful device service with other WS-* back-end service into process.

With a real-life use case study on elderly care in Finland, we illustrated how to use the proposed approach to solve the issue of the practical case. Compared to the related study, the main advantage of the proposed approach is that it unifies the application interface of legacy devices in a simple way, and makes the device integrated into reusable enterprise process service for business usage via compatible and agile business process middleware.

Our next steps include integrating the reference implementation in a bigger setup and testing it with more production systems and quantitative evaluations. And from research perspective, since the RESTful web

service does not support asynchronous communication and devices are dynamic and sometimes transient during the service composition, we will focus on how to provide asynchronous message and real-time notification for event-driven process based on devices, and see how life-cycle of dynamic device service could be managed.

Acknowledgements
This study was supported by the Active Aging project and China Finland Cooperation Project on the Development and Demonstration of Intelligent Design Platform Driven by Living Lab Methodology. Moreover, I will especially appreciate for Panu Harmo's material about Valpas system and Maksim Golivkin's contribution on SIDE project.

Author details
[1]School of Information and Telecommunication Engineering, Beijing University of Posts and Telecommunications, Beijing, China [2]Aalto University School of Science, Espoo, Finland [3]China Unicom, Beijing, China

Competing interests
The authors declare that they have no competing interests.

Received: 30 November 2011 Accepted: 23 April 2012
Published: 23 April 2012

References

1. D Guinard, V Trifa, F Mattern, E Wilde, in *From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices*, Springer, Berlin Heidelberg, 2010). ch. 5
2. RT Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, (University of California, Irvine, 2000)
3. E Thomas, *Service-Oriented Architecture: Concepts, Technology, and Design*, (Prentice Hall PTR, Upper Saddle River, NJ, 2005)
4. OASIS, Web Services Business Process Execution Language (WSBPEL) 2.0. (2006)
5. M Weske, *Business Process Management: Concepts, Languages, Architectures*, (Springer-Verlag, Berlin, 2007)
6. D Guinard, V Trifa, E Wilde, A resource oriented architecture for the web of things, in *Proc of IEEE International Conference on the Internet of Things*, Tokyo, Japan, pp. 1–8 (November 2010)
7. D Guinard, V Trifa, Towards the web of things: web mashups for embedded devices. in *2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009)* Madrid, Spain (April 2009)
8. SENSEI (Integrating the Physical with the Digital World of the Network of the Future) http://www.sensei-project.eu/ (2010)
9. IoT-A (Internet of Things - Architecture) http://www.iot-a.eu/public (2011)
10. G Schmidt, Business Process Management Common Body of Knowledge–BPM CBOK. Leitfaden für das Prozessmanagement herausgegeben von der European Association of Business Process Management (2009)
11. T Tang, Z Wu, K Karhu, M Hämäläinen, Y Ji, An Internationally Distributed Digital Ubiquitous Living Lab Innovation Platform for Digital Ecosystem Research. in *Proceedings of the International Conference on Management of Emergent Digital EcoSystems (MEDES '10)* (ACM, New York, NY) 159–165 (2010)
12. X Jiang, C Zhang, A Web-based IT framework for campus innovation, in *Third International Workshop on Education Technology and Computer Science*, vol. 2. (Wuhan, China, 2011), pp. 90–93
13. A Pintus, D Carboni, A Piras, A Giordano, Connecting smart things through web services orchestrations, in *Proceedings of the 10th international conference on Current trends in web engineering (ICWE'10)*, ed. by Daniel F, Michele Facca F (Springer-Verlag, Berlin, 2010), pp. 431–441
14. LM De Souza, P Spiess, D Guinard, M Kohler, S Karnouskos, D Savio, Socrades: a web service based shop floor integration infrastructure, in *IOT 2008. LNCS*, vol. 4952, ed. by Floerkemeier C, Langheinrich M, Fleisch E, Mattern F, Sarma SE (Springer, Heidelberg, 2008), p. 50
15. F Jammes, H Smit, Service-oriented paradigms in industrial automation. IEEE Trans Ind Inf. **1**(1), 62–70 (2005). doi:10.1109/TII.2005.844419
16. C Pautasso, BPEL for REST, in *Proceedings of the 6th International Conference on Business Process Management (BPM '08)*, ed. by Dumas M, Reichert M, Shan M-C (Springer-Verlag, Berlin, 2008), pp. 278–293
17. E Chinthaka, REST and Web services in WSDL 2.0. http://www.ibm.com/developerworks/webservices/library/ws-rest1/ (2007)
18. Active Life Home: "http://www.activelifehome.fi" Accessed 20 Nov 2011
19. S Monte, Web portal implementation for elderly home living solution. Master Thesis, Aalto University (2010)
20. SIDE projecthttps://github.com/sizzlelab/side. Accesed 20 Feb 2012
21. A Rex, Design of a caregiver programmable assistive intelligent environment. Master's Thesis. Aalto University (2011)
22. H Järvinen, A Litvinov, P Vuorimaa, Integration platform for home and building automation systems. in *Proceedings of the IEEE Consumer Communications and Networking Conference*, CCNC'11, Las Vegas, NV 292–296 (2011)
23. SQLitehttp://www.sqlite.org/. Accessed 20 Feb 2012
24. I-jettyHttp://code.google.com/p/i-jetty/
25. Noelios Consulting (December 2008) Restlet, Lightweight REST framework for Java, homepage of the Restlet project. [Online]http://www.restlet.org
26. Apache ODE 1.3.5 http://ode.apache.org (2011)
27. ActiveBPEL 4.1 http://www.active-endpoints.com (2007)
28. C Pautasso, JOpera: process support for more than web services.http://www.jopera.org

doi:10.1186/1687-1499-2012-150
Cite this article as: Wu *et al.*: A web-based two-layered integration framework for smart devices. *EURASIP Journal on Wireless Communications and Networking* 2012 **2012**:150.