Coupling Enterprise Systems with Wireless Sensor Nodes: Analysis, Implementation, Experiences and Guidelines

Christian Decker[1)], Patrik Spiess[2)], Luciana Moreira sa de Souza[2)], Michael Beigl[1)], Zoltan Nochta[2)]
[1)]TecO, University of Karlsruhe, [2)]SAP Research, CEC Karlsruhe
{cdecker, michael}@teco.edu, {patrik.spiess, luciana.moreira.sa.de.souza, zoltan.nochta}@sap.com

Abstract
This paper presents an approach for closely coupling enterprise systems and tiny wireless sensor nodes that are embedded into physical items. Coupling is done by implementing services from the enterprise system on the wireless sensor node system and by letting part of the business logic run on the nodes. The paper analyses concrete application scenarios and discusses implications of this approach. The discussion is underpinned by implementation and testing of demonstration applications. The collected experiences lead to four guidelines a designer of such a system should follow.

1. Introduction

In the last years, we have seen an increased engagement to electronically support business processes that involve physical items. For instance, RFID technology stores identification and other data directly on the items and makes them wirelessly accessible by any computer system. Current solutions *strongly couple back-end enterprise systems (BES)* with physical items. At previously defined points, information from the items is acquired and sent to a BES. In the case of RFID, reader devices are installed on doors where items pass, in the assembly line, or even on trucks and other transport vehicles. Depending on the acquired data, actions are invoked from the BES to control the progress within the business process.

In this paper, we explore a novel approach for coupling physical items and enterprise systems. Instead of coupling BES and physical items at certain points, we *relocate process tasks* from the BES onto the items. The execution of these tasks is delegated to *physically embedded systems (PES)*, implemented as tiny, embedded computers, which are directly attached to the items. As a result, the process tasks run among the items.

The execution of relocated process tasks on PESs requires a more powerful enabling technology than state-of-the-art RFID transponders. In particular, this technology has to provide sufficient computing power to execute different tasks, a bi-directional wireless communication interface to enable a direct communication among the items, and onboard sensors in order to perceive the situation around the items. With wireless sensor network platforms, sufficient PESs are already available. In this paper, we will show how sensor networks can support the coupling between enterprise systems and physical items in business processes. The physical items augmented by a PES running relocated process tasks we call collaborative business items – or CoBIs.

The next section anchors our approach and its advantages in three concrete examples. The requirements for the coupling of business logic of relocated tasks are analysed in section 3. As a result, section 4 presents an architecture, which couples BES and physical items running process tasks and section 5 shows the implementation of it in one concrete example. Experiences we gained from this approach and guidelines to follow are given in section 6 before we conclude in section 7.

2. CoBIs Scenarios

In this section, we present the results of an application-oriented analysis where our approach of relocating process tasks is applied. In a research collaboration with BP, we explore safety issues when handling hazardous chemicals. Wireless sensor nodes that we attached on chemical drums execute the relocated process tasks collaboratively in order to *detect hazardous situations* like an exceeded storage limit, prohibited storage combinations of materials or an invalid storage area. Alerts can be both raised visually on the drums for notification of nearby workers and communicated to the BES.

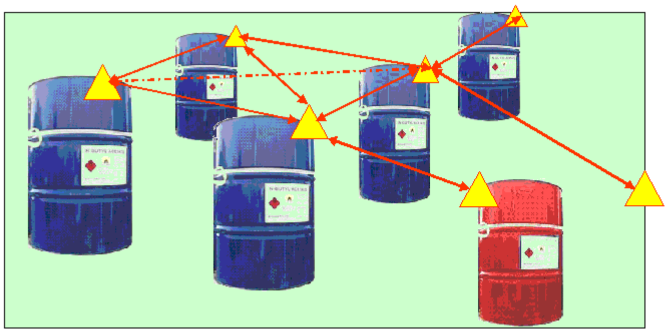
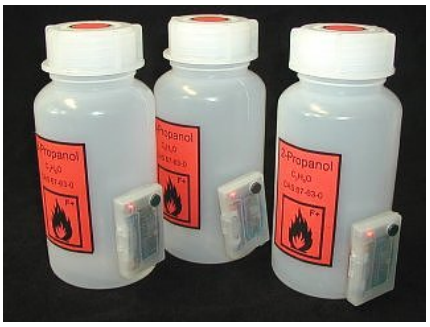

Figure 1. Left: Drums detecting hazardous situations (Source: BP); Right: Prototype implementation

The advantage of our approach in this scenario is fast response in dangerous situations. The in-situ detections and alerts shortcut long communication paths for notifications. Furthermore, relocated processes do not rely on a permanent connection to the BES. As a result, business applications can also handle very mobile items without being dependent on a full monitoring infrastructure.
In the second scenario, we explored process tasks on electronic and paper-based documents within SAP [1]. Electronic documents were managed by SAP's Record Management System (RMS). Their physical counterparts, e.g. legal contracts, were augmented by a PES called DigiClip, which monitored all handling activities. The requirement was to *keep both documents instances in sync*. Relocated process tasks are advantageous because they enable a permanent monitoring directly on the document in contrast to remote observation approaches. The coupling of those process tasks back to the enterprise system allows RMS applications to handle highly mobile paper documents in a very similar way like the electronic ones.
With eSeal [2] we explore an SAP supported supply chain scenario. The terms of delivery between two parties are transferred as a rule set down to a PES attached to the goods. Sensor nodes, as an implementation of this PES, execute these rules and establish an *electronic seal reporting any violation of rules* in a secure manner. As a result, high load on the BES is avoided and the coupled system scales well. Further, eSeals's processes on the goods guarantee the integrity of the terms of delivery while the goods are in transit, even if no support by a BES is possible.

3. Business Logic Coupling

From the perspective of a BES, coupling with PESs raises the following questions: How are the systems *communicatively coupled* and how is business logic *modelled and distributed to the nodes* of a PES? The broad heterogeneity of PES hardware and communication protocols has led to many different solutions for both problems.
Communication coupling. One could argue that PESs do not need to be connected to BESs. However, communicatively coupling PES and BES can increase the benefit for business processes. In the drum scenario described above, the drums detect violations of storage rules locally without back-end connection (e.g. during transportation), but it would be useful if violation incidents were

reported (even ex-post) to a BES. This leads to the question, how the coupling should be organized, and particularly, how the exposed abstraction of the PES to the BES should be designed.

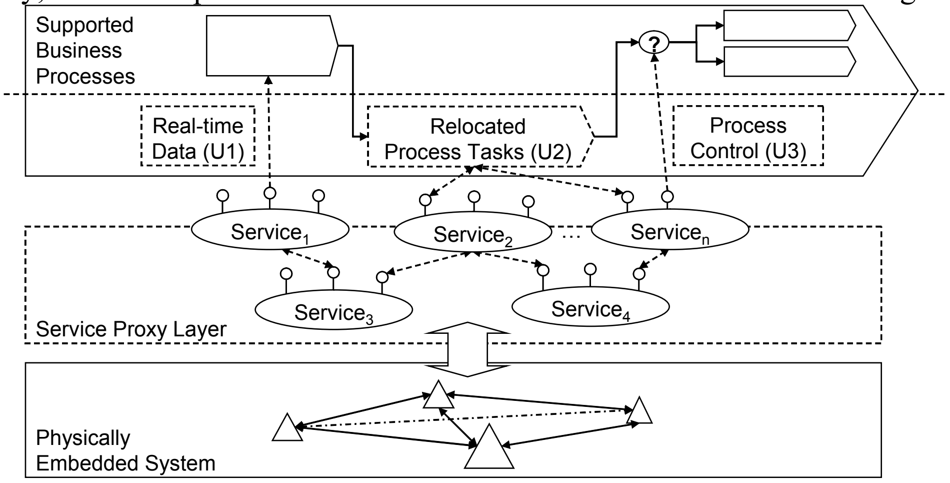

Figure 2. Coupling back-end enterprise systems with physically embedded systems using relocated process tasks

Figure 2 outlines a service-based approach. We assume the software running on each network node is organized as a set of services. Middleware components (middle box) create so called service proxies, i.e. run-time representations of the embedded services that the BES can access just like any other service in the BES. In the top part, you see three possible ways of using data from service proxies. Real-time data provided by a service proxy can be used to support the execution of one business process task (U1). Parts of a business process can be relocated to the PES and the business process can be executed by using the corresponding service proxies directly (U2) or the proxies can be used to control which process task to execute next if more than one option is available (U3).
U1 and U3 are straightforward ways of integrating PESs into business processes (e.g. for integrating RFID tracking data). We believe that U2 is a novel and most beneficial approach for many use cases, especially our application examples. It allows for management by exception, where the PES only notifies the BES of extraordinary situations, increasing scalability (the BES has to process less messages) and speed of detecting situations that require action (avoiding latency of control loop between PES and BES) and does not require a constant connection to the back-end.
Modelling and deployment of business logic and distribution to the PES. For implementing the embedded services that execute the relocated process tasks, one could write code for a microprocessor or an operating system like TinyOS. However, business users prefer tools that abstract consequently from hardware and operating system, such as interpreted programming languages [3], SQL-like query formats [4], or rules [5]. If the generation of these higher-level task descriptions is supported by convenient tools that support e.g. graphical drag and drop modelling, and deployment of embedded services, then business domain experts could model embedded services themselves.

4. Coupling Architecture

The technical goal of the architecture designed for CoBIs is to enable the coupling of relocated process tasks provided by a heterogeneous software and hardware landscape. We also stress the seamless technical integration into existing service-oriented platforms [6]. The architecture depicted in Figure 3 provides mechanisms for managing the relocated processes.
Service Design and Implementation. The relocation of a process task starts with specific design and coding. During the design phase, it is necessary to define a set of services that collaboratively can provide the functionality required by the business process. The main output of the design phase is a descriptive model defining services functionality and interfaces by which the services can be

called. We call this descriptive model CoBIs Language (CoBIL). This model is essential for the business applications because it gives a common interface that allows them to connect to the relocated process tasks and gives information reflecting technical needs of a given service.
The result of the implementation phase is an executable program code that can either run on different platforms or on a specific platform, which is stored in the *Service Repository*.

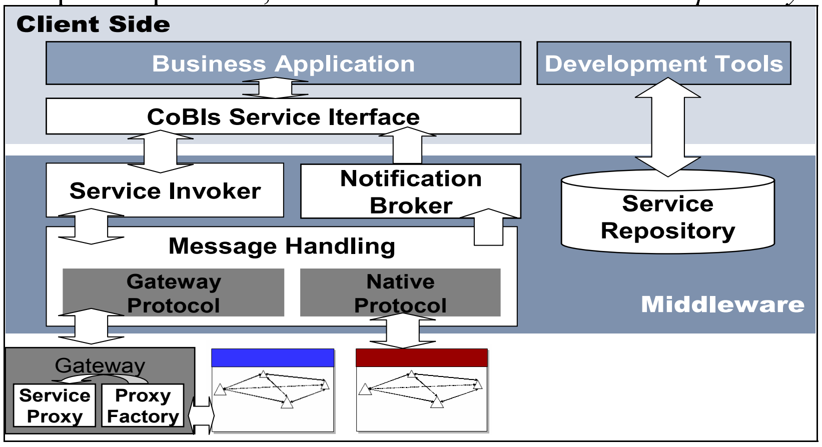

Figure 3. CoBIs relocated process management architecture

Service Invocation and Events. The ability to create invocations to and receive events from the relocated process tasks allows a transparent coupling of relocated process tasks to the business process being executed in the BES. The *CoBIs Service Interface* provides the functionality described in CoBIL to the business application. When the application makes an invocation, the *CoBIs Service Interface* forwards the request to the *Service Invoker*, which uses the *Message Handler* component to convert the invocation into the required protocol. The business application can also subscribe to receive events from the PESs. In this case, when new events are generated they are forwarded to the Notification Broker component, which will distribute the event among the subscribed business applications.
Message Handling and Gateway Architecture. Due to the heterogeneity of the PESs, it was important for our architecture to define a common abstraction for all platforms. This is needed in order to incorporate the relocated services residing on different platforms and to enable uniform management of the services. This is achieved through the component called *Message Handler*. This component uses two mechanisms for making conversions of the PES's protocol and the BES's protocol: the platform gateways and the native handler. These two approaches differ on the layer on which they convert the protocols. Figure 3 shows service proxies making the translation between service calls from the *MessageHandler* component and the platform specific message format. These proxies represent embedded services to higher layers of the architecture. When the platform does not provide a gateway, native handlers can be used to make a direct conversion between the PES protocol and the BES protocol.

5. Implementation

Prototype Implementation Setup. We implemented relocated business process tasks for the drum scenario from section 2. The drums were equipped with Particle sensor nodes as a PES for the task of detecting hazardous situations. The prototype implementation incorporated further a business application for the safe storage of chemicals supported by a SAP EH&S (Environment, Health and Safety) solution. Each Particle sensor node implemented services for the detection of the storage limit and incompatible material. All services were previously described in CoBIL documents. Since code generation support as suggested in section 3 was not available yet, the implementation of executable program code of a service was carried out manually.

Coupling to Back-end Enterprise System (BES). Service proxies on platform gateways (see Figure 3) couple PES and BES utilizing UPnP interfaces. Therefore, the relocated process task – not the platform devices – is represented in a platform-neural manner. UPnP is standardized and is specifically designed to enable a service-oriented architecture for embedded systems (www.upnp.org). The UPnP model of state variables supports the communication with the sensor nodes by generic primitives like state queries, e.g. the drum's current storage conditions, or state update, e.g. a storage parameter for a group of drums. Events from the sensor network, e.g. an alert due to a hazardous situation, can be directly supported by UPnP's GENA event system through subscriptions and notification on the state variables.

The key element in our current implementation is the automatic message translation between the different components within the architecture. It allows a type-safe and transparent communication between different PESs and the BES. The figure below depicts the message transformation sequence.

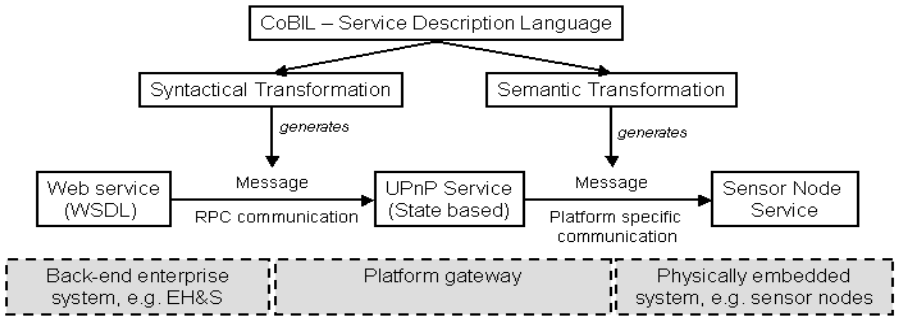
CoBIL – Service Description Language
Syntactical Transformation
Semantic Transformation
generates
generates
Web service (WSDL)
Message
RPC communication
UPnP Service (State based)
Message
Platform specific communication
Sensor Node Service
Back-end enterprise system, e.g. EH&S
Platform gateway
Physically embedded system, e.g. sensor nodes

Figure 4. Message transformation sequence

Crucial for the transformation are the service description documents in CoBIL. In first step, the CoBIL document serves as an input for a syntax transformation in the communication between the web services in the EH&S solution and the UPnP service proxies on the platform gateways. Basically, the transformation strips down WSDL documents and maps specific identifiers to the correct state variables. This XML-to-XML transformation generates messages, which can be understood by both, the web services and the UPnP service proxies. Independently from the underlying PES, this step can be re-used. The message generation below the platform gateway runs proprietary. For each platform, a manually declared semantic transformation maps state variables to the platform specific protocol format. In our implementation, the UPnP service proxies utilize message templates. Several state variables are aggregated into one message via a template in order to take low bandwidth and energy constraints of the sensor nodes into account. Native message handlers in the architecture leave out the syntactical transformation and proceed directly with the semantic one. However, this binds the BES services strongly to a specific underlying PES. Due to two-step transformation, an additional PES like the µNode sensor nodes for an in-situ environmental monitoring as an extension of the drum scenario was integrated within one day.

6. Experience and Guidelines

During installation and run of the drums scenario, we collected several experiences both in the practical implementation and in the more general design of the system and architecture. This section should give an insight on the most important experience that we made up to now.

Technical problems. First, we gathered several experiences by technically coupling PES to (existing) BES – in hardware, software, and architecture. We found that enterprise and wireless sensor systems follow an almost orthogonal approach: The logic behind enterprise applications assumes certainty from the inputs and is often not programmed to make decisions based on uncertain information. Sensor networks on the other hand come with an inherent uncertainty factor, simply because the technical conditions make it impossible to provide 100% reliability. In practice, independently from the technical platform, most unreliability originates from the RF-based network connection. Minimizing the RF path is therefore a necessary requirement for a successful coupling between PES and BES. Consequentially, mobile ad-hoc network routing solutions are not an option for most application cases. In our experience, a compromise is a good solution to bridge this gap: While parts of the enterprise application must be enabled to deal with unreliable information to a certain degree, from the sensor networks the solution with the most reliability should be selected.

Application complexity. In a setting with several BESs and a large number of PESs an abstraction from the technical solution hides the underlying complexity and enables a software developer to concentrate on the application logic. In our experience, this approach fails when a technical problem requires application domain knowledge to be solved. A chemical container supervision application with dangerous goods cannot accept a temporary sensor break down, but needs to send out an alarm. This failure in sensor reading must therefore be accessible to the application to trigger application specific reactions. In our experience, it is important to identify selected technical situations and parameters that need to be made available to the application and to enable applications to understand the nature of this situation.

Application reaction. In many pervasive and ubiquitous computing systems, as well as in "traditional" enterprise applications, reaction of an input is not computed locally but done on a central server. Reaction commands are then forwarded back to the actuators in a PES. We found that this approach has several disadvantages. Firstly, reaction time increases because of the inclusion of several communication and processing steps. Secondly, the error rate increases, leading to a decrease of performance of the overall system. Thirdly, due to the involvement of many devices and application modules system complexity increases. We tried to minimize complexity, communication and processing steps by keeping processing of system output local if the input causing the reaction is also local. This strategy reduces both reaction and error handling complexity of the system as a minimal set of devices and functional modules are involved.

Coupling back-end enterprise and physical embedded systems. In our experience, a large amount of time is spent for defining, describing, and implementing interfaces between the various parts of a coupled system. We found it advantageous for both the system itself and the communication between the various developers, to rely on standards. We also found that using service orientation addresses our application settings best. For us, UPnP seems to be a natural choice as it is already directly supported by some components used within the overall system. For some other parts it was quite simple to integrate them into the UPnP. For instance, the GENA event model from UPnP could be used without modification to notify higher layers about events in the sensor network. For other parts integration is more difficult and still up to research: While some of the PES communicate via exchange of information, the UPnP state model allows only a RPC semantic for the communication (SOAP). We coupled these systems via a UPnP gateway, but this reduced some of the information and flexibility of the sensor network

For our further research and development, we derived guidelines based on our experience. We are convinced these guidelines will help us to focus our research and to avoid pitfalls in future work. We found the below four guidelines most important:

- **Minimize technical problems.** Minimizing technical problems makes a solution more feasible, more acceptable for users in industrial settings, and gives us the opportunity to concentrate on our research task. Especially, stay away from (in practice) highly unreliable approaches as multi-hop sensor networks.

- **Identify critical technical problems and make them visible to the application**. Critical decisions have to be made by the application. Technical problems should not be shielded since this may lead to improper application's reactions. Applications should be able to understand technical problems and provide a solution strategy.
- **Keep application reactions local if possible (Local loop).** Try to react on an input or problem with a minimum number of remote systems involved. This way it minimizes the overall complexity and the number of interfaces between the various parts of the system. A good guideline is to keep input/reaction local.
- **Use standards for coupling back-end enterprise and physical embedded systems.** Standards are a common ground for developers from various parts of a system and are therefore well suited as a technical interface between these parts. Service oriented coupling, e.g. UPnP, seems to be a good choice here.

7. Conclusion and Future Work

We presented a novel approach for coupling service-oriented enterprise systems with physical items in business processes by relocation and execution of process task on wireless sensor nodes. The process tasks are executed collaboratively among the items, which we named as collaborative business items – or CoBIs. The usage of CoBIs lead to shorter communication paths and support business processes, which would otherwise rely on a permanent connection to the back-end. The experiences from implemented demonstration applications resulted in four guidelines a system designer should follow. Future research follows these guidelines and addresses in particular appropriate error handling and resolving strategies within the business application for critical technical problems occurring in the embedded sensor network. Furthermore, we plan a large-scale, long-term application trial in a real business environment to deliver detailed insight. Finally, our goal is to build efficient and user-friendly tools for monitoring, management, and support for the whole service lifecycle from modelling and deployment to termination and removal.

References

[1] C.Decker, M.Beigl, A.Eames, U.Kubach. DigiClip: Activating physical documents. IWSAWC 2004, In proceedings of the ICDCS 2004, Tokyo, Japan

[2] C.Decker, M.Beigl, A.Krohn, U.Kubach, P.Robinson. eSeal - A System for Enhanced Electronic Assertion of Authenticity and Integrity of Sealed Items. Pervasive 2004, Vienna, Austria

[3] P.Levis, D.Gay, D.Culler. Bridging the Gap: Programming Sensor Networks with Application Specific Virtual Machines. UCB//CSD-04-1343. August 2005.

[4] S.Madden, M.Franklin, J.Hellerstein, W.Hong. TinyDB: An Acqusitional Query Processing System for Sensor Networks. ACM TODS, 2005

[5] M.Strohbach, H.-W.Gellersen, G.Kortuem, C.Kray. Cooperative Artefacts: Assessing Real World Situations with Embedded Technology. Ubicomp 2004

[6] A.Woods. Enterprise Services Architecture, Galileo Press, 2004