



Managing Technical Processes Using Smart Workflows

Matthias Wieland, Daniela Nicklas, Frank Leymann

Institute of Architecture of Application Systems

University of Stuttgart

Universitätsstraße 38, 70569 Stuttgart, Germany

<http://www.iaas.uni-stuttgart.de>

in: Towards a Service-Based Internet, First European Conference, ServiceWave 2008, Madrid, Spain,
December 10-13, 2008. Proceedings.

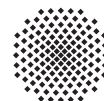
See also $\text{BIBT}_{\text{E}}\text{X}$ entry below.

$\text{BIBT}_{\text{E}}\text{X}$:

```
@inproceedings{INPROC-2008-111,  
  author = {Matthias Wieland and Daniela Nicklas and Frank Leymann},  
  title = {Managing Technical Processes Using Smart Workflows},  
  booktitle = {Towards a Service-Based Internet, First European Conference,  
  ServiceWave 2008, Madrid, Spain, December 10-13, 2008. Proceedings},  
  editor = {Petri Maehoenen and Klaus Pohl and Thierry Priol},  
  publisher = {Springer},  
  series = {Lecture Notes in Computer Science},  
  volume = {5377},  
  pages = {287--298},  
  month = {December},  
  year = {2008},  
  isbn = {978-3-540-89896-2},  
  language = {english},  
  cr-category = {H.4.1 Office Automation},  
  department = {University of Stuttgart, Institute of Architecture of Application Systems},  
}
```

© 2008 Springer-Verlag.

See also LNCS-Homepage: <http://www.springeronline.com/lncs>



Managing Technical Processes using Smart Workflows

Matthias Wieland¹ *, Daniela Nicklas², and Frank Leymann¹

¹ Institute of Architecture of Application Systems, Universität Stuttgart
{wielanms, leymann}@iaas.uni-stuttgart.de

² Computer Science Department, Carl von Ossietzky Universität Oldenburg
dnicklas@acm.org

Abstract. Technical processes that are crossing the boundary to the physical world can be found in many application domains, like logistics or in Smart Factory environments. We show how these processes can be realized by so-called Smart Workflows. To integrate external information sources like context provisioning services, we introduce the Integration Process architecture pattern. This pattern generally solves the problem of integrating different complex systems that provide functional similar services with non-fitting interfaces into workflows. The pattern allows that workflows use simple domain specific interfaces that are the same for any of these systems and by that allow the exchange of underlying systems without changing the workflows. This is accomplished by reducing the interface complexity of the systems via a hierarchical Web Service stack that reaches from the lowest technical granularity needed by IT experts to the domain specific granularity needed by the domain experts. Furthermore the paper presents a concrete realization of the pattern for integrating different context provisioning systems into workflows.

1 Introduction

Technical processes that are crossing the boundary to the physical world, like production processes or maintenance processes in a smart factory, are not well supported by workflow technology yet. Our vision is compared to the current state of the art to automatically execute and control such technical processes. This leads to the same amount of flexibility enterprises gained by introducing workflow management systems [1]. Furthermore, technical processes are enabled to easily interact with the back office, bridging the gap between business and production. Modeling a technical process becomes similar to modeling of a business process. The difference is that for modeling technical processes various information about the physical world like current state of all involved real world entities is needed, e.g., machines, tools, and workers in a production environment. If this information—often referred to as context [2]—is captured automatically by sensors in a smart environment, technical processes can be executed pervasively

* This work was funded by the Collaborative Research Center *Nexus: Spatial World Models for Mobile Context-Aware Applications* (grant SFB 627).

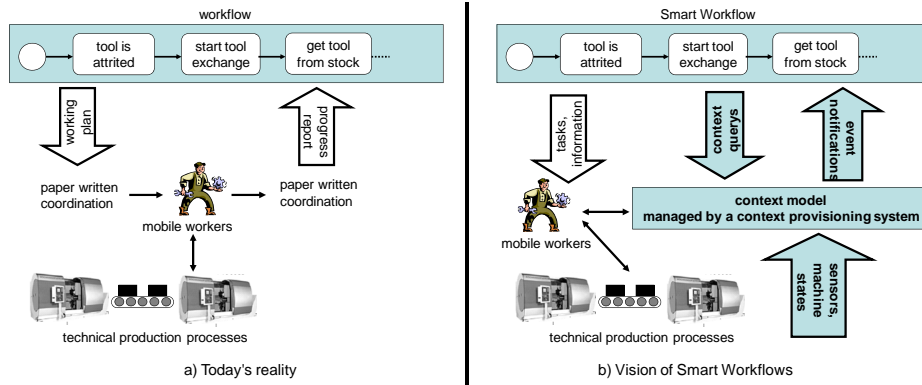


Fig. 1. Smart workflows: Incorporate context information into workflow technology

and adapt to changes in their physical context. We call such context aware processes *Smart Workflows* (SW). To realize this vision we employ context data managed in a *context model* (see Figure 1). In Section 1.1 we illustrate a sample scenario in a Smart Factory [3].

While designing a service oriented system, a main challenge is to choose an adequate granularity of the services. It could be either generic and rather technical (like complex SQL queries) or domain-specific and rather semantic for a given application domain (like functions for retrieving machine tool information for a set of parameters). This choice cannot be made in general. Also there are no standards available how services can be re-used, e.g., in a service hierarchy where services are derived from each other (comparable to a class hierarchy in a programming language).

For coping with both problems we developed the integration process architectural pattern. Integration processes (IPs) are processes that are derived from each other, i.e., their interfaces get more restricted and thereby more concrete and domain specific. With that concept it is possible to build up a hierarchical Web Service stack that reaches from the lowest technical granularity needed by IT experts to the domain specific granularity needed by the domain experts.

In the remainder of this section we describe a concrete scenario and derive challenges and requirements. After a discussion of the related work in Section 2 we present the main contribution of this paper, the concept of IPs, in Section 3. In Section 4 we show a concrete realization of that concept, the context integration processes (CIPs). Finally, we describe the prototype implemented for evaluation and proof of concept in Section 5 and conclude the paper in Section 6.

1.1 Example scenario: Machine maintenance process

The following process is used as concrete example throughout the paper and represents the kind of processes we implement with Smart Workflows (SWs): A sensor in a machine measures the attrition of an installed tool, a drill. When the

tool is attrited, a SW starts to arrange the replacement of that tool. First, the SW finds out whether a spare drill is in stock. If not, it starts a business process in the back office that orders new tools and registers a notification for when the spare part is available. Now that a spare part is in stock, the SW creates a human task: somebody should transport the spare part from the storage to the measuring device to prepare it for installation. The exact location of the spare part within the storage and the available measuring device are part of this task. Also, the SW monitors the execution of this task using location information. Soon, a mobile agent (a transport robot or a worker) that is near to the storage picks up this task. When the transport task is completed, the SW creates a preparation task which can be picked up by another agent that is capable of doing this work. Again, the completion of this task is monitored so that a transport task to the machine can be invoked. Arriving there, the old drill can be exchanged by the spare part. The last action triggers an event for the tool life-cycle SW of the old drill that now checks whether the old drill can be refitted or has to be recycled.

1.2 Challenges and Requirements

To realize scenarios as the one given, several challenges must be addressed:

1. *Smart workflow modeling:* A domain expert should be able to easily model SWs. For that, new concepts for workflows are needed, like context-based decisions, or context information requests. These concepts require the access of context data. Since the domain expert should not be burdened with unnecessary technical details (like the syntax of a general context query language), the context access for SWs needs a high number of simple, domain-specific functional interfaces. Here, we need a design that offers good maintainability of these domain-specific functions because they are often extended to new application needs. Finally, the SWs must be executed in an efficient manner. The easiest way would be to use existing workflow execution engines. Thus, existing standards like BPEL [4] should be used for modeling SWs whenever possible.

2. *Context provisioning:* We need an environment to capture, manage, and provide context information in an efficient way. This information is distributed over various systems within the factory. Also, it varies regarding update rate, selectivity, usage for selection, and required data quality. Hence, a single server solution is not feasible. In the Nexus project², we developed a model-based, extensible context provisioning platform that is able to cope with high amounts of distributed context data [5]. Since context modeling and management is expensive, the main goal is to provide reusable context information for different applications, like in a shared database. Thus, the interfaces of such a platform are rather generic: for context information requests, a flexible query language is provided; for event-based interaction, complex physical world events can be declared using an event definition language.

3. *System integration:* Since the context access functions used for SW modeling are often domain-specific they should be understandable by domain experts.

² Nexus project website: <http://www.nexus.uni-stuttgart.de/index.en.html>

Hence, there is a gap between generic provisioning of context (e.g., query languages) and the concepts needed in the SW (e.g., "spare tool available?"). This context access functions must be organized in an extensible modular architecture that allows the easy reuse of existing functions. Also, they should be realized using the same modeling techniques as for the SWs to allow domain experts to easily extend the available domain-specific functional interfaces.

2 Related Work

IBM's information integration for BPEL (II4BPEL) [6] allows simple and efficient access of relational database systems, using SQL, from within business processes. Other vendors provide comparable SQL support in BPEL [7]. These solutions could be used to integrate context information into SWs. However, only BPEL engines implementing this extension could be used. Our approach works with every engine conforming to the BPEL standard.

The PerCollab System [8] extended workflow technology to support adaptive collaboration between people in business processes. While this work can be used to enhance the collaboration of workers in a shop floor, it does not take the context of the tasks or activities of the process into account. In [9], a ubiquitous workflow service framework for the development of context aware services named uFlow is introduced, which defines an own workflow description language and workflow engine. Again, the usage of standard BPEL allows us to leverage the advantages of existing workflow engines, e.g., the process management or server stability.

There are many approaches specialized on the handling of context data like the Context Toolkit [10], the Location Stack [11], or the CML based approach by Henriksen et al. [12]. They provide simple software components to access sensor data, to refine the raw sensor data to some higher level and for the interpretation of situations based on the sensed data. In our approach we need a more powerful context model which offers us the ability to model and access a wide range of common objects in a unified manner. We also model our applications as workflows, thus we are rather focused at accessing software components defined as Web Services.

In previous work we proposed an extension to BPEL to cope with the special needs of context aware workflows, called Context4BPEL [13]. To execute Context4BPEL workflows an extended workflow engine is needed. The SW approach in this paper can be either build on top on Context4BPEL or using only standard BPEL. In contrast to the Context4BPEL approach [13], this paper presents a full end-to-end solution providing both domain-specific workflow functions needed by domain-experts and a generic execution architecture using standard BPEL workflow technology. Nevertheless the CIPs on the higher layers, especially the domain-specific CIPs are needed in both cases because it is not feasible to define a workflow modeling language that contains all domain specific extensions. In [14] we presented a short work in progress overview of the SW vision and the layered system architecture needed for realization.

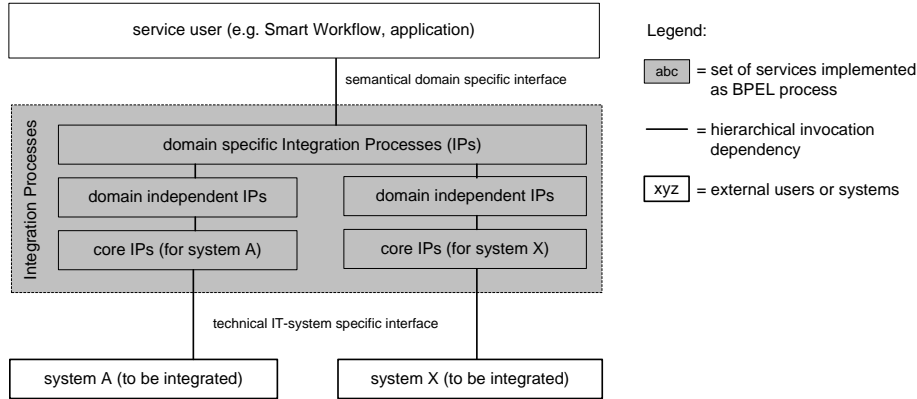


Fig. 2. Architectural pattern for the concept of integration processes

3 Concept of the Integration Processes Pattern

The concept of Integration Processes (IPs) addresses the service granularity problems described in Section 1. On Figure 2 the generic design of the Integration Process architecture pattern is shown.

At the bottom of the architecture the *systems* that provide generic services (data and functionality) are shown (system A, system X). These systems are the artifacts of integration. The aim is to access them from the top most part of the architecture, the *service users* (here: a Smart Workflow (SW)). The SW represents the technical processes executed in the production environment in our scenario. For defining SWs, we chose the Business Process Execution Language (BPEL) [4]. This language can be used to orchestrate Web Services. The SWs use the functionality provided by the IPs analog to external Web Service invocation. In future work, the BPEL extension for subprocesses [15] will be used to invoke the IPs.

The *IPs* are the main part of the architecture. They are located between the systems and the users and are implemented similarly as BPEL workflows. Hence, they are accessible as Web Services. There are three different granularity levels for IPs: core, domain independent, and domain specific IPs. The lowest level processes are the *core IPs*. They are responsible for the integration of each service the systems provide. Each service has its own interfaces and one concrete core IP that wraps that interface.

On the highest level the *domain specific IPs* are located. The SWs on the application layer use them to access the services of the systems in a simplified manner. In general, the domain specific IPs receive fewer parameters than the other IPs. The aim of this design is that domain experts can easily model SWs using only the domain specific IPs. The interface parameters of these IPs derive from the application area and thus form the terminology of the domain experts. If functionality is missing, the domain expert could create a request for that

interface and an information scientist could implement this functionality by deriving from and thus reusing existing domain independent or core IPs.

The most important level is located in between the other two layers. The *domain independent IPs* mediate between the core IPs with complex interfaces and the domain specific IPs with easy-to-use interfaces. The domain independent IPs can be reused in different application domains because they offer a general and not application specific functionality. The main functionality of the IPs on this layer is the transformation from the simple request/response message formats of the IPs on the higher levels to the more complex request/response message formats of the IPs on the lower level.

The reusability of services is further enhanced by the hierarchical structure of the pattern. This has the following advantages: if the interfaces of system A are changed, only the core IPs wrapping that interfaces have to be changed. The process does not notice the changes and even the IPs on the higher level do not have to be changed. In addition, if a new interface is requested by a domain expert it can be provided easily by attaching it on top of an appropriate existing IP.

The hierarchical structure should be part of the naming scheme of the processes, i.e., the processes on the core layer should have a name describing their functionality (e.g., FunctionA). The processes on the domain independent layer should add their specialization characteristics to the end of the name of the core IP they are using (e.g. FunctionAFiltered). In contrast, the processes on the domain specific layer should use names from the application domain and not technical functionality descriptions (e.g. QueryTool).

4 Realization of Context Integration Processes

To solve the challenges described in Section 1.2 we implemented following system based on the Integration Processes pattern. On system level we use the Nexus Context Provisioning System, which solves challenge 2 (context provisioning). On the user level we use BPEL or Context4BPEL [13] for modeling the technical processes in the Smart Factory, which addresses challenge 1 (Smart Workflow modeling). The integration of the Nexus system (challenge 3) or other context provisioning systems is realized by a set of implemented Context Integration Processes (CIPs) described in Section 4.2.

4.1 Nexus Context Provisioning System

To provide the highly dynamic context information for SW in an adequate manner, a mature context management is needed. We use the Nexus platform for that purpose. It is an open platform that supports the development of various context aware applications. It is based on a common context model, the so-called Augmented World Model (AWM). This context model is used to integrate and cache the highly dynamic context information from various sensor sources and

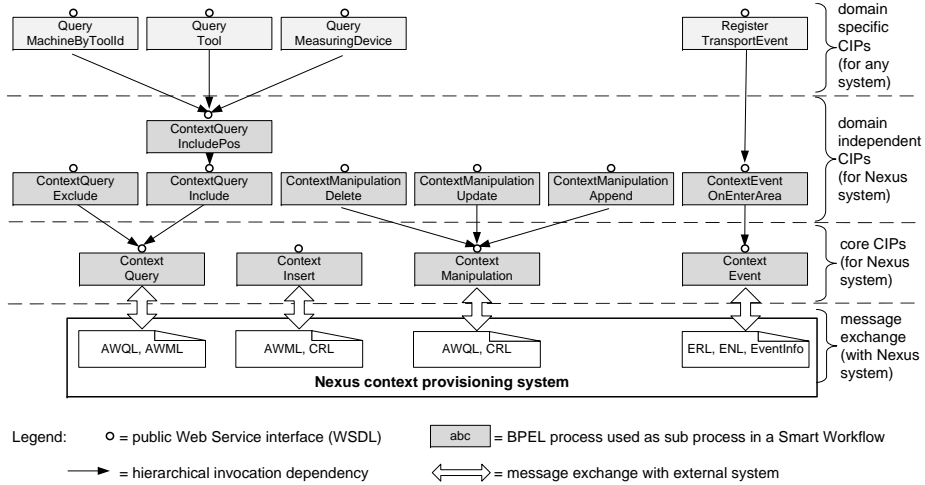


Fig. 3. Context Integration Processes for the Nexus context provisioning system

to provide an abstraction for different context aware applications [5]. It models context data in different areas, like geographical data, dynamic sensor data, infrastructural context, or related information such as documents. Nexus consists of basic services and value added services to provide the context information. Nexus is a federated system and there exist many different kinds of implemented and ready to use context servers for diverse needs [5]. To exchange context information between applications and the Nexus platform, several data formats have been defined [16]:

- the Augmented World Modeling Language (AWML) for data modeling and serialization of context information;
- the Augmented World Query Language (AWQL) for querying and manipulating context information. The results for manipulations (success or error) are reported with the Change Report Language (CRL);
- the Event Registration Language (ERL) and the Event Notification Language (ENL) for context event observation;
- the map service can be used to generate topographic maps based on the objects (e.g. buildings) stored in the context servers. As exchange formats the map service offers the Map Predicate Language (MapPL) and the Map Modeling Language (MapML); and
- the navigation service is responsible for calculating travel routes and offers the Navigation Parameter Language (NPL) and the Navigation Result Language (NRL) for exchanging data.

4.2 Context Integration Processes

Figure 3 shows as a concrete implementation of the Integration Process pattern for the integration of the Nexus services. We call this concrete set of IPs Context

Integration Processes (CIP) hence it is used for integration of context data into Smart Workflows (SW).

As *core CIPs* we need the following four processes:

- *ContextQuery* is used for querying context data. It integrates the querying functionality of the Nexus federation component. ContextQuery integrates the AWQL format for the query declaration and the AWML format for the result presentation.
- *ContextInsert* is used for inserting new context data. It integrates the AWML format for describing data objects that have to be inserted to the context model and the CRL format for reporting the result of an insert operation.
- *ContextManipulation* is used for changes on existing context data. It integrates the AWQL format for the manipulation request and the CRL format for manipulation results.
- *ContextEvent* integrates the functionality of the event component from the context provisioning layer. It integrates the ERL format for the registration of events and the ENL format for event notifications (more detailed description in 4.2).

The CIPs on this level have the advantage of offering the full functionality of the integrated Nexus components by supporting the complex exchange formats of the Nexus components. However, for client processes (e.g., SWs) the interfaces of these CIPs are too complex for an easy usage.

Hence, following *domain independent CIPs* are defined that simplify these interfaces:

- *ContextQueryExclude* allows the blanking out of a set of attributes to minimize the size of the result set. It uses the core CIP ContextQuery for the query execution.
- *ContextQueryInclude* allows the selection of a set of attributes. Only this set of attributes is included in the objects of the result set. That can be used to downsize the messages that have to be transferred. It uses the core CIP ContextQuery for the query execution.
- *ContextQueryIncludePos* is used by QueryMeasuringDevice, QueryTool and QueryMachineByToolId. It returns just the location attribute of an object. It gets a restriction parameter that holds the condition for the objects that should be searched for. It uses the CIP ContextQueryInclude for further processing of the query.
- *ContextManipulationDelete*, *ContextManipulationUpdate*, *ContextManipulationAppend* are specializations of the core CIP ContextManipulation. They allow the deletion of complete objects in the context model, the update of already existing objects and the extension of existing objects with new attributes.
- *ContextEventOnEnterArea* is used by RegisterTransportEvent. It requires the ID of the mobile object, the observation period of the event, the geographic area that should be observed, a threshold probability for the event firing, and further configuration parameters (more detailed description in 4.2).

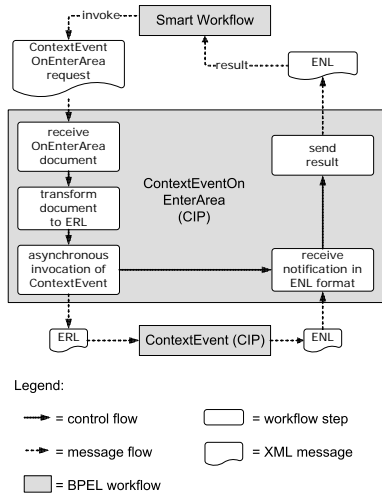


Fig. 4. ContextEventOnEnterArea

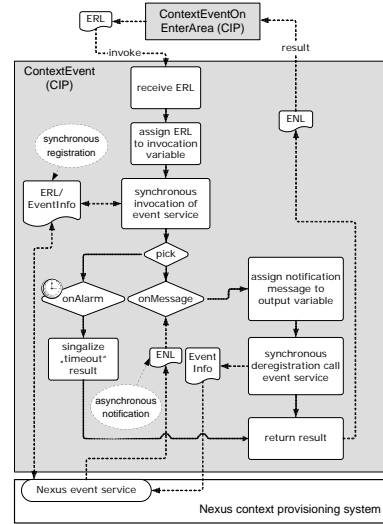


Fig. 5. ContextEvent

For the Smart Factory application domain we modeled following *domain-specific CIPs* (see Figure 3):

- *QueryMachineByToolId* queries the machine where the tool has to be exchanged. This CIP takes the ID of the old tool that should be exchanged by the worker, and searches for the machine with this tool installed. The return value is the location of the machine.
- *QueryTool* queries a tool as spare part for a machine. This CIP takes a tool type as parameter and returns the exact geographic location of an available and not worn out tool in the factory.
- *QueryMeasuringDevice* queries a measuring device, where a tool can be calibrated by a factory worker. This CIP accepts only a location area parameter, e.g., the factory area, because it is intended to search measuring devices in a given area. It returns the location of the measuring device.
- *RegisterTransportEvent* observes the transportation process of a tool. It gets three parameters: an arbitrary mobile object, a geographic area as transport destination and the observation duration of the event.

Detailed realization of two example CIPs To illustrate the concrete realization, we show the detailed structure of the core CIP ContextEvent and the domain-independent CIP ContextEventOnEnterArea building up hierarchically on that core CIP. In Figure 4, we see the realization of the ContextEventOnEnterArea CIP. It calls the ContextEvent CIP as a subprocess to register its specialized event (an object enters a specified area). Therefore it offers a simpler interface to clients by defining its own request format. The new request format holds, compared to the ERL format, only fewer parameters necessary for that special event type. Thus, a transformation step is needed before the invocation

of the ContextEvent CIP to adapt the more complex ERL format. The received result from the ContextEvent CIP will be forwarded to the client. The client has thus unmodified access to the complete result data described in ENL format.

In Figure 5, the realization of the ContextEvent CIP is illustrated. A client process (e.g., the previous ContextEventOnEnterArea CIP) sends an event registration request in ERL format to the ContextEvent integration process. ContextEvent itself takes the message and sends it synchronously to the event service located on the Nexus platform. The request is send synchronously with an immediate response that confirms the registration of the event. However, the notification of the event occurrence is performed asynchronously. The ContextEvent CIP stores the current process state and waits for the event notification by suspending the process activity. When a notification arrives, the process wakes up and continues processing by forwarding the result to the client process. Alternatively, a timer activates the process if the event was not observed in a given time, thus allowing the client process to take appropriate action in this case. This CIP is used to integrate the event service of the Nexus platform by integrating the complex documents described in ERL and ENL.

This example showed the hierarchical approach of enhancing and reducing the complexity of the requests and responses and showed how the CIPs can be implemented as BPEL workflows. All other CIPs are implemented and wired the same way.

5 Implemented Prototypes

To evaluate the concept of SWs, we implemented two prototypes in the Smart Factory environment [3]. This example factory contains a storage area, a measuring device for drills, and some machines that use different drills for producing personalized plastic coins. Furthermore several sensors are available for context observation. The drills are equipped with RFID tags for identification. Tools are transported in an intelligent transport box. This box is tracked by an UbiSense³ indoor positioning system. Different possible locations of the tools (i.e., the transport box, the storage, or the machine) are equipped with a RFID reader. Thus, the position of a specific tool is available anytime using indirect localization. Also, the usage time of drills within machines is measured. This allows the context management system to calculate the attrition of the tools. The whole factory layout—i.e., the positions of the machines and workstations—is managed by a Nexus context model. Also, the transport cart, which is used to transport the tool boxes, is tracked by the UbiSense system. Transport carts are moved only by workers. Hence, the system can infer the positions of workers without tracking them directly. This improves the acceptability of the system due to the enhanced privacy for workers.

Within this setup we implemented following two SWs: First, the machine maintenance process described in Section 1, and secondly, a process for handling

³ UbiSense Real-time Location System: <http://www.ubisense.de/>

individual customer orders. For the execution of the modeled BPEL processes we used the Oracle BPEL process manager⁴. The provided Dashboard gives a good overview of all deployed, in-flight, and recently completed BPEL processes. Furthermore the Oracle BPEL process manager provides a Human Task Management Service with web interface which is used for the interaction with the workers. A worker can list and access all his ongoing tasks via the human task manager. Thus the workflows used in the Smart Factory can be controlled both directly by human interaction and pervasively by observing changes and events in the real world (e.g., movement of the transport cart). From our experience with these first prototypes we derived the great need for an easy maintainable and hierarchically structured service layer. So we invented the integration process architecture pattern and implemented the CIPs as BPEL workflows therewith. The most important benefit was that domain experts could now better understand and use the resulting workflows because of the mainly domain specific interfaces called by the SWs.

6 Conclusion

In this paper we introduced the notion of Smart Workflows (SWs), which cross the boundaries to the physical world. Many application areas like logistics or the upcoming domain of Smart Factories could benefit from workflow technology if these are enhanced by context information to SWs. To realize SWs, we need context-based features at the process modeling level, an efficient provisioning of context information, and a maintainable integration layer for providing the context information at the right semantical level.

Our prototype leverages the usage of standards like WSDL and BPEL, for both the realization of SWs and the provisioning of various context services by Context Integration Processes (CIPs), which are used to bind an off-the-shelf workflow engine (Oracle) with an existing context provisioning platform (Nexus). This binding is realized by implementing the Integration Processes pattern described in this paper. By splitting up different semantical layers (from general context access to highly application-dependent services) into a hierarchy of different CIPs, we achieved a high maintainability and reusability of the services. This approach dramatically facilitated the development of SWs, which is an important pre-condition for the adoption of that technology by industry. Like in business process engineering, SWs should be developed by domain experts using modeling techniques rather than be programmed by computer scientists. The usage of BPEL as a well known workflow modeling language speeds up the modeling of new SWs compared to using an application specific workflow language, or even against programming a context aware application supporting the process (for example in Java). It enables domain experts to model their SWs themselves. These resulting workflow models can be used very good as a basis for discussion between domain experts and the computer scientists.

⁴ Oracle BPEL Process Manager <http://www.oracle.com/technology/bpel/>

References

1. Leymann, F., Roller, D.: Production Workflow: Concepts and Techniques. Prentice Hall PTR (1999)
2. Dey, A.K.: Understanding and Using Context. *Personal and Ubiquitous Computing* **5**(1) (2001)
3. Westkaemper, E., et al.: Smart Factory - Bridging the gap between digital planning and reality. In: Proc. of the 38th CIRP Intl. Seminar on Manufacturing Systems. (2005)
4. OASIS: Web Services Business Process Execution Language Version 2.0. <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html> (4 2007)
5. Großmann, M., et al.: Efficiently Managing Context Information for Large-Scale Scenarios. In: Proc. of the Third IEEE Intl. Conf. on Pervasive Computing and Communications. (2005)
6. IBM: Information Integration for BPEL on WebSphere Process Server. <http://www.alphaworks.ibm.com/tech/ii4bpel> (2005)
7. Vrhovnik, M., et al.: An Overview of SQL Support in Workflow Products. In: Proc. of the 24th International Conference on Data Engineering. (2008)
8. Chakraborty, D., Lei, H.: Pervasive Enablement of Business Processes. In: Proc. of the Second IEEE Intl. Conf. on Pervasive Computing and Communications. (2004)
9. Han, J., et al.: A Ubiquitous Workflow Service Framework. In: ICCSA (4). Volume 3983 of Lecture Notes in Computer Science., Springer (2006)
10. Salber, D., Dey, A.K., Abowd, G.D.: The Context Toolkit: Aiding the Development of Context-Enabled Applications. In: CHI '99: Proc. of the SIGCHI Conf. on Human factors in computing systems, ACM Press (1999)
11. Hightower, J., Brumitt, B., Borriello, G.: The Location Stack: A Layered Model for Location in Ubiquitous Computing. In: Proc. of the Fourth IEEE Workshop on Mobile Computing Systems and Applications. (2002)
12. Henriksen, K., Indulska, J.: A Software Engineering Framework for Context-Aware Pervasive Computing. In: Proc. of the Second IEEE Intl. Conf. on Pervasive Computing and Communications. (2004)
13. Wieland, M., et al.: Towards Context-Aware Workflows. In Pernici, B., Gulla, J.A., eds.: CAiSE07 Proc. of the Workshops and Doctoral Consortium Vol.2, Tapir Academic Press (2007)
14. Wieland, M., Kaczmarczyk, P., Nicklas, D.: Context Integration for Smart Workflows. In: Proc. of the Sixth IEEE Intl. Conf. on Pervasive Computing and Communications. (2008) (work in progress paper).
15. Kloppmann, M., et al.: WS-BPEL 2.0 Extensions for Sub-Processes. Whitepaper, IBM, SAP AG (2005)
16. Bauer, M., et al.: Information Management and Exchange in the Nexus Platform. Technischer Bericht Informatik 2004/04, Universität Stuttgart, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme (2004)