

Integrating Sensor Networks with Business Processes

Patrik Spiess
SAP Research
Vincenz-Prießnitz-Str 1
76131 Karlsruhe, Germany
+49 721 6902 67
patrik.spiess@sap.com

Hannes Juetting
SAP Research
Vincenz-Prießnitz-Str 1
76131 Karlsruhe, Germany
+49 721 6902 67
hannes.juetting@sap.com

Harald Vogt
SAP Research
Vincenz-Prießnitz-Str 1
76131 Karlsruhe, Germany
+49 721 6902 67
harald.vogt@sap.com

ABSTRACT

A new paradigm of software design for sensor networks and ubiquitous computing (ubicom) systems are service-oriented architectures where each node runs just the set of services it needs in its particular role. This paradigm has brought sensor networks conceptually closer to novel business applications, which have started to be modelled as orchestrations of coarse-grained web services that can be described in a standard, executable language like the *business process execution language* (BPEL). Accessing sensor node services directly from such a backend-located business process provides transparent but inefficient real-world integration. This poster describes an approach where BPEL processes are automatically partitioned and transformed such that parts are directly executed within the sensor network using lightweight node services and in-network orchestration with compact, executable orchestration descriptions roaming among minimal execution engines. We expect this to save bandwidth, lower response times, and allow for better utilization of sensor network resources.

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems – *Distributed applications*

General Terms

Algorithms, Documentation, Performance, Design, Economics, Experimentation, Human Factors, Standardization, Languages.

Keywords

BPEL, Wireless Sensor Networks (WSN), Ubiquitous Computing, Smart Items, Embedded Systems, Service-oriented Architectures (SOA), Middleware, System Architecture, Information Systems.

1. INTRODUCTION

1.1 Service-Oriented Architectures

To be commercially viable, ubicom systems¹ have to be integrated into business processes. However, for most of today's

ubicom systems, back-end integration is limited to low-level, proprietary, language-specific APIs provided by the system vendor. The authors of [1] state the need for a middleware for ubicom systems. While such integration middleware is already commercially available for RFID systems [2], no standard solutions exist for ubicom systems. The reason may be the difference in complexity of integration between passive RFID and proactive, collaborating ubicom systems. A particular problem for designing middleware for ubicom systems is choosing an appropriate level of abstraction. The next generation of business applications will use coarse-grained web-services that encapsulate all aspects of a step of a business process like order processing. Such high-level services are composed of multiple, hierarchical layers down to the simplest services that e.g. manipulate database records. The BPEL language is a standard means to combine a set of web services into one integrated web service. We believe that low-level functionality of ubicom systems should also be made available as services that use a uniform scheme for inter- and intra-node communication and are open to remote deployment. Using them to compose higher-level services will enable holistic, service-oriented systems that allow for transparent service composition across all system levels. However, today's ubicom systems are not yet capable of providing web service interfaces, but from a back-end point of view, a middleware-generated proxy may be sufficient. The overhead introduced by this virtualization layer has to be taken into account.

1.2 Service Composition and Mapping

We plan to implement a system where an application programmer can model a process in BPEL, while transparently using existing services of a ubicom system. From outside, the business process will appear to be running on a BPEL engine within the back-end system, while internally, the process will be an interaction of the relocated tasks running in the ubicom system. Peer research succeeded in automatically parallelizing Java applications where the programmer manually assigns the classes of the application to networks sites [4]. Our approach will tackle parallelization at the higher level of service orchestration. To underpin service composition through orchestration, it should be possible to provide the configuration of an ubicom system in a declarative manner, abstracting from the actual deployment of physical devices, their type, availability, etc. This frees an application developer from micro managing a large number of devices. We have implemented a mapping component, which takes as input a specification of the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference '04, Month 1–2, 2004, City, State, Country.
Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

¹ Using the term *ubicom systems*, we assume that sensor networks are part of that class of systems.

desired services together with high-level constraints such as mutual dependencies, location, neighbourhood, etc. and computes a possible deployment of these services (i.e. their low-level implementations), taking into account the capabilities of available sensor nodes. This allows for automatic service deployment and reconfiguration after changes to the physical infrastructure.

1.3 Automatic BPEL Decentralization

Building on the services provided by the ubicomp system arranged by service mapping, an application developer can model a process using BPEL without considering the underlying infrastructure. Executing such a BPEL process centrally would inflict a significant performance penalty on the underlying nodes, though. Therefore, we adopt a distributed execution scheme. A BPEL process consists of tasks like services calls, variable assignments, forks and joins, etc. that each can be executed within the back-end system, or (depending on hardware capabilities) on nodes of the ubicomp system. After analyzing it to identify data dependencies and other runtime specific constraints, a set of allowed partitionings can be created heuristically, each made up of different arrangements of task sets. Each partitioning is assessed based on a suitable cost function. Our cost model considers individual node capabilities of the diverse execution environment. Resources like computation and battery power are limited, demanding sparse message exchange and light-weight tasks. This cost model differs from the one discussed in [3], where decentralized execution of BPEL processes is used in order to minimize communication and maximize throughput when serving concurrent requests. Considering those results, we are confident to find proper heuristics to automatically create partitions suitable for being relocated and executed in a ubicomp system. Additionally, we intent to change the control flow between those tasks executed within the ubicomp system. BPEL's central orchestration, where one node coordinates subsequent tasks and returns the result, will be converted into a partially ordered control flow that utilizes inherent parallelism where the terminal object returns the process result.

2. IMPLEMENTATION

2.1 Scenario and Implementation

The service-oriented system described in this poster is driven by and evaluated in an application scenario will increase safety of handling of hazardous chemicals. It was developed in the EU research project *CoBIs* [5] jointly with BP as a potential corporate end user. Sensor nodes attached to chemical containers must collaboratively ensure continuous compliance with certain storage regulations including: (1) Only a certain amount of a chemical may be stored in one place; (2) joint storage of certain combinations of chemicals is prohibited; (3) some substances must be kept in a special storage area. Any violation of these rules has to be reported and logged. Ideally, the system also gives advice on how to resolve alerting situations. We expect our partitioning algorithm to push most tasks of the business process into the ubicomp system, leading to more timely and accurately detection of rule violations. The system described has been implemented partially and we tested it using the above scenario under laboratory conditions. Chemical containers have been equipped with sensor nodes that autonomously check storage rules (1) and (2). Alerts were generated on breach of rules. The services on the nodes allowed dynamically changing the acceptable amount for rule (1). We also implemented a service-

oriented abstraction of the ubicomp system, which uses UPnP [6] to create one discoverable proxy for each low-level node service running inside the ubicomp system. We plan to migrate this to DPWS², designated successor of UPnP. Middleware components then use these service proxies to create web services for access by back-end applications. Any invocation of these web services is automatically converted and forwarded to the correct device. A component constantly monitors the services in the ubicomp system and maintains a *system model* that can be queried. Within the ubicomp system itself, BPEL execution and strict service-orientation is still in conceptual state but is planned to be implemented. For the real-world test, we decided to use SAP standard software at middleware and back-end application layer as we expect a smoother transition of the project results into a product.

2.2 Real-World Test

The scenario described above will be tested under real-world conditions in a storage area for chemicals at a production site of BP. Tests will include a period where we are on site, manipulating containers, provoking rule violations and measuring detection speed, reliability, stability, and incorrectly reported alerts. Later, we will leave the sensor nodes on site for unattended operation by untrained staff. During this period we will test battery lifetime, long-time system stability, and real-time inventory tracking.

3. SUMMARY AND CONCLUSION

We presented an integrative software architecture that uses services both in ubicomp systems and back-end enterprise systems, allowing seamless integration of both classes of systems. Based on this, we expect to be able to model overarching BPEL processes and partition them automatically to run partly in the back-end and partly in the ubicomp system. The partitioning will ensure that the number and size of messages exchanged among tasks is minimal and take each node's capabilities into account.

4. REFERENCES

- [1] M. Strassner, T. Schoch, *Today's Impact of Ubiquitous Computing on Business*, First International Conference on Pervasive Computing, 2002
- [2] C. Bornhövd, T. Lin, S. Haller, J. Schaper, *Integrating Automatic Data Acquisition with Business Processes Experiences with SAP's Auto-ID Infrastructure*, Proceedings of 30th VLDB Conference, Toronto, Canada
- [3] M. G. Nanda, S. Chandra, and V. Sarkar. *Decentralizing Execution of Composite Web Services*. In Proceedings of OOPSLA'04, Vancouver, Canada, 2004
- [4] N. Liogkas, B. MacIntyre, E. Mynatt, et al., *Automatic Partitioning for Prototyping Ubiquitous Computing Applications*, Volume 3, Issue 3, July-Sept. 2004
- [5] Collaborative Business Items, EU project website, <http://cobis-online.de/>
- [6] UPnP Forum, <http://www.upnp.org>

² DPWS, means *Device Profile for Web Services*, a subset of the WS-* family of standards for constrained devices, <http://specs.xmlsoap.org/ws/2005/05/devprof>