Modelling, Simulation, and Performance Analysis of Business Processes Involving Ubiquitous Systems

Patrik Spieß[1], Dinh Khoa Nguyen[1], Ingo Weber[1], Ivan Markovic[1], and Michael Beigl[2]

[1] SAP Research, CEC Karlsruhe, Germany, http://sap.com/research/
{patrik.spiess,dinh.khoa.nguyen,ingo.weber,ivan.markovic}@sap.com
[2] IBR, Braunschweig, Germany, http://www.ibr.cs.tu-bs.de/dus/

Abstract. A recent trend in Ubiquitous Computing is that embedded software (e.g. in production machines, wired or wireless networked sensors and actuators, or RFID readers) directly offers Web Services. This allows it to directly participate in IT business processes. Our work suggests an approach to assess and compare the performance of such hybrid process models. It critically evaluates BPEL, the standard instrument for creating executable process descriptions, for its support of the dynamic nature of ubiquitous services, e.g. due to device mobility and unreliable wireless communication. [3]

1 Introduction

The near future vision of industrial production (which we see merely as a special case of Ubiquitous Computing) is moving towards an Internet of things, in which smart machinery and smart semi-finished products are integrated into enterprise business processes, to enable more flexibility and adaptability by including some business intelligence already at a very low, technical level.

Following a SOA concept, we argue that the functionalities of those devices should be made available as Web Services, which could be consumed by other devices or composed into higher-level Web Services. As the shop-floor devices are now powerful enough regarding hardware resources such as computing power and memory, they can be designed to host Web Services (SOA-ready devices) and hence can easily be accessed from within business processes.

Given a completely service-oriented device landscape, the next challenges are how to describe business processes orchestrating such hybrid systems using executable process description languages like BPEL [1] and what are appropriate metrics to assess the operational performance of a given process model. In this paper, we propose the modelling and simulation approaches for standard,

[3] The authors would like to thank the European Commission and the partners of the European IST projects SOCRADES (www.socrades.eu) and SUPER (www.ip-super.org), for their support.

executable business process descriptions together with ubiquitous systems. We explain also how to use simulation results for performance analysis of such hybrid business processes.

2 Modelling and Simulation Approach

In order to evaluate the performance of an executable process description in a ubiquitous environment, we suggest explain the benefits an problem when using the BPEL standard, introduce a model of the heterogeneous device landscape and lay out our simulation approach.

2.1 Business Process Description and Execution

For the modelling of an executable business process, we suggest to use BPEL as the de-facto industry standard. A BPEL process description contains references to all services that it interacts with. This limitation is against the nature of a ubiquitous environment, where devices (and consequently, the services hosted on them) can appear and disappear unpredictably at run-time. Our solution to overcome this problem is to extend the BPEL engine, providing the ability to resolve generic partner link descriptions with concrete endpoints stored in the device landscape model at run-time.

Another deficiency of BPEL, mentioned by Wohed et. al. in [5], is the missing support for broadcasting a message to several partners through partner links. Constrained, embedded devices, especially when using a wireless link, would benefit from this functionality. We mitigate this limitation by suggesting and supporting modifications in BPEL and the engine executing it. Our work is based on the engine presented in [3] where Hackman et al. introduced a new BPEL element called *Partner Group*. During run-time, a BPEL engine supporting partner groups features addition and removal of partner links to a partner group, as well as binding and unbinding a partner link dynamically at run-time. For instance, when a message from a previously unknown service is received, its endpoint is mapped (bound) to a currently unbound partner link and this partner link can be added to a partner group. After that, the partner link can be unbound again and is ready for the next incoming messages. Reply messages can be sent to a whole partner group.

2.2 Modelling the Service Landscape

The service landscape model is a data structure, designed in XML, that describes a landscape containing hierarchical layers, groups of nodes, nodes, and services hosted on a node. Each service is specified by a service endpoint and a service type. By discovering the service type of a BPEL partner link, the simulation engine can map this partner link to an existing endpoint in the landscape.

Using simulated services generated from the service landscape definition is useful in the following situations: A process model should be tested with more

devices than physically available, or a process needs to be repeated many times to get a stable average assessment which might take too long. Our simulation approach is straightforward: Replacing the ubiquitous devices with a web application server and letting the server provide the services instead of the devices. If left uncompensated, this would however lead to unrealistically high performance measurements. To compensate, we introduce technical virtual costs for service invocations.

The virtual costs of service invocations that cross the boundary between standard PC-based back-end subsystem and the embedded subsystem should be set considerably higher than communications within these subsystems. The cost for interactions within the embedded subsystem should be set higher than the cost for interactions within the back-end subsystem. For more complex systems, more than two layers may exist, each with its own cost of internal service interactions. For convenience, a default access cost can be assigned at each level of the landscape: on the whole landscape, on a layer, on a group of nodes, and on a node. Each service on a node can have a specific extra cost. The costs defined for a simulation must reflect the resources that are used for each step of the process execution (in the best case known by measurements) and their priority for the individual application. A time critical application would e.g. assign a higher cost to high-latency steps; an application that uses a volume-based 3G subscription would assign high cost to steps that send large messages over the 3G link.

Instead of measuring the performance of the simulated devices, we let the simulation engine sum up the virtual cost of each service invocation. The total cost (and other statistical meta data) of each process instance (from instantiation to termination) under varying environmental conditions (i.e. varying return values of simulated services) is summed up during the simulation. We assume the estimation to be Gaussian distributed, thus we are able to compute a reliable averaged performance indicator by averaging the results over many runs.

2.3 Simulation approach

For simulating the process model on top of virtual devices and services (modelled in a landscape definition document) we need a business process engine that supports the following non-standard features. (1) Dynamic mapping of a partner link at runtime with a service binding stored in the device landscape. (2) Reusing a partner link at runtime, i.e., unbinding a partner link and then rebinding it to a new device. (3) Using the landscape definition to get cost values of cross-layered service invocations in order to calculate the total cost of process instance.

As a basis for our implementation, we chose the Sliver [4] open source BPEL engine, which can parse and execute a BPEL process description and covers most of the BPEL concepts. It stores all partner links in a hash map, and provides getter and setter methods to modify them at runtime. That means partner links can be mapped (or bound), unbound, and reused dynamically at runtime. We extended the engine for many other purposes, especially to link it with the device landscape in order to resolve partner links at run-time and to calculate invocation costs during process execution.

To simulate the connection and disconnection of devices, we modify the device landscape while a process instance is running. New devices can join or leave the device landscape providing new or taking away existing services. At the end of the simulation, when the process ends successfully or in an erroneous state, the calculated total cost is returned to a GUI for users to evaluate the process. The simulation engine can be instructed to repeat this process n times and return the average costs.

3 Conclusion and Future Work

This article analysed the widely adopted standard for executable business process description BPEL for its feasibility to support processes including ubiquitous systems and made suggestions for extensions to that standard. We also introduced an approach for estimating and comparing the performance of the execution of a process description.

This work is part of a larger project that is intended to facilitate the modelling of the behaviour of a hybrid system including services provided by both enterprise applications and ubiquitous systems. Process modelling experts should be enabled to model freely without the need to care for the resource constraints of ubiquitous systems. An automatic optimization step will partition the process and deploy the connected fragments e.g. on small execution engines running on the ubiquitous devices themselves [2]. The distributed version of the process will feature more locality during its execution and will use less system resources although it shows the same behaviour as the original one. The work presented in this paper will be used to evaluate the performance gain of the automatic optimization.

The concepts presented in this paper are based on an ongoing implementation by extending an experimental, open source BPEL engine.

References

1. Web Services Business Process Execution Language Version 2.0 (OASIS Standard), April 2007. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.
2. Patrik Spieß and Stamatis Karnouskos. Maximizing the business value of networked embedded systems through process-level integration into enterprise software. In *Proc. Second Intl. Conf. on Pervasive Computing and Applications*, July 2007.
3. Gregory Hackmann, Christopher Gill, and Gruia-Catalin Roman. Extending BPEL for interoperable pervasive computing. In *Proceedings of the 2007 IEEE International Conference on Pervasive Services*, pages 204–213, 2007.
4. Gregory Hackmann, Mart Haitjema, Christopher D. Gill, and Gruia-Catalin Roman. Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices. In *ICSOC*, pages 503–508, 2006.
5. Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, and Arthur H. M. ter Hofstede. Pattern based analysis of bpel4ws. Technical Report Technical Report FIT-TR-2002-04, Faculty of Information Technology, Queensland University of Technology, December 2002.