# Sentient Processes – Process-based Applications in Pervasive Computing

Stephan Urbanski, Christian Becker, Kurt Rothermel
*Institute of parallel and Distributed Systems, University of Stuttgart, Germany*
*{stephan.urbanski, christian.becker, kurt.rothermel}@informatik.uni-stuttgart.de*

## Abstract

*Users' tasks are typically a sequence of steps. Today, the goal of Pervasive Computing applications – to support users in their tasks by using context information – is only partially met as applications supported by current infrastructures focus on individual steps instead of supporting the coordination of the whole sequence. In this paper, we present an approach for supporting multi-step user tasks by using a process-based application model, called Sentient Processes. Along with the model, the paper introduces the design of an execution engine for such processes.*

## 1. Introduction

Today, we already face the ubiquitous availability of computing devices. Handheld computers, cellular phones and sensors as well as full-scale personal computer or mainframes constitute the pool of resources available when building Pervasive Computing environments. These devices are often interconnected by wireless communication technologies, e.g., 802.11 or 802.15. Context information is provided by various sensors and other information sources like a user's calendar.

Most Pervasive Computing applications support individual steps of a user task by leveraging the resources of their environment and using context information. Similarly, Pervasive Computing infrastructures typically focus on support for a specific task domain or a specific type of step. However, many tasks are comprised of more than one step and involve the interaction of several users. Also, steps of a task may be executed in varying environments as either users are mobile or a step's execution depends on a specific location. Thus, we need a model for defining applications as a sequence of steps that can be executed in different Pervasive Computing environments. The definition of the sequence should include contextual constraints to enable context aware coordination, e.g., suspending a step or task if not allowed to access required resources and restoring it as soon as they become available.

Our goal is to provide a model for applications that enables the application developer to design application support for multi-step tasks that can cross the boundaries of different Pervasive Computing environments. The applications must be context aware and make best use of the execution environment even if the execution crosses boundaries at runtime or if the exact environment is unknown at design time. The application model capturing the tasks must be complemented by an adequate execution engine for runtime support. Such an engine must map the steps of a task on the present environment at runtime and it also needs to support user mobility, e.g., by sticking a certain step to a (mobile) user.

The paper is organized as follows. In the next section, we analyze the requirements for multi-step pervasive applications. Following the discussion of some related work, we introduce our approach for a suitable application model and execution engine. The status of the current implementation and future work is presented before the paper's concluding summary.

## 2. Requirement Analysis

For deriving the requirements consider a maintenance task in a Pervasive Computing environment. An air-conditioning unit breaks down in company C. Company C calls a service company S and requests a mechanic. A mechanic of company S gets the assignment of repairing the air-conditioning unit at company C. The mechanic travels to company C, finds the broken unit and by using a repair manual carries out the necessary repairs. The whole service interaction is billed to company C.

Several steps within this task are already supported by existing computing infrastructure. For example, the mechanic uses navigation software for finding company C. The repair manual for the air conditioning unit is stored in portable computers. The billing process is supported by an IT infrastructure. However,

Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)
0-7695-2520-2/06 $20.00 © 2006 IEEE

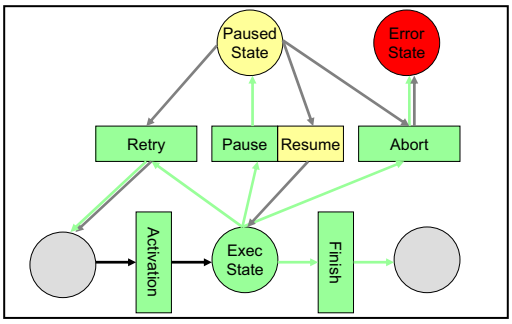


**Figure 1: Specification of a process step**

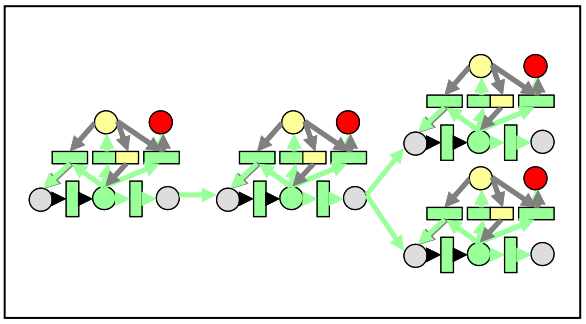

**Figure 2: Several steps with dependencies**

information exchange or coordination of the different steps is not supported, e.g., neither is the navigation software automatically provided with the address of company S nor is the matching repair manual automatically stored on the portable computer.

Hence, we need to support the coordination of all steps in this task, taking into account the multiple, mobile users, loosely coupled steps and unknown environment at the client company. To enable the development of such integrative and context aware applications, a new application model is required that is able fulfill the following requirements:

### 2.1. Dynamic Spatial Binding

The maintenance task of the example crosses the infrastructures of company C, company S and the one of the mechanic's vehicle. A supporting application should offer pervasive support across all these different infrastructures. Since the application developer cannot foresee every possible environment, an application model must abstract from specific environmental references. In our example, the step of repairing can use the display of the mechanic's portable computer. But if there is a larger display available next to the air-conditioning unit, this could be facilitated for more convenient reading.

### 2.2. Context aware Coordination

The execution of a step depends on the context in which it is executed, i.e., the step of repairing an air-conditioning unit is only executed in the presence of a defective unit. We need to distinguish between conditions that must hold to initially start a step and conditions that must hold during the step's execution.

Clearly, many steps of a task are typically started in a temporal ordering, e.g., one step is started when other steps are finished. However, in Pervasive Computing environments additional activation requirements arise. This could be the presence of specific hardware or of a specific user. In our example, the mechanic must be present at the defective unit to start the repair step.

During the step's execution, the environment might change unexpectedly as implied by Pervasive Computing scenarios. Several infrastructures for single-step Pervasive Computing applications tackle this obstacle by automatically reconfiguring the application. For multi-step applications there are two levels on how to react to such changes. First, the currently executing step can be adapted to fit the new environmental situation as done by the aforementioned infrastructures. Second, the step could be suspended or aborted if adaptation fails. Considering the example, if the mechanic leaves the air-conditioning unit before finishing the repairs, the repair step should be suspended until he returns.

## 3. Related Work

Several projects provide infrastructures for Pervasive Computing environments. Some, like Gaia [7] and iROS [5] focus on Smart Environment (SE) scenarios and orchestrate applications in one distinct environment. Others, like BASE [2] or one.world [3] assume spontaneously forming networks. While these projects provide excellent support for their specific environments, they do not support the interaction or migration of applications between smart environments and ad-hoc settings. Aura [8] is an exception and allows migrating complete applications from one SE to another. However, Aura's concept of Task Driven Computing envisions single-step applications only.

Most experience with multi-step, process-based applications is collected in the area of workflow systems. These systems are typically tailored towards well analyzed static business processes [6] and thus do not address dynamism as present in Pervasive Computing.

## 4. Sentient Processes

*Sentient Processes* is our approach for an application architecture model and supporting

```
process repairProcess; […]
step repair {
 activation {
  return (context.isInRange("ACUnit");
 }
 pause {[…]}
 abort {[…]}
 failure {[…]}
 finish {[…]}
 activity {
  invoke("pppc.MainComponent");
 }
}
[…]
{
 start repair;
}
```

**Figure 3: Excerpt from a SPCL script (simplified)**

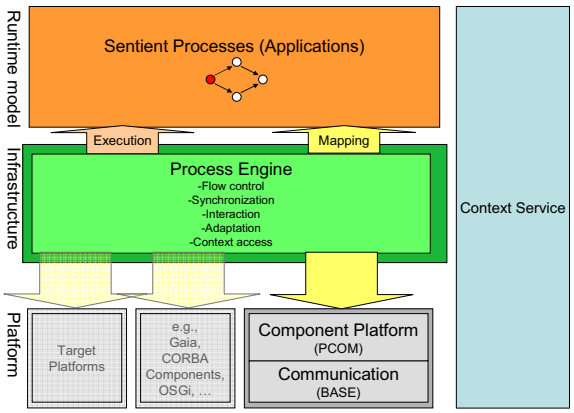


**Figure 4: Architecture of the execution engine**

execution engine that realizes context aware, process-based applications. The model allows application developers to specify processes, consisting of a sequence of individual steps. It enables the developers to include references to context for handling environmental changes and defining activation conditions. Thus the name Sentient Processes.

### 4.1. Application Model

A Sentient Process is formed by Steps. Steps are an abstract specification of an action that is executed in a specific environment. An action resembles a unit of execution of the present environment, e.g., an application in Gaia.

The activation of a step depends on two conditions. One is the completion of previous steps and the other is the environmental situation. Thus – on a modelling abstraction – the step "Repairing the air-conditioning unit" depends on completing "navigating to the customer" and the context "being at the defective unit".

While a step is executed, the environment might change unexpectedly. If the step cannot be adapted to the changed environment, Sentient Processes allow the developer to specify further actions. These include suspending the step, starting over the step and raising an exception that results in aborting the whole process. "Repairing the air-conditioning unit" should for example be suspended if the mechanic leaves the room but has not yet finished the repairs.

### 4.2. Execution Model

During its execution a Sentient Process is controlled by a Process Engine. The engine controls synchronization between the steps of a process. Using a context service that is typically provided by the environment, it evaluates contextual conditions specified in the process description. Furthermore, the Process Engine maps the abstract steps on specific actions at runtime and monitors their execution using an underlying platform. To perform this task, the Process Engine uses information about available resources that are present in the environment. To enable the crossing of infrastructure boundaries, Sentient Processes can migrate between different Process Engines. This migration also allows the development of Process Engines that are optimized for a specific environments which in turn, enables efficient support of static scenarios, i.e., Smart Environments, and dynamic scenarios based on spontaneously networked devices.

### 4.3. Developing Sentient Processes

For developing Sentient Processes we provide a specification language called Sentient Process Command Language SPCL. Figure 3 shows an excerpt from an SPCL script. The structure of a process description in SPCL resembles a petri-net. Places represent the internal dependencies of a process, e.g., if step A has to be finished before step B. Transitions represent the steps. The developer specifies the process by describing all the included steps and internal dependencies. The external dependencies, i.e., the context dependencies are defined by five blocks containing the finish, activation, suspend, retry and abort condition. An API provides access to the execution engine helping to invoke actions and define context conditions. A development environment featuring a visual editor for Sentient Processes is currently under development.

## 5. Implementation Status

Sentient Processes add an additional layer of abstraction to other distribution infrastructures. The integration of different infrastructures implies facilitating these infrastructures as much as possible.

Figure 4 shows the architecture of the Sentient Process engine and its interfaces to and interactions with external infrastructures.

### 5.1. Process Engine

The Process Engine is essentially a container for processes. It controls the life-cycle and implements an interpreter for SPCL. The engine interfaces and interacts with a context service and an underlying component platform. Sentient Processes feature a distributed execution model. Hence communication between Process Engines is required. Sentient Processes can be serialized including their context and be transferred to other Process Engines. The description of protocols for synchronization of parallel executed steps and the migration of Sentient Processes are beyond the scope of this paper.

The interaction with the component platform includes the mapping of steps to actions and controlling their execution. Hence, the implementation needs to be component platform specific.

### 5.2. Component Platform Integration

As a proof of concept, our current implementation of Sentient Processes operates on top of PCOM [1], a Pervasive Computing component platform. PCOM provides automated application configuration. Thus, a step within a process is mapped on one PCOM application and resource configuration is done by the PCOM container. If a step is processed by a Process Engine, the engine invokes a PCOM application and monitors their execution. A simple API allows the PCOM application to access contextual information of the process. However, no special adaptation of PCOM applications is used by Sentient Processes in order to remain general enough to provide mapping to other platforms.

## 6. Future Work

At the present time, we are integrating a context service based on a generic context model and we are developing an extension of SPCL for describing events in that context model. This will enable the automated, context aware life-cycle management of Sentient Processes. One of the next milestones will be the design of a graphical front-end for developers to visually design Sentient Processes. Furthermore, we plan to investigate the migration of Sentient Processes using adequate migration mechanisms and policies.

## 7. Conclusion

In this paper, we have analyzed the requirements for multi-step applications in Pervasive Computing environments. Sentient Processes are our approach to satisfy these requirements. The Sentient Process application model enables developers to design applications as processes represented as sequences of steps with context dependencies that allow the specification of context aware execution conditions. Apart from starting conditions, context awareness is also maintained during the execution of a step as conditions for finishing, pausing and aborting a step.

Our current proof-of-concept implementation indicates that the proposed model for multi-step applications is feasible in Pervasive Computing environments. In the near future, we will validate this impression by adding a context service and implementing exemplary applications. Furthermore, we will provide application development support by implementing a visual editor for Sentient Processes.

## 8. References


[1] C. Becker, M. Handte, G. Schiele, "PCOM - A Component System for Pervasive Computing" *International Conference on Pervasive Computing and Communications (PERCOM) 2004*, Orlando, USA, 2004

[2] C. Becker, G. Schiele "BASE: A Minimal yet Extensible Platform for Pervasive Computing" *In Proceedings of the International Conference on "Tales of the Disappearing Computer"*, Santorin/ Greece, 2003

[3] R. Grimm, J. Davis, E. Lemar, A. MacBeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, and D. Wetherall, "System support for pervasive applications" *ACM Transactions on Computer Systems*, 22(4):421-486, 2004

[4] M. Handte, G. Schiele, S. Urbanski, C. Becker, "Adaptation Support for Stateful Components in PCOM" *Workshop on Software Architectures for Self-Organization: Beyond Ad-Hoc Networking at Pervasive*, Germany, 2005

[5] B. Johanson, A. Fox, T. Winograd, "The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms" *IEEE Pervasive Computing Special Issue on Overviews of Real-World Ubiquitous Computing Environments*, April-June 2002

[6] F. Leymann, D. Roller, "Production Workflow" *Prentice Hall*, September 1999

[7] M. Román, C.K. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces." *IEEE Pervasive Computing*, pp. 74-83, Oct-Dec 2002.

[8] J.P. Sousa, D. Garlan, "Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments" *Proc. of the 3rd Working IEEE/IFIP Conference on Software Architecture*, Kluwer Academic Publishers, August 25-31, 2002