ABSTRACT
We describe the use of proximity relations for enhancing a service platform regarding the support of context-aware mobile business-to-business and business-to-employee processes. Such relations constitute a special type of spatio-temporal context which is assumed to be produced by a configurable set of different sensing systems. We propose a networked service (context engine) to which the context data is delivered for further processing and for making the data accessible to other services and applications of the software platform through various interaction modes. As will be seen, a mobile business processes can be optimized in various ways (e.g., acceleration or reduction of occurrences of errors during the process execution) through this. The service-oriented wrapping results in the key advantage that the underlying context-sensing (i.e., tracking) technology can be easily exchanged without affecting the rest of the system or – even worse – the process. As an application of the concepts, we describe a setup in which the proximity of an assembly (or a respective part of it) to be maintained and the mobile worker doing the maintenance is used to configure the services in the service platform.

Categories and Subject Descriptors
D.2.1 [Software Engineering]: Software Architectures

General Terms
Design, Human Factors.

Keywords
Context-awareness, adaptive services, mobile maintenance.

1. INTRODUCTION
The support of mobility in business processes involving the lifecycle (manufacturing, maintenance, repair) of high-complexity goods like for instance airplanes, ships, plants and sophisticated larger appliances is becoming increasingly important. From a technical point of view, service-orientation [3] constitutes an elegant approach for wrapping data sets that belong together in conjunction with the interfaces and functions working on this data which comprises the following two major benefits:

1. starting from one application sector (e.g., aeronautics industry) the services contained in the platform can be used to quite rapidly develop applications for other domains as well;
2. within one application domain a variety of types of mobile processes (repair or quality control instead of maintenance) can be supported.

In [2] the requirements and the conceptual design of a platform for the case of supporting mobile maintenance in the aeronautics industry were explored. Concerning the typical types of cooperation services depicted in Figure 1, a nomadic worker has to interact with different entities which can be either constituted by technical systems as well as by co-workers (back-office experts giving technical support while identifying or fixing a problem). Though this is of no major importance, we would like to mention that the worker is assumed to be nomadic (as opposed to seamless mobility) in the sense that temporal disconnection from basic networking and communication services may occur.

Within this paper, we describe how to use proximity information as a special type of inferred contextual information within the service platform for the purpose of accelerating the process execution and improving the process execution in terms of a reduced error rate. This is done by implementing the collecting...
and reasoning on the sensed data as a networked service which can be easily customized to different purposes of application and accessed through according interfaces. Herein, proximity relationship information as a special type of inferred contextual information can be used in various ways. We consider the tracking and guiding of activities depending on the sensed context as a very intuitive and straightforward application of it. General concepts (e.g., SOCAM [5]) have been introduced but have not been adapted to the specifics of the targeted application area.

We will start off by introducing the service platform and then sketch the enhancement of the platform through the use of contextual information.

2. NOMADIC WORKER SCENARIO

Though the use of the concepts is quite generic, we would like to make clear the type of mobile services and thus the application domain we are envisaging.

We target the support of nomadic workers doing maintenance of aircraft. Here the nomadic workers are equipped with a smart mobile device (PDA or tablet PC with ultra-light headset and camera as peripherals) which enable them to move freely at site which is huge due to the dimensions of the maintenance object. The workers are in charge of efficiently performing maintenance operations on airplanes. For this, a worker needs to have continuous access to documentation residing in a repository; it might for instance happen that the worker has to maintain an engine of an aircraft he has never handled before. Since each aircraft has its specifics, the maintenance workflow may differ even for instances of the same type. In the traditional setup the mobile worker has to go to the office and retrieve the right workflow from a huge amount of paper. When looking at the huge amount of different procedures caused by the high number of different object types (e.g., aircraft), the identification of the appropriate procedure is laborious. In the particular case described in [1], a significant amount of the overall working time had to be spent on information retrieval. In order to optimize the search process and even the workflow itself, maintenance procedures are provided in digital format. Via his mobile device the worker can access all the necessary information concerning the workflow to be executed.

The information might be contained either directly on a tag attached to the gadgetry or via linking to the according information residing on a background system through a part ID. In order to speed up and ease the entire interaction with the technical systems, the data is instantly formatted to the capabilities of the client device on a server and can be accessed through a wireless network. We refer to this part of the scenario as Context-aware Procedure Loading. We aim at the reduction of the time (up to 10% of the overall time budgeted to the process) for retrieving maintenance documentation and setting up the maintenance process.

3. MOBILE SERVICES PLATFORM

3.1 Mobile Services Taxonomy

For the services platform, we do not interpret service-orientation in a narrow (i.e., completely focused on web-services which represent a specific implementation option) but in a bare and rather stylish sense. To our mind the following properties of the service-oriented paradigm [4] appear to be most relevant for the design of a platform supporting mobile maintenance procedures:

- a service wraps data and functions;
- access to a service is only feasible through well-defined interfaces;
- services are intended to be modular, i.e., one (higher-level) may orchestrate the access to other (basic) services.

This interpretation quite much aligns to the REST principles introduced by Fielding [7] since specific services handling the user interaction (speech input) require rather specific interfaces.

For the plethora of available mobile services in the enterprise domain, a distinction is usually drawn between business-to-business (B2B) and business-to-employee (B2E) services in order to indicate the parties that interact. Whereas several service platforms for the aggregation of B2C (business-to-consumer) applications based on rather simple services exist, generic service-oriented platforms particularly for the B2E area are rather seldom. B2E services have proven to leverage mobile businesses in several cases through assisting the mobile employee to carry out operations in the field. Such services support cooperation by including all types of back-office resources (agents, information systems, applications etc.) thus saving time and effort, and improving the quality of the outcome of the work, e.g. for carrying out maintenance processes. Such services constitute the main ingredients for the services platform addressing the above scenario.

3.2 Application Domain-specific Services

In order to make the access to the maintenance procedures convenient to the mobile worker, we distinguish the following major system elements:

- Adaptation Manager: the Adaptation Manager is the component responsible for generating output adequate for client devices. For this purpose, it has access to device descriptions characterizing the mobile devices and user preferences. It adapts the procedure to different modalities
- Annotation Accessor: the Annotation Accessor is responsible for storing and retrieving annotation information by interacting with the user annotations database. The annotations are linked to detailed procedural steps which can contain personal data issued by the mobile worker.
- Procedure Manager: the Procedure Manager is the component responsible for loading and preparing the maintenance procedure exactly appertained to the maintained assembly.
- Procedure Repository: the Procedure Repository comprises all available procedures for all the assemblies of the aircraft to be maintained.

3.3 Platform Details

The Maintenance Server is the main instance inside the maintenance solution and arranges the adaptation of the procedures depending on the capabilities of the Mobile Device.

- Adaptation Manager: the Adaptation Manager is the
interface between the client- and the server side. In the current setup, the Adaptation Manager is developed as a Java Servlet, thus making it independent of the underlying operating system and ensuring scalability through dispatching requests for content in a client specific modality to the according adaptors. Currently visual (HTML) and acoustic output (audio) are supported. During the adaptation, the visual content is paginated according to the capabilities (screen size), images are resized and style sheets are modified if necessary to avoid scrollbars if possible.

- **Annotation Accessor**: the Annotation Accessor is developed as a Java application, which implements an object-relational mapping (OORM) of the annotations to a MySQL database. Via the accessor annotations can be inserted or deleted. An annotation database entry is based on the jobURI, which denotes the particular step inside the maintenance procedure, the annotation ID and the annotation data.

The Procedure Server provides the maintenance procedures for the special assemblies.

- **Procedure Manager**: the Procedure Manager is designed as a Java Servlet. In the current configuration it provides a list of all available maintenance procedures to the Adaptation Manager.

- **Procedure Repository**: the Procedure Repository is a standard Java application which loads/stores the maintenance procedures from/to the file system.

We will now describe the use of proximity relations within the services platform.

## 4. CONTEXT-ENHANCED SERVICES

### 4.1 Proximity Relations

We model technical units and mobile workers both as objects. Then we can assume the proximity \( P \) of objects \( O \) to be a reflexive, symmetric, non-transitive relation between objects, i.e. \( P \subseteq O \times O \). It is

- **reflexive**, because each object \( o \in O \) is in its own proximity \((o, o) \in P\)
- **symmetric**, because if an object \( o_2 \) is close to \( o_1 \), \((o_2, o_1) \in P \), then obviously also the opposite applies \((o_1, o_2) \in O \)
- **usually not transitive**, because if \((o_2, o_1) \in P \) and \((o_1, o_3) \in P \) then \((o_2, o_3) \notin P \) might occur (suppose a circle around \( o_1 \) containing all objects in its proximity where \( o_2 \) and \( o_3 \) are on the diameter exactly on the opposite sites).

The relation is a result of evaluating some lower level context properties of the objects being put into a relation. Though there are many interpretations and types of context [8], the spatio-temporal context (i.e., the location in terms of e.g., Gauss-Kr"{u}ger coordinates, at a certain point of time) of an object is most relevant here. Thus, one needs to determine

- the **location** (e.g., in terms of coordinates) of the objects being part of the relation
- potentially the **fidelity / confidence** of the location data
- and the **time** of determining the above information

in order to construct the proximity relation. It is important to notice that in principal these attributes are always tagged to the relation and that the derivation of a proximity relation is a two-stage process. This is exactly modelled in the context-awareness service, the **Context Engine** which in contrast to former approaches [4] resolves proximity through a rule-based approach.

## 4.2 Design Context Engine

### 4.2.1 Sensor Input

In order to evaluate the proximity relation, sensor data representing location contexts have to be continuously processed. In order to keep the entire system quite flexible concerning the type of tracking system used, we have to take into account that the particular sensing system delivers its location data in proprietary location syntax. For further processing, the data will be transmitted via a **Context Provider Adapter** (CPA).

The integration and enlargement of an additional or alternative location system can be done by deploying a new Context Provider Adapter. This CPA will be similar to other CPAs but has to provide a conversion between the system dependent proprietary context data and the BCO, which will be described in the subsequent section.

### 4.2.2 Data Transformation and Management

The sensor systems provide the context information via a CPA in an appropriate structure (BCO) shown in Fig. 2. The Branch Context Object (BCO) comprises at least one context structure describing for instance a symbolic mapping or Gauss-Kr"{u}ger coordinates to specify a location context. The Context Engine expands the received BCO by a timestamp and stores it as a Root Context Object (RCO) in the Context Data Cache.

![Figure 2. Basic schema for handling location data](image)

Based on the RCOs of two objects, the proximity relation can be computed. The services or applications which apply context information (e.g. Procedure Manager) connect to the configuration mechanism of the Context Engine and insert their
Required Context Syntax (RCS) which contains the relationship information in the requested format. The translation mechanism for transforming from the RCO into the RCS is implemented by featuring a rule engine.

4.2.3 Context Engine
The Context Engine (see Figure 3) is designed as a web service. The communication to the Procedure Server and mobile device running the mobile worker application is handled via a REST interface (i.e., http requests) where the messages form the sensor systems are transmitted via SOAP, i.e. the CPA of the location sensing system then connects to the Context Engine and delivers its location information as BCOs.

Figure 3. Context transformation via Context Engine
There are different technical options for implementing the above transformations. In order to be flexible concerning the reasoning especially with respect to the evaluation of the proximity relations, we feature a rule engine which permits to:

- organise rules depending on the involved data sets, their respective purposes and priorities; for instance with respect to the featuring of context-awareness in a scenario as the one described here, rules for the transformation of location contexts (i.e., contexts represented by an RCO have to be adapted to the application specific RLS) and rules for triggering changes as a consequence of executing a rule (when a user enters the vicinity of a maintenance object, the procedure should pop up) have to be distinguished;
- manage rules, i.e., rules can be added to and removed from the rule database; moreover, it should be possible to modify rules at runtime;
- enter rules in different formats (external representation in XML, spreadsheets) and to interface with state-of-the-art programming languages; within the current implementation, the Java programming language has a prominent place;
- administer the access to rules concerning authentication and authorization, either for changing the rule database or for applying the rules contained therein.

Of course, it would be preferable to utilise a widely accepted external representation of rules (e.g., based on XML like it is envisaged by the RuleML initiative [9]), such approaches are not quite mature at the moment and thus we use the proprietary format coming with the rule engine Drools [10] which is featured in current implementation of the system.

4.2.4 Overall System Integration
Fig. 4 shows the placement of the Context Engine within the mobile worker environment. For the Context-aware Procedure Loading scenario the Procedure Manager retrieves the location of the assembly (proximity data) to be maintained from the Context Engine and provides the procedure matching the location to the calling process by loading it from the procedure repository.

4.2.5 Extensions
There is one straight-forward application of the concepts described here; i.e., the context engine may serve as a sensor system to another connected engine. For this, the engine has to feature a CPA and transmit the BCOs through it to the other engine.

4.3 Further Scenarios
Aside from the scenario described above, further applications can be easily served, like for instance:

- depending on the positioning of a mobile worker relative to the services available to him, an appropriate service environment (e.g., consisting of services facilitating the communication with back-office experts) can be configured;
• depending on the proximity of nomadic worker with respect to a maintenance object, a request for support concerning an instant problem can be routed to a qualified back-office agent;

• similarly, for a group of mobile workers maintenance jobs can be dispatched depending on the proximity of the group members relative to objects to be maintained;

These are just a few examples each bearing a certain potential for further optimizing the execution of the maintenance process.

5. IMPLEMENTATION ISSUES
The Mobile Device is a standard PDA running a Windows Mobile 2005 operating system equipped with a WLAN interface (see Fig. 5). Although one might think of more advanced wearable devices (e.g., ultra-lightweight headsets, smart closing) the PDA turned out to be the mobile device preferred by a representative group of mobile workers according to [1]. The necessary system components are as follows:

• Web Browser: the maintenance procedure runs on top of a standard web browser (e.g. Internet Explorer) on a PDA. The mobile worker can select one out of the list of maintenance procedures, delivered by the Procedure Manager and can navigate through it by using the linked procedure steps (e.g. next step, previous step ...).

• Mobile Worker Application: the Mobile Worker Application is developed as a .NET application especially for a PDA running Windows Mobile 2005. Besides the same functionalities like the web browser solution, the user can navigate by using voice commands and in turn can listen to the procedure steps indicated by the application. The voice and speech recognition is handled by the mobile worker application. The navigation inside a maintenance procedure can be controlled by a standard headset.

6. CONCLUSIONS
We have presented the integration of context-awareness concepts into a service platform for the support of collaborative mobile maintenance processes. The proposed context engine permits to consistently evaluate proximity information gained through identifying tagged assembly parts and tracking the user in order to dynamically configure the maintenance procedure. Especially for maintenance processes carried out in a non-stationary mode the acceleration of the according processes is significant. Moreover, the prevention of errors can be achieved by using such spatio-temporal context information.

Concerning future work, we believe that the system should not be limited to the consideration of spatio-temporal context elements only. Other context elements relevant to the targeted maintenance application domain address the status of the worker in terms of activity and with respect to the due workflow, the physical working conditions (e.g., the background noise which might trigger the modality selection) and properties being located in the user domain (user preferences and profiles). The context engine then has to feature a more complex rule set.

7. ACKNOWLEDGEMENTS
This work is partially supported by the IST FP6 PLASTIC project (Providing Lightweight Services Technology for pervasive Information and Communication) under contract no. EU FP6 26955.

8. REFERENCES


