A System Architecture for Knowledge Exchange in the Industrial Domain

Fabian Quint¹, Jörn Kreutel², Frieder Loch¹, Magnus Volkmer³, Peter Pollmanns⁴

Innovative Factory Systems, German Research Center for Artificial Intelligence (DFKI)¹
Department of Informatics and Media, Beuth University of Applied Sciences Berlin²
Business Development & Innovation, Infoman AG³
Software and Project Development, SemanticEdge GmbH⁴

Abstract

Exchange and documentation of knowledge is vital in the industrial environment. The access to knowledge of colleagues can be essential for fulfilling several tasks. Just as well, workers can make experiences that are valuable for others. Within the project AmbiWise a collaboration system including adaptive and multimodal user interfaces for mobile interaction is developed to make knowledge easier accessible in companies. This paper presents the conception and implementation of the developed system architecture.

1 Introduction and Motivation

In daily work life several situations arise, where colleagues require knowledge of other colleagues to successfully fulfill their tasks, or made experiences by themselves informative and valuable for the colleagues. Especially, experts’ knowledge of complex technical tasks is a valuable resource of today’s companies and the documentation and exchange of this knowledge needs to be possible in an easy, seamless and non-disruptive manner. Within the project AmbiWise, new assistance and collaboration systems are developed which allow a culture of exchange and participation in companies. The technological basis for an improved accessibility to knowledge resources is formed by intuitive, adaptive and multimodal user interfaces (UI) which are developed for a mobile and context-sensitive interaction. This paper presents the concept of such a system, the requirements towards it and the resulting system architecture.

In the production domain the knowledge of standardized, planned processes is formally documented. Nevertheless the groundbreaking, situational knowledge is usually situated in the heads of the employees (Thom & Sollberger 2008). Especially, implicit knowledge, as described by (Polanyi 1985), which is gained by longtime employment is hard to verbalize...
and is hence often missed capturing by traditional tools. For a general exchange of knowledge between employees, collaborative platforms based on Web 2.0 technologies have proved to be of value in several application scenarios (Stocker & Tochtermann 2011). Nevertheless, they have only shown limited success externalizing implicit knowledge, although this is from special interest for several scenarios (Probst et al. 2010). Allowing this kind of knowledge exchange in a production environment requires mobile, context-sensitive and bidirectional access to knowledge to provide on spot assistance to the worker. By using appropriate assistance modalities, production workers can have immediate access to companies’ and colleagues’ knowledge and transfer knowledge to the system by themselves.

2 Application Scenarios

A comprehensive requirement analysis is necessary as the basis to design an appropriate system. This analysis should involve all substantial stakeholders and their real working environment. The conducted analysis is described by (Weber et al. 2014). Two main scenarios where identified where the system should be applied to. These scenarios given by the industrial end users Daimler and Schaeffler, are described in section 3.1 and 3.2.

2.1 Assembly Processes

For Daimler the AmbiWise system is used to manage standardized procedures in assembly processes. The first task is to support audits of production stations based on these standards. These audits of production stations are conducted in a mobile way. At the beginning, the relevant documents are retrieved based on context information or manual selection. Then, the audit is conducted and deviations from the standard are identified, discussed and stored in the system. The second task is to provide mobile access to the latest production standards for qualification. This allows the personnel to examine standards and check if their knowledge is up to date or if they have to improve their knowledge. Learn instructors who manage individual qualification can use these documents to conduct training on spot. The third task is to adapt production standards. This happens, for example, when improvements were identified during an audit. These adaptations (e.g., editing text, updating pictures or adding videos) are created and transferred to the system on spot. Thereby, the editorial effort is minimized compared to handling these tasks on stationary computers.

2.2 Maintenance

For Schaeffler the AmbiWise system is used to document maintenance processes in a global production environment. The exchange and conservation of knowledge of maintenance standards is facilitated over different production sites. Primarily, experts’ knowledge of complex maintenance tasks should be gathered to establish an extensive knowledge repository. Video recording is applied to document this knowledge. Maintenance personnel are thus able to document and explain procedures at the point of action with the least possible distraction from the actual work. To achieve this, the documented task is
automatically sent to the knowledge repository and processed there. Furthermore, the externalization of implicit knowledge is enabled, since it directly affects the adaption of the standard.

3 Concept

An overall concept was developed at the beginning of the project. The concept abstracts from the concrete scenarios and the results of the requirements analysis. The concept, as shown in figure 1, illustrates how an UI which is adapted to the available context (e.g. user, location, role, etc.) is used to retrieve or store knowledge from or to a knowledge repository. This repository also integrates existing sources, not yet enabled for collaboration and exchange in a production environment. The UI offers different modalities like speech, multi-touch, videos and pictures, to represent the knowledge in an appropriate form.

Information about the context is essential to access and store knowledge effectively. Several technologies are available which need to be chosen and adapted towards the respective scenarios. The variety of available sensors of mobile devices provides many possibilities to detect the context (e.g. camera for recording videos, scanning barcodes, microphone for speech recognition and recording). These technical capabilities allow both knowledge and context to be gathered immediately on spot. The presented concept supports different hardware platforms comprising for example tablets and glasses with different operating systems. Besides choosing the appropriate hardware, the software components which enable the different in and output modalities need to be designed according to the requirements as well. These software components are encapsulated, so they can be aggregated for different variants. Thereby, both scenarios can be covered with one system design although they have different requirements and task perspectives. The requirement and design of these encapsulated services is described in the following chapter.
4 System Design

Based on the overall concept a system design is developed hereafter. The considered requirements are discussed in section 4.1 to be met by the system architecture shown in section 4.2.

4.1 Requirements

The system design comprises a functional decomposition to address the requirements of the industry stakeholders and the projects’ objective of creating a reusable and extensible system architecture. The system functionality is decomposed into particular, yet abstract functions that form the core of the system and address the requirements of the application scenarios. The architecture uses primary functions which are complemented by secondary functions that address cross-cutting concerns relevant for all primary functions. As shown in figure 2, primary functions include read and write access to the particular content, content search, and functions that allow communication between users (e.g. providing a workflow to all stakeholders that participate in the process of creating and releasing content). Secondary functions include for example context detection and speech recognition. A common foundation to all functions is provided by a shared model of the respective application scenario that is based on generic concepts of the production domain. It serves as the basis for content creation and content access, for context detection and for the recognition and interpretation of spoken input.

![Diagram of system functionality](image)

Figure 2: Decomposition of system functionality into primary services for content and communication (bottom) and services that address cross-cutting concerns (left)

Another requirement is to address a wide range of target platforms for running the mobile UI. Platforms range from legacy Windows 7 tablet computers with stylus input to mobile devices based on Windows 8. Hence, development of a single native mobile application for the most recent Windows platform version would neither meet the requirement to access the AmbiWise system from less recent devices, nor would it allow the seamless portability of the UI to alternative mobile platforms like Android or iOS.
4.2 Service Architecture

The software architecture adheres to the proposed decomposition of system functionality. Each functional aspect described above is modeled as a service for which a particular service interface can be identified. Service interfaces hide the actual implementation of a service from its users and allow to realize the overall system based on smaller functional units. These may be operated in a distributed way over various locations, rather than being assembled to a single monolithic runtime system. Interface design adheres to the principles of RESTful service interfaces (Fielding 2000). As for the overall service architecture, recently the paradigm of microservices (Fowler & Lewis 2014) has been positioned as an evolution of the last decade’s paradigm of Services Oriented Architecture (SOA) (Arsanjani 2004) for realizing distributed systems. With regard to this distinction, the system employs a vertical decomposition of system functionality as proposed by the microservices approach, differentiating services that make up the primary system functionality. However, while aiming at minimizing dependencies between services, it assumes that an integration service acts as an orchestration layer that provides the functionality of the specified service interfaces to the layer of the UI. The integration service mediates between the UI and an underlying layer of more basic services that implement the actual backend functionality, e.g. using a particular enterprise content management system like Microsoft Sharepoint (Microsoft). This way, the UI components stay agnostic with respect to the particular backend systems which are employed in a particular deployment of the AmbiWise solution. Thus the project’s idea of providing a reusable system architecture that can be integrated flexibly into various IT infrastructures is addressed.

All service interactions are mediated by a component that can be seen as a lightweight realization of what was foreseen as a “Service Broker” in SOA. It provides bindings of descriptions of the services to be called to particular service endpoints. Service description comprises, among others, a unique identifier of the abstract service interface and optional additional information regarding, e.g. the tenant for which the service is called. Service bindings contain, most importantly, the URL of the service endpoint and may include information regarding whether service calls require authentication or not. In the AmbiWise project the two industry partners Daimler and Schaeffler are modelled as tenants, where each tenant provides individual endpoints for the primary system functions to a centralized integration service. Once a service binding is available for a given service description, the actual service interaction will then be carried out directly between the service consumer and the given endpoint as a service provider, rather than using a centralized, yet usually heavy weight, enterprise service bus (ESB, see (Chappell 2004)) for communication.\footnote{In a certain way, however, the mediation role of the integration service may be seen as functionally comparable to an ESB.}

Given the idea of a distributed system architecture and the fact that content provided by the AmbiWise industry partners must be treated as confidential, one crucial issue for the system design is how service interaction can be secured. This should be realized in a generic way.
without each service having to implement its own mechanism for authenticating and authorizing users. Therefore, AmbiWise draws upon the OAuth 2.0 standard (Hardt 2012) that is widely used in consumer applications for “outsourcing” the authentication task to social network authentication services, like the ones of Facebook, Twitter or Google. In AmbiWise, authentication and authorization is realized by the cloud-based OAuth service of Microsoft Azure (Microsoft). There, access rights for both end users and services that may act in the name of an end user are managed. At runtime, each service call requires obtaining an access token from the authentication service. This is either done on the basis of the token that was received on end user authentication or using a token that was issued to a previous service call and that encodes the information about the actual end user. Access to an actual backend system involves a chain of such delegated authentication steps, starting at the UI, and allows the backend to finally verify whether the end user actually has the permission to access some content item or not.

![Figure 3: Overview of the AmbiWise system architecture using an integration service as mediator between UI and backend services and a 3rd party service for authentication and authorization](image)

The decomposition of system functionality supports various deployment scenarios, ranging from on-premise solutions to mere cloud-based solutions and including hybrid variants in between. It does not only prove convenient for the organization of development activities, but also allows to address infrastructure circumstances and requirements of a wide range of potential application scenarios beyond the particular scope of the project.

### 4.3 User Interface Architecture

The realization of the UI has to take into account that various devices may be employed. For that reason the architecture foresees a two-layered approach where the core functionality of the UI is realized as a “fat client” web application that uses the system’s integration service for accessing backend functions. This application can be run on any recent browser including the versions of Microsoft Internet Explorer that are available for the legacy Windows 7 platform. It supports realizing all tasks of the different scenarios the system is designed for. For addressing the advanced UI functions, including context detection and speech...
recognition and interpretation, this web application is wrapped into a native application for the Windows 8 platform, which supports the respective UI functionality. The wrapper interoperates with the web application using the provided interfaces of the native API for invoking JavaScript functions. For example, the generic mechanism for obtaining an access token for a service call is implemented by the web application and may be called by the wrapper, which will use the obtained token for calling the respective service. The implementation of the wrapper on the basis of a single native API allows to realize advanced UI features without being compromised by API restrictions imposed by cross platform frameworks like Cordova (Apache 2015) or Xamarin (Xamarin 2015). Thinking ahead, porting the UI to other platforms will allow to reuse the existing web application and will at the most require an implementation of the native wrapper for the target platform.

5 Summary & Outlook

This paper presented the architecture of a collaboration system targeting knowledge exchange and participation for the industrial domain. It gave an overview on the application scenarios which the system should facilitate. Based on a general concept, abstracting from the concrete scenarios, the considered requirements were summarized. The further derivation of the developed system architecture was shown. The motivation and benefits of the used service oriented architecture were discussed. The encapsulation of functionalities to services and their interaction was illustrated. Finally, the architecture of the UI designed to run on several operating systems and devices was introduced.

The current status of the system gave a proof of concept and will in near future be evaluated under real conditions for the two application scenarios. This evaluation ends the first phase of the AmbiWise project. In the second phase the focus is set on further UI modalities, which will enhance the system functionality and usability.

6 Bibliography


**Acknowledgement**
The research described in this paper was conducted within the project AmbiWise funded by the German Ministry of Education and Research under grant number V4ISS035.

**Contact Information**

**Fabian Quint**
fabian.quint@dfki.de  
German Research Center for Artificial Intelligence  
Trippstader Straße 122  
67663 Kaiserslautern

**Joern Kreutel**  
joern.kreutel@beuth-hochschule.de  
Beuth University of Applied Sciences  
Luxemburger Straße 10  
13353 Berlin

**Frieder Loch**  
frieder.loch@dfki.de  
German Research Center for Artificial Intelligence  
Trippstader Straße 122  
67663 Kaiserslautern

**Magnus Volkmer**  
magnus.volkmer@infoman.de  
Infoman AG  
Meitnerstraße 10  
70563 Stuttgart

**Peter Pollmanns**  
peter.pollmanns@semanticedge.de  
SemanticEdge GmbH  
Kaiserin-Augusta-Allee 10-11  
10553 Berlin