Enabling Users of Enterprise Systems to Mashup Resources and Develop Widgets

Michael Spahn, Julian Dax, Fahri Yetim, Volkmar Pipek

in: V. Wulf; V. Pipek; D. Randall, M. Rohde; K. Schmidt; G. Stevens (eds): Socio Informatics – A Practice-based Perspective on the Design and Use of IT Artefacts, Oxford University Press, Oxford 2018, pp. 421-444

Abstract. Companies are operating in a dynamic environment and need continuously to adapt their information systems to changing business processes and associated information needs. Viewed from a micro-perspective, business users are managing and executing business processes on a daily basis, but are not able to adapt used software to their individual needs and working practices. In this chapter, we present the development and evaluation of a prototypic environment which enables users to create enterprise widgets tailored to their personal information needs without the need of programming knowledge. It allows the mashing up of enterprise resources using a lightweight visual design technique. The evaluation of the prototype in business contexts indicates its usefulness and usability.

1. Introduction

Many companies use enterprise software systems like Enterprise Resource Planning (ERP) systems to support and facilitate their business. Companies operating in dynamic environments need to continuously adapt their information systems to changing business processes and practices as well as their associated information needs (Henderson & Kyng 1989; MacKay 1990; Wulf & Rohde 1995). Small and medium-sized enterprises (SMEs) in particular often suffer from their inability to adapt enterprise software to their needs, due to a lack of resources and expertise (Roth & Scheidl, 2006; Dörner et al 2011., Yetim et al., 2010). As a consequence, they are often forced to adapt their work processes and practices to the possibilities offered by the used enterprise software or to delegate configurations to IT professionals. This adaptation process is both lengthy and costly (Markus & Tanis, 2000; Brehm et al., 2001; Wulf & Jarke 2004) and often has adverse consequences for businesses competitiveness.

One possible way to deal with this challenge is to enable end-users of ERP systems, from now on referred to as business users, to tailor systems to their needs, since they are managing and executing business processes on a daily basis and thus know best about changing requirements and the consequent need for adaptations. Usually, ERP systems allow users to access relevant business data directly via the user interface, or to extract data using queries and reports. However, in many cases, users need to create custom queries and build tools consuming and

processing information in a way uniquely tailored to their needs (Spahn et al., 2008b; Dörner et al., 2011). The complexity of ERP systems and available tools remain, it seems, challenging for business users with less technical knowledge.

Enabling users to adapt systems on their own, without the need of specialized technical knowledge, is one of the main concerns of approaches to Tailorability and End-User Development (Henderson & Kyng 1989; Fischer et al., 2004; Lieberman et al., 2006; Wulf et al., 2008; Spahn et al., 2008a; Dörner et al. 2009). The aim is to do rather more than design systems that are flexible and easy to adapt. It is, above all, a call for socio-technical infrastructures to enable users to participate in their work contexts through the use of their application systems (Pipek 2005; Pipek and Wulf, 2009; Stevens et al. 2010; Yetim, et al., 2012). End-user oriented approaches emphasize both the flexibility of systems for adaptability in a technical sense and the methods to leverage this flexibility at the hand of end-users. To achieve these goals may require the integration of several technologies (Dörner et al. 2011, Hess et al. 2012, Boden et al. 2014). For example, Service Oriented Architectures (Erl, 2005) provide an increased flexibility and rich possibilities for IT professionals. However, technical flexibility cannot be leveraged by end-users alone, as they typically lack the IT-expertise to make use of these complex standards. In contrast, web-based applications such as mashups (e.g. Microsoft Popfly or Yahoo! Pipes, IFTT) embody light-weight design principles and allow users to mash up data from different resources into a single integrated tool and thereby to create a new and distinct service. These mashups and composite applications were considered one of the top ten strategic technologies of recent years (Gartner 2007).

The goal of the research described in this chapter is to develop methods and tools to enable business users to create individual information artifacts tailored to their individual information needs and work practices. In order to archive this, we followed a design case study approach. In a design case study, there are three phases (Wulf et. al, 2011 and 2015):

1. Empirical investigation of the practices in a specific field.
2. Development of a hard- or software prototype based on the findings of phase 1.
3. Empirical investigation of the appropriation of the hard- or software prototype over a longer period of time.

This translates to the following three phases in our study:

First, we conducted empirical investigations into the work practices of users in SMEs to identify data-centric adaptation problems that business users face in their work context. We analyzed the results on a meta-level to identify existing composition processes of information artifacts that our solution has to support and the relevant building blocks used. As identified components need to be composed by users of a system in an intuitive and lightweight way, we also conducted a participatory design workshop (PDW) to get insights into how users intuitively create information artifacts using a simple design technique like boxes and wires.

Second, to address some of the identified problems and needs, we developed a prototypical design environment for business users, which allows them to create widgets. Widgets are small, inter- active applications for displaying data, packaged in a way to be executable on a user's machine (Caceres, 2008). By using a very lightweight mashup design technique and encapsulating mashups as widgets, we aimed to enable business users to develop small, interactive applications using enterprise resources and to deploy these applications directly to their machine, without the need for any programming knowledge. In this way, the whole chain of developing building blocks and composing them to widgets are put in the hands of SMEs.

Finally, we deployed the prototype in three German SMEs, evaluated its usability and usefulness and investigated its appropriation. We were interested how business users can create widgets using the lightweight design approach, and which practical problems can be addressed using simple applications such as widgets.

In the following sections, we present each of the steps and their results and also provide some conclusion.

2. Related Work

There are many definitions of the term 'mashup.' Depending on the perspective and context of the researchers, they define mashups according to technological or economic criteria. Some have attempted to find an overarching definition which encompasses all these factors. Koschmider et al. (2009), for example, define a mashup as 'a Web-based application that is created by combining and processing on-line third party resources, that contribute with data, presentation or functionality.' In this paper, we adopt the definition of enterprise mashups from Hoyer and Stanoevska-Slabeva (2008) 'An Enterprise Mashup is a Web-based resource that combines existing resources, be it content, data or application functionality, from more than one resource in enterprise environments by empowering the actual end-users to create and adapt individual information centric and situational applications.' When we discuss mashups in this chapter, we always refer to enterprise mashups in this way.

Several tools and approaches allow end-users to create mashups (e.g., Yu et al. 2008). In this chapter, we will focus on those that are most closely related to our work and served as an inspiration for it. We categorize these approaches and tools according to whether they support resource integration, resource transformation or widget orchestration.

2.1 Resource integration tools

Resource integration tools encapsulate resources and provide access to them. They do this in a way that allows other components to use them. This needs to be done in a seamless, substantial and appropriate fashion. Examples of resource in-

tegration tools are Dapper (http://www.dapper.net/open/) and OpenKapow (http://openkapow.com). These tools allow users to define rules to extract structured data from unstructured HTML files. The research prototype, Karma (Tuchinda et al., 2008), uses this approach and extends it by allowing users to specify examples of how the extracted data should look and automatically generate transformation rules from these examples. Another research prototype called Marmite (Wong 2007) allows end users to combine existing Web content and services from multiple Websites into new applications that were not envisaged by the websites' designers. Marmite uses a dataflow architecture that is similar to Unix pipes. It supports the extraction of content from websites (e.g., names, addresses, and dates), which can be processed in different ways, such as filtering values or adding metadata. Its output can be directed to different sinks, such as databases, map services, and web pages. However, Marmite cannot use data sources other than websites and hence cannot be utilized in an ERP context.

2.2 Resource transformation tools

Resource transformation tools support the creation of newly configured software, which recombines and transforms resources with the aim of providing the result of the transformation process as a single resource. Examples of such tools are Yahoo! Pipes, JackBe Presto Wires or IBM Damia. Users can use these tools to combine parameterizable function blocks from a predefined catalog. Yahoo! Pipes uses a similar UI design to the aforementioned Marmite. It allows end- users to manipulate, integrate and visualize data using the box-and-wire metaphor. Users can share the "pipes" they create with other users on the platform. The research prototype DERI Pipes (Le-Phuoc et al. 2009) is similar to Yahoo! Pipes but extends its approach to the usage of semantic web technologies. All of these approaches are arguably not well suited to the ERP context, however.

2.3 Widget orchestration tools

Widget orchestration tools allow the creation of widgets from the resource transformation and resource integration tools provided. We classify the created mashups into presentation mashups and application mashups.

Presentation mashups combine different widgets into one user interface without interconnecting these widgets. Often, users can simply choose from a collection of predefined widgets (stock prices, weather, news) and add them to the mashup interface. After that, the selected widgets can be customized. Examples for this are MyYahoo (https://my.yahoo.com/), Netvibes (http://www.netvibes.com/) and uStart (http://www.ustart.org/).

Application mashups are also created by combining different widgets in an integrated user interface, but they also allow the definition of application logic. Widgets can be connected to create reactive and interactive applications. One example for this is IBM QEDWiki, a prototype which lets users create portal pages out of widgets. Intel MashMaker (Ennals and Garofalakis 2007), for instance, is a

browser plug-in which allows the manipulation of existing web pages. Another system in this category is EMAP (Enterprise Mashup Application Platform), which enables users to create mashups of enterprise applications (Gurram et al., 2008). Like many such systems, EMAP comes with predefined widget components that can be combined and parameterized. In contrast to Yahoo! Pipes and Marite, it uses events for the communications between the widgets. It is one of the view systems focussed on the business domain. Ardito et al. (2014) present an integrated, general-purpose mashup environment which allows the creation of so-called Personal Information Spaces (PIS). In these spaces, users can gather, integrate and visualize data from different web services and APIs. This environment was then developed further into the EFESTO platform (Desolda et al. 2016) which also supports other data sources like Linked Open Data.

More reviews of mashup technologies can be found in Hoyer and Fischer, (2008), Koschmider et al., (2009), Albinola et al. (2009), Aumüller and Thor (2008) and Yu et al. (2008).

2.4 Research gaps

Currently, there is something gof a a lack of empirical studies in the area of enterprise mashups. Most research is focused on technology and tools. Papers discussing the use of mashups in companies (e.g. Soriano et al. 2007, Hoyer and Stanoevska-Slabeva (2008)) are based in the main on theoretical considerations and explorations of general trends. There are some empirical studies, but they only deal with the usability aspect of mashup tools (e.g. Wong, J. (2007)). Evaluation is not typically done in the field. In this paper we provide insight into how these tools are used in real work environments and investigate the utility of mashup approaches in the organizational contexts for information access.

3. Empirical Investigations in Practice

3.1 Exploration of Users' Problems and Needs in Practice

To understand the problems experienced and requirements for the design of practically relevant tools, we conducted 14 semi-structured interviews based on qualitative research methods (Kvale, 1996) in three German midsized companies that use the SAP ERP system to support their business.

Company A is a producer of bags, backpacks, suitcases, and similar items with a staff of 140. A partnering company in Asia procures most of the raw materials and manufactures subassemblies. Final assembly, marketing, and sales of the products take place in Germany. Company A does not sell directly to retail customers but uses resellers. Company B is a software company which develops and sells its software products but also develops software on commission. It has 500 employees. Company C produces custom textiles for the automobile industry and employs 137 people. So, two companies we investigated were from production industry (137 and 140 employees), and one was a larger software vendor (500 em-

ployees). In all three companies, the IT department manages the introduction or modification of software together with technically adept staffers from the affected department. These so-called "key users" help to disperse the knowledge about the new system in the company and are local counterparts for other users who experience problems or have questions about the software. Every department has one key user. In some cases, external consultants are also involved in the introduction process.

We conducted semi-structured face-to-face interviews in an exploratory way to get insights into operational tasks and existing work practices. This investigation revealed that users face two classes of problems associated with information display:

- Information related to a particular task is scattered around the ERP application, hard to find, or only available in several different applications.
- Information cannot be filtered, accumulated, combined or displayed in a way suitable to a specific task.

As an example for the first category of problems, one interviewee who was in charge of reviewing customers credit lines reported the following: "I occasionally have the problem that I need to access four or five things in one row in order to get what I need […] for example, when I do the annual review of credit lines. I need master data, data from SD [SAP's ERP module for sales and distribution] and some data from accounting [...]". An example related to finding information was given by another interviewee: "When it is about master data, then I know that I can jump to the field in most cases and press F1 for help, where the data is saved. [..] But there are things which you cannot find this way, not even X[1] from the IT department."

As an example for the second category of problems, an IT manager described the following case: "[the ERP system] sums up the revenue numbers on a monthly basis. To calculate the daily revenue, we access the revenue numbers every morning at the same time and manually enter them in an Excel sheet. By subtracting the collected sums, we then calculate the daily revenue numbers."

We found several cases in which Excel sheets were used to deal with problems in the second category. These Excel sheets can get quite sophisticated, as one user describes: „To plan how much to order, I separate the year into several time periods based on experience. […] I can estimate when demand is going to be high or low, which colors sell the best and if a product introduction causes additional demand." The user created his individual, complex solution which he has to update manually at regular intervals to carry out this planning and forecasting task. More concrete examples of problems in the two categories can be found in Spahn (2010).

Due to the complexity of the data model exposed by the ERP system and the complexity of the tools for query creation provided by the ERP system, most users

[1] All names are anonymized to protect participant's privacy.

were not able to create custom queries on their own in order to get data out of the ERP system in the way they desired. Additionally, business users were not able to create any custom information artifact providing interaction on live data with the ERP system. Many users have to access a particular set of data relevant to their individual working tasks from the ERP system many times a day. In many cases, this data cannot be accessed from a single location within the Graphical User Interface (GUI) of the ERP systems, forcing users to access multiple locations and collect the needed data in an unnecessarily cumbersome fashion. As they are not able to create any customized GUI or interactive application, providing access to relevant data at a glance, many working tasks can only be executed in an inefficient and cumbersome manner. More details of this study are documented in (Spahn et al., 2008b).

3.2 Observing the Users' Design

A prerequisite for building lightweight design environments for composition is the identification of a straightforward and easy-to-use design technique for composition. We therefore investigated the effects of confronting users with a simple design metaphor like boxes and wires (e.g. employed by Mashup tools like Yahoo! Pipes). We chose the boxes and wires technique because it enables visual modeling while using a minimum of different design entities. To get initial insights into how users interact with a fairly elementary modeling approach, we complemented our interview study with a PDW, putting users in the role of designers (Muller, 2003).

In the PDW business, users had to design an information artifact representing a tool that could support users in an analytic task which participants took from their work context. To create the solution, the users could add boxes to the design space that represent data or functionality. Boxes had output ports and input ports and users could connect these ports by drawing lines to define data or control flow between the boxes. The boxes and wires model was left underspecified. We gave no concrete instructions to the users on how to formally specify the meaning of the used design elements. The underspecified semantics enabled us to observe users' natural design behavior, while they were using the design elements (Pane et al., 2001) more or less intuitively. The design behavior and the created artifact were analyzed to get a better understanding of how users approached this design task.

The PDW revealed that users intuitively thought of boxes as a representation of tabular data, organizing data in rows and columns. By connecting boxes, business users related data from different boxes with each other and defined the data flow of the solution. The users had no problems using the boxes and wires design paradigm itself to specify a solution but had problems expressing what data a box should represent. The users decided to describe the data by giving a short description on how they would access the data in their used ERP system. Details of this study are presented in (Spahn et al., 2008b; Dörner et al., 2011).

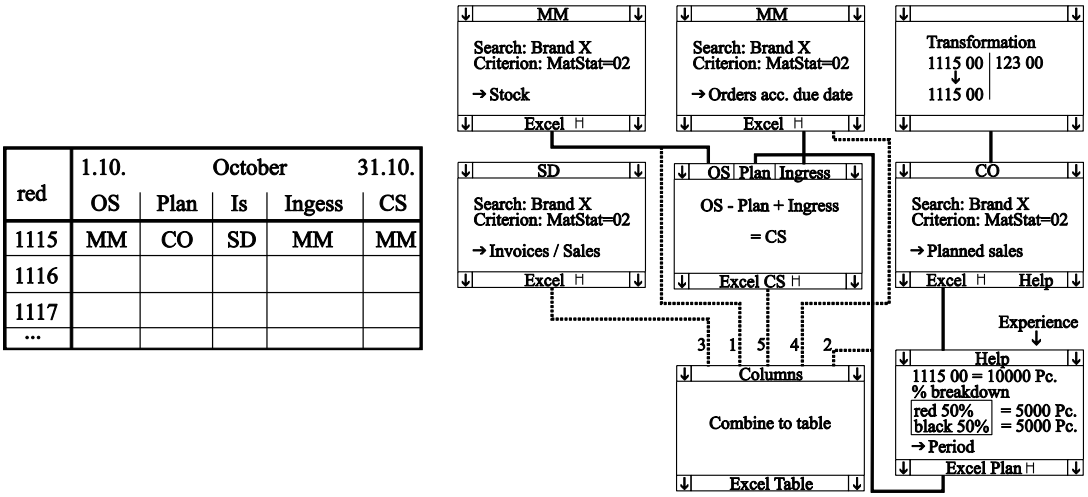

Figure 1: Visual concept of the desired output of the system (left) and of the user interface (right) designed in the PDW.

4. Design and Development of Widget Composition Platform

4.1 Motivation and Preliminary Remarks

To address the problems and needs identified in the empirical investigations, we subsequently developed two prototypes for supporting business users. These tools solve two main problems: First, business users create custom spreadsheets and rely on getting relevant data from the ERP system by using queries, but face considerable challenges when trying to define custom queries for their individual information needs. The first application, called "Semantic Query Designer" (SQD), provides sophisticated visualization, navigation, search and query building possibilities and enables business users to create custom queries. A detailed description of this application is provided in (Spahn et al., 2008c).

The second central problem concerns the fact that business users needed to access the same set of data within the GUI of the ERP system over and over again in a cumbersome manner and were not able to create custom interactive applications that provide the information relevant to individual working tasks at a glance. In response to this problem, we developed a web-based environment called "Widget Composition Platform" (WCP), to enable business users to mashup enterprise resources in a visual design environment in a very lightweight way and to deploy the created mashups in the form of widgets to their local machines.

In this section, we will describe the functionalities and architectural components of the WCP. In response to the positive results of the PDW on the usefulness of box-and-wire technique, we decided to implement this metaphor in the WCP for offering a very lightweight visual mashup design environment. Boxes represent services that provide enterprise data that are in most cases rendered as a

table. As the empirical investigations revealed, business users were able to use services providing data in a tabular format in an intuitive way for designing software artifacts.

For the realization of a visual mashup design environment, we could build on an early internal prototype of SAP Research, one of the partners in the consortium, that we modified and extended to suit our needs. Despite this, we faced considerable challenges in realizing a solution that could be deployed and used within SMEs for two main reasons. First, SMEs often use ERP systems that are not (yet) service-enabled and thus cannot provide enterprise resources as services that are accessible using standard web protocols, which is an essential precondition for deploying developed mashups as a widget on standard widget runtimes. Second, even if enterprise systems were exposing a fixed set of resources as services, this set could not be extended without programming skills and thus would limit the creatable solutions to combinations of predefined services. To address these challenges, we implemented middleware which can wrap resources from not previously service-enabled ERP systems and provided these as services which were accessible using standard web protocols. SMEs must be able to create new services using existing knowledge and without any programming skills, since some of their members, at least, do not possess the requisite skills. There are SMEs which can build queries within the existing ERP system, so we enabled the implemented middleware to wrap queries stored within the ERP system and expose these as services. Using the existing skills of query creation to allow the creation of new building blocks for widget creation has several advantages. It limits entry barriers, enabling users to utilize the new technology in a flexible way. It can also be seen as a kind of gentle slope of complexity approach (MacLean et al. 1990, Wulf et al. 2008, Spahn et al., 2008a) that puts the whole chain of widget creation in the hand of SMEs. Additionally, by enabling the use of queries as services, a relation between the prototypes SQD and WCP is established, as end-users can use SQD for easy query creation, and WCP to mash up these queries to widgets.

In the following subsections, we describe the system prototype in detail. We first describe the basic conceptual layers that are implemented by the WCP environment and then explain architectural components of the WCP environment and their interaction. Finally, we describe the user interface aspects and explain how business users can create widgets by mashing up enterprise services.

4.2 Conceptual Layers

On a conceptual level, the WCP and created widgets are based on a layered architecture. Significant components of this conceptual architecture are classified and structured in the widget stack depicted in Figure 2. The widget stack consists of five primary layers: resource layer, application programming interface (API) layer, wrapper layer, service layer, and mashup/widget layer. We will describe the layers in figure 2 bottom-up, as each layer requires the functionality provided by its subjacent layer (Spahn & Wulf, 2009, Spahn, 2010).

Resource Layer. The resource layer consists of all resources that can be integrated into mashups. Resources can be data such as customer master data stored in an ERP system, or functionality, such as locating an address and returning an according image of a map provided by a map application. Resources are managed and provided by systems that can be internal or external to an organization. Internal systems might be various enterprise systems, like ERP or Customer Relationship Management systems, or general-purpose relational database management systems (RDBMS). External resources might be provided by systems accessible to a closed user group, including systems of suppliers, customers or B2B marketplaces, or by systems publicly available over the internet, e.g. map services or stock quotes.

API Layer. The API layer consists of the well-defined interfaces provided by the systems managing the resources. Depending on these systems, different formats and protocols may be needed to call the APIs and access the resources. APIs of modern service-enabled systems may be exposed as web services that can be called using standard web protocols, while legacy systems may expose APIs only as libraries that can be linked into source code and communicate with the system using proprietary protocols. Required formats of input parameters and formats of returned results vary accordingly and range from XML structures to proprietary binary formats.

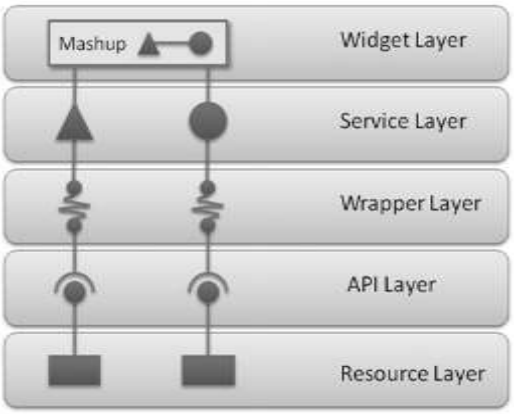

Figure 2: Conceptual layers of widget stack in the WCP environment.

Wrapper Layer. The wrapper layer provides a unified service model. The service model is consumable by the service layer and abstracts from the heterogeneity of APIs, protocols, and formats. More specifically, wrapper components transform abstract requests to the abstract service model to concrete requests using the respective API, protocol and parameter format of the addressed resource. These components convert raw results received from the API to a format that the unified service model can process (e.g. data structures storing tables, lists or even images). Each resource does not require its individual wrapper component. Generic wrappers can wrap certain types of resources, like SAP ERP queries or RSS feeds, in a

generic way. These generic wrappers are exposed as service types in the service layer.

Service Layer. The service layer provides a repository of services that can be mashed up in the mashup environment. All services are parameterized instances of service types, relating services to a certain wrapper component in the wrapper layer. For example, a service "Customer data" providing data from a SAP ERP query is an instance of the service type "SAP ERP Query" that relates the service to an appropriate wrapper component. The service instance uses parameters to specify the concrete query inside the SAP ERP system which the wrapper component can access. Services provide further configuration possibilities that vary depending on the type of service. For example, the desired type of data visualization (e.g. rendering data as a table or a list), or the set of available data filters that the GUI exposes in the mashup environment, can be configured. Services can be further described by adding information like name and description text.

Mashup / Widget Layer. The mashup and widget layer adds mashup functionality to the unified service model of the service layer and provides the functionality to deploy mashups as widgets. Users can define mashups by specifying a wiring of design elements, interconnecting design elements and defining the desired data flow. Design elements can be services or additional UI elements (e.g. text boxes). The wiring defines how data from one design element is used to parameterize calls to other (de- pendent) design elements. Dependent design elements react on data updates of connected design elements and update their data accordingly. Mashups can be encapsulated as widgets and deployed as self-contained applications to individual runtimes, e.g. the Yahoo! Widget engine (http://widgets.yahoo.com). The mashup creation and widget deployment functionality are provided by the mashup and widget layer to end-users by an integrated EUD environment.

Relevant business data needs to be available as services in the service layer to enable business users to create custom widgets supporting their individual working tasks. Therefore wrapper components and service types need to be implemented to be able to wrap resources from relevant enterprise systems.

4.3 Architectural Components

Our prototype instantiates the conceptual widget stack (Spahn & Wulf, 2009; Spahn, 2010). Figure 3 provides an overview of the system components. The central component of the prototype is the WCP, a web application which is implemented using Java technology and runs inside a web application server. The WCP provides an integrated, web-based, graphical end-user development environment for widget creation. A service repository manages all services that users can mash up within a widget. A widget deployment component within the WCP is responsible for creating source code, encoding the currently created widget for multiple runtime environments. During the process of widget composition within the WCP user interface, this component generates code for the WCP browser runtime libraries, so that the widget is fully functional within the user interface that the browser

renders. If the user decides to deploy the developed widget to a widget runtime engine, the component generates code tailored to that engine. It also packages the widget to a file of according structure and format. To persist data, like the service repository, the personal repository of end-users' created widgets, or login and access right information, the WCP is using an RDBMS.

On the client side, end-users access the user interface of the WCP by using a web browser to call a certain URL. This simplified access to the WCP makes it very easy for end-users to start with the development of individual widgets. Within the browser, widgets run based on a WCP runtime library implemented in JavaScript using common web standards. This enables a rendering of the widget and providing full widget functionality inside the browser during the development process. For deployment, the WCP is delivering a single file that contains the widget packaged in the specific format required by the runtime environment.

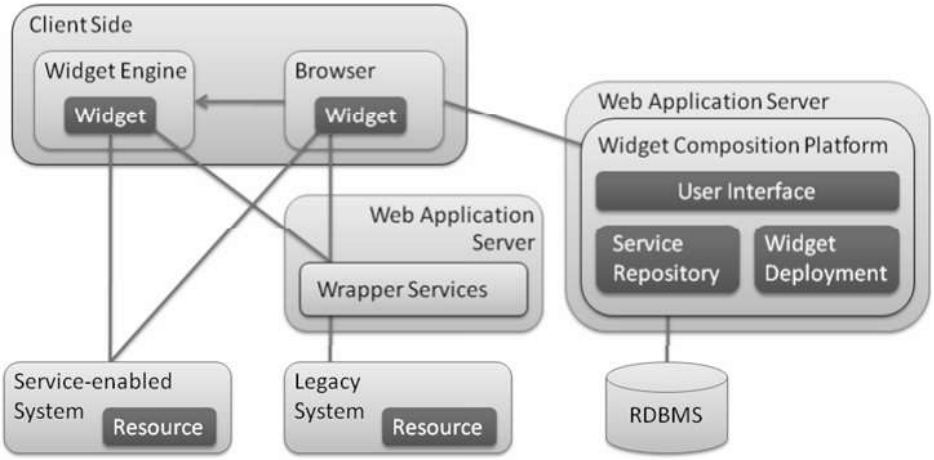

Figure 3: Architectural components of the Widget Composition Platform.

In accordance with the widget stack described in the previous subsection, widgets access the resources that are mashed up within the widget via wrapper components. If service-enabled systems manage resources, wrapper components can address the resources directly using standard web protocols. If legacy systems that do not provide an API addressable using standard web protocols manage the resources, wrapper components are assisted by wrapper services. Wrapper services are deployed to a dedicated web application server and provide access to legacy systems. In our particular setup, we implemented wrapper services to access queries within not service-enabled versions of SAP ERP systems by encapsulating SAP Remote Function Calls using SAP Java Connector, and execution of queries expressed in Structured Query Language to RDBMS or Excel files using Java Database Connectivity. This implementation option enables wrapper components to access resources within such systems using simple web protocols, keeping

the needed technology on the client side as straightforward and lightweight as possible.

4.4 The User Interface of Widget Composition Platform

The WCP provides a browser-based user interface representing an integrated development environment enabling the visual development of widgets without any programming knowledge. Figure 4 shows a screenshot of the user interface.

We separated the user interface into several panes. On the left side, a list of all services contained in the service repository is provided. The list groups services according to their service types. Users can add services to a mashup, simply by dragging and dropping the service to the design pane, located in the middle of the interface. Besides services, additional design elements, like text boxes, can be dragged from the upper right of the interface into the design pane. If a UI element in the design pane is selected, its properties are visible and modifiable in a properties pane on the right. Properties that can be modified include visual properties (such as color or font faces), but – more importantly – structural properties of services. From the background pane at the bottom of the user interface, the user can select a background image, enhancing the visual appearance of the created widget. After adding services or design elements to the design pane, they are immediately populated with live data and provide runtime interaction possibilities. All design decisions create immediate effects, thus blurring design time and runtime and enabling development close to a WYSIWYG manner, increasing the confidence of the user in creating the desired results.

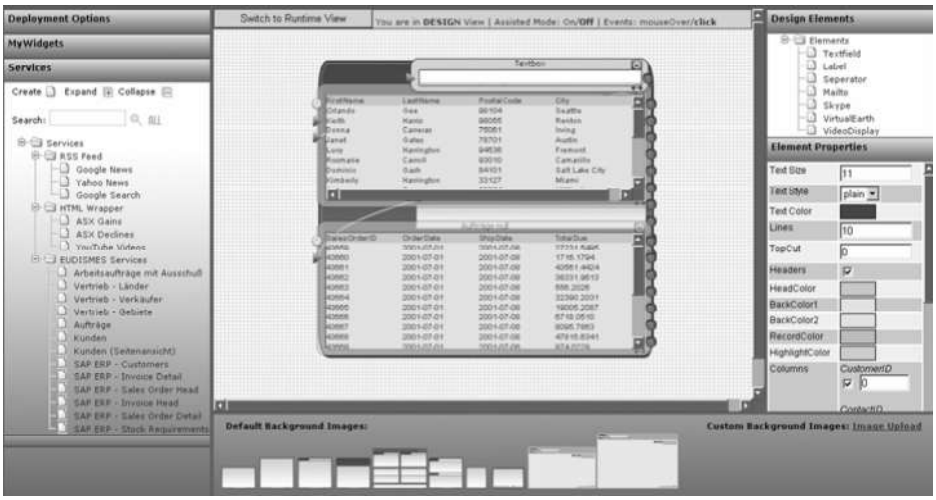

Figure 4: User Interface of the Widget Composition Platform.

In order to mashup services, a wiring has to be defined using a simple box and wires. As depicted in Figure 5, services and other design elements offer input ports and output ports that can be connected to define the wiring. Users can connect elements by drawing a line, originating from an output port of a source element to an input port of a target element. Whenever data in the source element changes, the new data is pushed as input to the target element, which updates itself accordingly. The update of data in target elements might again push new information to dependent elements, which in turn might trigger updates. In the given example, a service providing customer master data is connected to a service providing sales order data. When selecting a customer in the customer table, the connected sales order service is automatically updated to show only sales orders of the selected customer. In the case of the depicted services, input ports correspond to filters on table columns and output ports to the column values of the currently selected row. In the given example, two text boxes are used to enable the user to interactively define values that are pushed to services as input. The text box connected to the customer data service restricts the shown customers according to the given name pattern. The text box joined to the sales order service restricts shown sales orders to such sales orders that have been received at or after a specific date. In addition to text boxes, other design elements exist, that provide additional functionality. Examples are design elements that use e-mail addresses or Skype names as input and provide the ability to send an e-mail or establish a Skype call just by clicking on an envelope or telephone symbol.

If a resource for the current mashup is missing in the service repository during the creation process, an appropriate service can be defined using the "Create" option of the service pane. To define a service, the user needs to select the service type (e.g. SAP ERP query) and then the resource that the service should encapsulate (e.g. by providing the SAP ERP system storing the query, and the name of the query). After saving the service to the repository, it can instantly be used in the mashup process.

At any point in time, the user can switch from the design time mode to a runtime mode (design principle of direct activation, see Wulf & Golombek 2001). Although the widget is fully functional even in design time mode, the run- time mode hides all the visual elements that users only otherwise need at design time, like input ports, output ports, or connections, and prevents modifications to the widget. In this way, the widget is presented as it would appear if deployed to an external widget engine. Widgets can be saved to and loaded from a personal widget repository managed by the WCP. This functionality is accessible through the "MyWidgets" pane on the left of the user interface. Using the "Deployments Option" pane, the system can deploy the widget to multiple widget runtime environments, like Yahoo!

Widget Engine (http://widgets.yahoo.com) or Microsoft Windows Vista Sidebar (Lal, 2008).

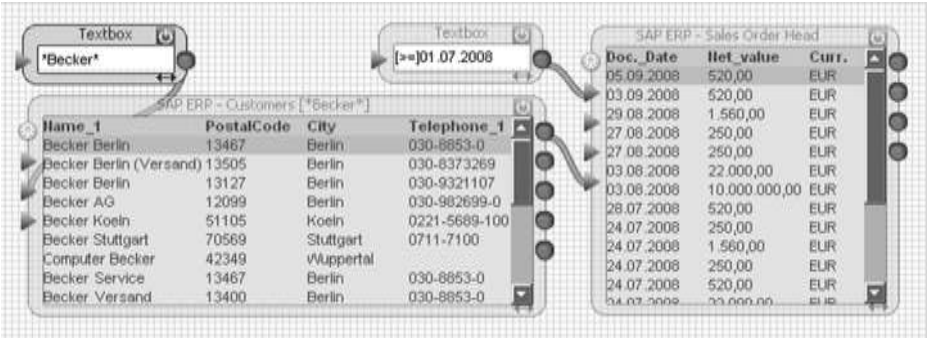

Figure 5: Wiring of services and design elements.

When selecting deployment, the WCP is returning a single file, which can either be directly opened and thereby gets deployed to the client-side widget engine or saved as a file, which can easily be sent via e-mail or moved to a file share, to share the widget with colleagues. Widgets that are deployed to a client-side widget engine run independently of the WCP and are small, self-contained, interactive applications.

5 Appropriation Study

5.1 Goals and Methods

The WCP environment designed for enabling business users to create custom widgets was introduced in practice to explore whether business users in SMEs would create widgets using the WCP and how they might appropriate the widgets for addressing practical problems in real work contexts.

For this purpose, we have conducted two complementary studies in the three German midsized companies in which we conducted the pre-study (interviews, PDW). The first study considered use cases in practice. Its goal was to observe how users adopt and use of the WCP to create widgets and services. The second study conducted a questionnaire-based survey among the employees as potential users of the WCP to get feedback from a broader user base on the proposed WCP-based solution. In the following, we briefly present the qualitative study.

5.2 Evaluation through Use Cases in Practice

In this section, we first describe the setup of the practical evaluation of the WCP and then the results of the use within SMEs through example use cases.

Setup of Evaluation in Practice. We deployed the WCP environment in the three German SMEs. Among those, 50, 70 and 116 employees have access to the SAP ERP system. 18, 60 and 70 employees use the SAP ERP system on a regular base. In two of the three companies, the WCP was initially installed in cooperation with the IT department, and in the third business, it was installed in collaboration with the person responsible for IT concerns, as no dedicated IT department existed. After installation, the WCP could be used by every employee having access to the internal network of the company.

In every business, one advanced end-user was nominated to be the contact person, responsible for all concerns about the WCP. This person, in principle, acted as an evangelist to promote the usage of the new technology, as well as provide support to users if questions arise. In a first phase, the contact person was given an introduction into the WCP and should experiment with it to get more familiar with its concepts and usage. We defined five distinct services encapsulating SAP ERP

data as a starting point for experiments. The services provided ordinary data related to customers, sales orders, and invoices. To increase the motivation for experimentation, we included some appealing external services that users could access, like a service for visualizing addresses on a map, a YouTube video service, Google news, and a stock quote service. In a second phase, we discussed any problem that might have arisen in the first one and provided help to solve these problems. After that, we discussed potential use cases of widgets within the company with the contact person. In a third phase, we encouraged the contact person to act as an evangelist and promote the usage of the WCP by approaching employees who might be interested in the discussed usages, giving them an introduction to the WCP, and motivate them to explore. During the third phase, we conducted accompanying interviews with the employees who were using the WCP for the creation of individual widgets. Our aim was to investigate WCP's appropriation and how end-users use the WCP and widgets to address practical problems.

Appropriating WCP and Widgets in Practice. In the following, we exemplarily discuss four instances which describe how business users appropriate widgets.

Case 1: Sales Support Widget. A staff member in the sales department of company A is in charge of answering questions from customers related to their orders. Customers contact the employee by phone to get information related to the content and status of their orders. Inquiries resulting from such calls might be questions as to whether or not certain goods have been bought to fulfill a specific sales order, or what the current state of the order's processing is, for example. To be able to answer such questions, the sales officer has to access multiple locations inside the GUI of the SAP ERP system. In a first step, the user needs to access the customer's master data to identify him. In a second step, the user accesses sales order header data to filter sales orders of the respective customer and the sales order in question. In a third step, the individual items of the sales order are accessed to track the ordered goods, the quantity, and their status. If the customer enquires about items on multiple sales orders, the user needs to switch back and forth between the different sales order list and their details. The sales employee we talked to considered incrementally gathering required information via the GUI by accessing multiple reports and switching back and forth between them to be rather cumbersome, especially since he has to access the information many times a day – following different calls.

The evangelist approached the sales staffer and introduced the WCP to them and the staffer started to experiment with the WCP. Using the services we had defined as standard demo services during installation, the employee was able to create a suitable widget for his needs. The widget shows customer master data, sales order header data, and sales order details as three distinct tables. Using text box UI elements, the sales employee added filters for customers by name or customer number, and filters on the order date of sales orders. He deployed the widget to his desktop to make it readily available when customers required information. By using the filters based on customers' master data, a client can be uniquely identified

in a rapid and convenient way. By clicking on a customer in the table, all sales orders of the buyer come up in another table, directly beneath the customer table. The user then can restrict the visible sales orders to the ones ordered at or after a specified date. By clicking on a sales order, all relevant details are shown in a third table. The sales staff in question configured the tables to only show data relevant to him, resulting in a compact overview providing required data at a glance. To view details of other sales orders in question, the employee can now simply click on the sales order in the sales order table.

The created widget is used by the sales staff some 40 times a day to answer standard questions of customers related to their orders. As the user does not need to access multiple tables/reports within the complicated GUI of the SAP ERP system but gets all relevant data at a glance, he can answer standard customer inquiries in approximately half the time compared to using the GUI of the SAP ERP system. Because of this, the sales staff decided to send the widget via email to a colleague who needs to access the same data occasionally. This employee had no programming skills and was able to create a custom widget for supporting his individual working task within three hours after having seen the WCP for the first time. He explained to us that experimenting with the WCP was fun for him. He perceived the WCP to have an appealing user interface and the composition of widgets to be simple, comprehensible and easy to learn.

Case 2: Material Lookup Widget. An employee in the procurement department of company A needs to access certain information related to material many times a day. For instance, he needs to identify a specific spare part by its material number and finding out about the quantity currently in stock and the quantity already scheduled for production. To get a first overview of the status of the spare part, the employee could determine a rather fixed set of information which she considers to be highly relevant for her work context. Similar to the previous case, this information is widely spread within the GUI of the SAP ERP application and users have to gather it in a cumbersome manner.

When the evangelist introduced this employee to the WCP, she immediately thought of building a widget to tailor her standard data set of related information. As predefined services that might deliver relevant data did not exist, the evangelist discussed with the employee which data was needed. Then he searched for possible data sources like tables and queries within the ERP system. Building on a larger query, the evangelist managed to create a SAP query joining five distinct tables and providing most of the requested data. The query was wrapped as a service for

the WCP and was used by the user to create a suitable widget.

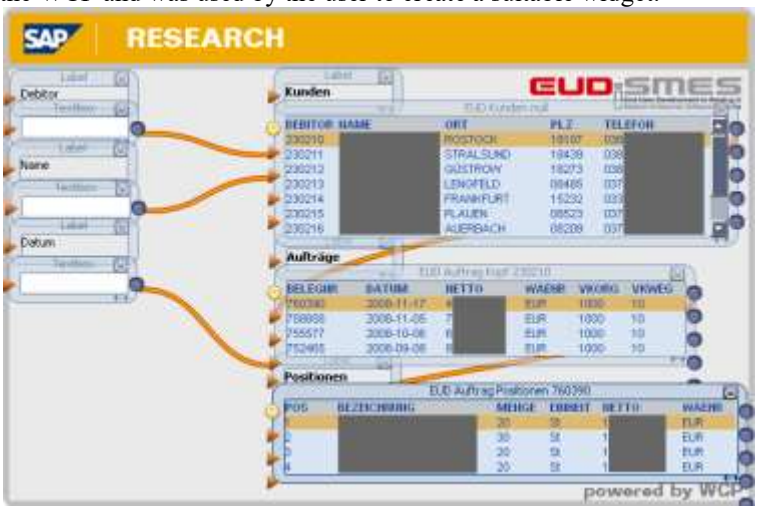

Figure 6: Screenshot of the material lookup widget in design mode

The widget is rather simply structured and consists simply of a text box and the created service. The text box defines a filter on the material number to the service. As the service just returns a single record, the results are not rendered as a table with a single row, but as a list, showing all attributes of the record with according values as rows. The user in this case configured the service to show only the most relevant data and arranged all design elements neatly to create an appealing widget. We also deployed the same widget on the PC of an employee working in the raw material storage facility, who did not have direct access to live data from the SAP ERP system before (due to SAP's license fees). By using the widget, he was able to access the most critical data related to material in a very easy way, without the need of having to learn and understand a complex enterprise system.

Case 3: Address Book Widget. All three businesses keep Excel sheets of employees and their contact data, like telephone, fax, and mobile phone numbers, e-mail addresses and departmental affiliation. Staffers distribute these sheets via email among themselves. To have this list readily available, employees tend to keep the latest version of it in designated folders on their PCs or print it out.

To improve on this process, the WCP evangelists in all three firms independently developed an address book widget. The evangelists took the data from the existing Excel sheets which they copied on network shares and integrated into the WCP as a service. Figure 7 shows two examples of the address book widget from two different companies.

Figure 7: Screenshots of two of the three address book widgets.

The left widget allows users to search for employees by name. The search results are presented one at a time and can be navigated using a "back" and a "forward" button. E-mail addresses are clickable links. The right widget uses a table to display the first 20 search results. In this widget, employees can not only be searched by name but also by departmental affiliation. The WCP evangelist shared the widgets using e-mail. These e-mails also contained links to the Yahoo! Widget Engine, which allowed staffers to use the widget directly on their desktops, without a web browser. The Yahoo! Widget Engine also enabled users to set a hotkey for the widget.

The widget was adopted quickly in all three firms, and users liked its handiness. An employee in company A remarked: "The address list was a good idea; it is something practical. [...] Everyone can make good use of that [...]. Otherwise, I would have to look for slips of paper."

The address book widget was widely used and distributed in the three businesses and therefore was the first contact many staffers had with WCP and widgets in general. It stirred interest in WCP and introduced employees to the idea of widgets which can be created, modified and used by end-users.

Case 4: Birthday widget. The birthday widget shows how users can appropriate widgets to their needs through modification and adaptation. Staffer X in the quality assurance department of company B received the address book widget via e-mail. He then asked the local WCP evangelist about creating such widgets himself. The local evangelist then introduced WCP to staffer X. As staffer X kept a list of his colleagues' birthdays in an Excel sheet, he decided to adopt the address book widget so it can be used to keep track of his colleagues birthdays. To archive this, the staffer copied the existing address book Excel sheet and added a column with the birthday information. Employee X then made this copy available in WCP creating a service. He completed this task without problems by using the existing address book service as guidance. Then he made a copy of the address book widget and modified it. The staffer changed the data source to the newly created ser-

vice, added a text field for filtering the list by the date or date range. He also changed the layout and added labels to the new UI element. The resulting widget is shown in figure 8.

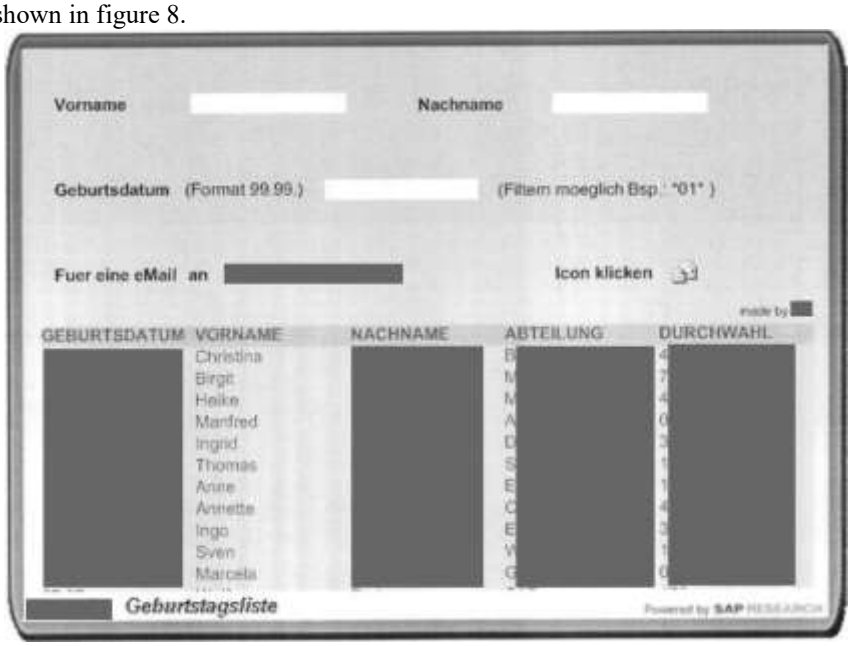

Figure 8: Screenshot of the birthday widget.

The staffer sent his widget to about ten of his colleagues, which in turn sent it to more co-workers. The widget became quite popular in the company and is jokingly referred to as "lunch optimizer", as many employees use it to check if there is a birthday celebration with some snacks or cake in which case they hold back on lunch.

Collaboration. In the evaluation, we identified three main types of users who collaborated in different ways:

1) Very advanced users (mainly evangelists who have the technical knowledge to create data sources and services), who helped other users to understand and use the application.

2) Key users, who often identified fields of application where the use of widgets would make sense. They came up with ideas for new widgets and implemented them by combining existing services. If these services did not exist yet, they created them in collaboration with the advanced users. Key users also distributed the widgets to other end users.

3) Other users, who were mainly widget-consumers, and who only used and shared widgets. They did not create or modify widgets but gave suggestions for improvements to the widget creators.

6 Conclusion

In this chapter, we presented a design case study from the domain of End User Development. The WCP environment enables business users to create enterprise widgets for their personal information needs. The design approach, i.e., the web-based environment WCP, enables business users (a) to mash up enterprise resources in a visual design environment in a lightweight way using a simple box and wires configuration technique and (b) to deploy the created mashups in the form of widgets to their local machines. In our case, these new functionalities for creating custom widgets allowed business stuff to create small, interactive applications to display relevant data in a fast and convenient way, without the need of starting heavyweight and complex enterprise systems and cumbersomely collect data from multiple locations. Additionally, advanced users can now create new services to be mashed up within the WCP by creating and wrapping related resources within enterprise systems.

Our approach to the development of the WCP follows the design case study approach. We first presented the results of empirical investigations in real enterprise contexts, which we conducted in order to understand the practical problems and the pressing needs of users when working with ERP systems. Next, we described the prototyping leading to the WCP. We explain its basic concepts, the architectural components, and how they work together. We also explained how business users can create individual mashups of enterprise resources using the box and wires tailoring metaphor, and how the tailored mashups can be deployed and exchanged as widgets on the local computers of the users. Finally, we describe how end-users appropriated the WCP in three German SMEs. Supported by a local expert, the office workers were able to build mash-ups which assembled tables in such a manner that their search for data from the ERP system was considerably eased. So, the WCP enabled business users to tailor custom widgets supporting their individual work tasks.

Beyond this, we have learned from our observation of the widgets' use in three different organizational contexts that the appropriation of WCP is a collective endeavor (cf. Nardi 1883; Wulf 1999; Kahler 2001; Pipek and Kahler 2006). Based on our observations, it makes sense to distinguish between three different types of users, widget consumers, widget creators, and service creators.

- Widget consumers are only applying widgets but do not create them. These users are typically less familiar with IT in general or show less of a willingness, at least at the moment, to engage in tailoring. They receive widgets from more experienced colleagues and use them to access data in an easy and comfortable way without the need of learning and understanding one or more complex information systems (Nardi 1993; Wulf 1999; Kahler 2001; Wulf et al 2008). The value of widgets for this type of users is that it allows them to efficiently access the relevant information without the need to ask colleagues.

- Widget creators are users that create widgets for themselves or others to support individual working tasks. They are motivated by multiple reasons, including the need to simplify data access for their working tasks or to provide others with specially tailored widgets. This improves data supply to others engaged in the same business processes. Providing better tools in support of their work practice is especially relevant for staffers that are responsible for certain processes and motivated to act as widget creators to optimize these processes and improve process performance.
- Service creators are advanced users that can create new resources that can be wrapped and added to the service repository, from where they can be used for widget creation. By extending the service repository with new services, they enable widget creators to address more use cases and create more and more precisely tailored widgets. Without service creators, widget creators would be limited to a certain set of predefined services which restricts the number of creatable solutions. Concerning the WCP environment, service creators are end-users having the skills of creating, modifying or at least locating suitable queries inside the SAP ERP system that match existing information needs of widget creators.

Overall, our research contributed to the field of End-User Development by providing the WCP environment, which enables business users to create simple software artifacts like widgets and to address practical problems in real enterprise work contexts. The study has also shown hot the design case study approach can be used to design and develop a usable and useful EUD environment. The approach allowed us to (1) analyze and document the work practices in the three companies, (2) develop a software system which supports these practices and (3) observe use, appropriation, and tailoring of this application in the different companies through various users. From this analysis and observation, we provide indications that by enabling SMEs to create enterprise resources using existing knowledge and wrapping these resources as building blocks, the whole development chain of software artifacts can be put in the hands of SMEs. This shift would reduce the need for external IT professionals and at the same time increase flexibility in adapting the used software infrastructure to emerging individual needs.

7 Acknowledgments

The research presented here was funded by the German Federal Ministry of Education and Research (BMBF) as part of project EUDISMES (number 01ISE03C).

8 References

Albinola, M./Baresi, L./Carcano, M./Guinea, S. (2009): Mashlight: a Lightweight Mashup Framework for Everyone. In: Proceedings of 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009). Madrid, Spain, 20.04.2009.

Ardito, C., Costabile, M., Desolda, G., Lanzilotti, R., Matera, M., Piccinno, A., & Picozzi, M. (2014). User-driven visual composition of service-based interactive spaces. Journal of Visual Languages & Computing, 25(4), 278–296. Elsevier.

Aumüller, D./Thor, A. (2008): Mashup-Werkzeuge zur Ad-hoc Datenintegration im Web. In: Datenbank-Spektrum, 8(26), S. 4-10.

Boden, A.; Dörner, C.; Draxler, S.; Pipek, S.; Stevens, G.; Wulf, V. (2014): Tangible and screen-based interfaces for End-user Workflow Modeling, in: IEEE Software, Vol. 31, Nr. 4, 65-71

Brehm, L., Heinzl, A., Markus, M.L. (2001): Tailoring ERP Systems: A Spectrum of Choices and their Implications. In: 34th Annual Hawaii International Conference on System Sciences (HICSS-34). IEEE (2001)

Caceres, M.: Widgets 1.0 Requirements. W3C Working Draft. W3C (2008)

Carter, K.; Henderson, A.: Tailoring Culture In: Hellman, R.; Ruohonen, M.; Sorgard, P. (eds): Proceedings of the 13th IRIS, Reports on Computer Science and Mathematics, No. 107, Abo Akademi University 1990, pp. 103–116.

Davis, F.D. (1986): A Technology Acceptance Model for empirically testing new End-User Information Systems: Theory and Results. PhD thesis, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, USA (1986).

Davis, F.D. (1989): Perceived Usefulness, perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly 13, 319 - 340 (1989).

Desolda, G., Ardito, C., & Matera, M. (2016). EFESTO: A Platform for the End-User Development of Interactive Workspaces for Data Exploration. In Rapid Mashup Development Tools (pp. 63-81). Springer International Publishing.

Dörner, C.; Draxler, S.; Pipek, V.; Wulf, V. (2009): End-Users at the Bazaar – Designing the next Generation of ERP Systems, in: IEEE Software, Vol. 26, No. 5, 45 - 51

Dörner, C; Yetim, F.; Pipek, V.; Wulf, V. (2011). Supporting Business users in Tailoring Business Processes. Interacting with Computers 23 (3), 226-238.

Ennals, R. J., & Garofalakis, M. N. (2007, June). MashMaker: mashups for the masses. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data (pp. 1116-1118). ACM.

Erl, T. (2005): Service-oriented Architecture: Concepts, Technology, and Design. Prentice Hall (2005).

Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N., 2004. Metadesign: a manifesto for end-user development. Communications of the ACM 47, 33–37.

Gartner (2007): http://www.gartner.com/newsroom/id/530109.

Henderson, A.; Kyng, M.: There's No Place Like Home: Continuing Design in Use. In Design At Work - Cooperative Design of Computer Artefacts, J. Greenbaum and M. Kyng, Eds. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers, 1991, pp. 219-240.

Gurram, R., Mo, B., & Gueldemeister, R. (2008, September). A web based mashup platform for enterprise 2.0. In International Conference on Web Information Systems Engineering (pp. 144-151). Springer Berlin Heidelberg.

Hess, J.; Reuter, C.; Pipek, V.; Wulf, V. (2012): Supporting End-User Articulations in Evolving Business Processes: A Case Study to Explore Intuitive Notations and Interaction Designs, in: International Journal on Cooperative Information Systems (IJCIS), Vol. 21, No 4, 263-296

Hoyer, V., & Fischer, M. (2008, December). Market overview of enterprise mashup tools. In International Conference on Service-Oriented Computing (pp. 708-721). Springer Berlin Heidelberg.

Hoyer, V., & Stanoevska-Slabeva, K. (2008, December). The changing role of it departments in enterprise mashup environments. In International Conference on Service-Oriented Computing (pp. 148-154). Springer Berlin Heidelberg.

Kahler, H.: Supporting Collaborative Tailoring, Ph.D. Thesis, Roskilde University, Denmark, Roskilde, 2001.

Koschmider, A., Torres, V., & Pelechano, V. (2009, April). Elucidating the mashup hype: Definition, challenges, methodical guide and tools for mashups. In Proceedings of the 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web at WWW (pp. 1-9).

Kvale, S.: Interviews: An Introduction to Qualitative Research Interviewing. Sage Publications (1996).

Lal, R. (2008). Creating Vista Gadgets. Sams.

Le-Phuoc, D., Polleres, A., Hauswirth, M., Tummarello, G., & Morbidoni, C. (2009, April). Rapid prototyping of semantic mash-ups through semantic web pipes. In Proceedings of the 18th international conference on World wide web (pp. 581-590). ACM.

Mackay, W.E.: Users and customizable Software: A Co-Adaptive Phenomenon, PhD Thesis, MIT, Boston (MA), 1990.

MacLean, A., Carter, K., Lövstrand, L., Moran, T. (1990). User-tailorable Systems: Pressing the Is- sues with Buttons. In: SIGCHI Conference on Human Factors in Computing Systems (CHI '90), 175--182. ACM (1990).

Markus, M.L., Tanis, C. (2000): The Enterprise System Experience: From Adoption to Success. In: Zmud, R.W. (ed.): Framing the Domains of IT Research: Glimpsing the Future through the Past, pp. 173--207. Pinnaflex (2000)

Muller, M.J. (2003): Participatory Design: The third Space in HCI. In: The Human-Computer Interaction Handbook: Fundamentals, evolving Technologies and emerging Applications, pp. 1051--1068. Erlbaum (2003).

Nardi, B.: A Small Matter of Programming: Perspectives of End User Computing, MIT Press

Pane, J. F., Ratanamahatana, C. A. and Myers, B. A. (2001). Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems. International Journal of Human-Computer Studies, 54 (2), pp. 237-264.

Pipek, V. (2005): From tailoring to appropriation support: Negotiating groupware usage, PhD Thesis, University of Oulu

Pipek, V. and Wulf, V. (2009): Infrastructuring: Towards an integrated perspective on the design and use of information technology. Journal of the Association for Information Systems (JAIS), Special Issue on e-Infrastructures 10 (5), pp. 306-332.

Roth, A., Scheidl, S. (2006). End-User Development for Enterprise Resource Planning Systems. In: Informatik 2006, pp. 596--599. GI (2006).

Soriano, J.,Lizcano, D.,Cañas, M,/Reyes, M., Hierro, J. (2007): Foster Innovation in a Mashup-oriented Enterprise 2.0 Collaboration Environment. In: System and Information Sciences Notes, 1(1), S. 62-68.

Spahn, M., Dörner, C., Wulf, V. (2008a): End User Development: Approaches towards a flexible Software Design. In: 16th European Conference on Information Systems (ECIS 2008), pp. 303--314. CISC (2008).

Spahn, M., Dörner, C., Wulf, V. (2008b): End User Development of Information Artefacts: A Design Challenge for Enterprise Systems. In: 16th European Conference on Information Systems (ECIS 2008), pp. 482--493. CISC (2008)

Spahn, M., Kleb, J., Grimm, S., Scheidl, S. (2008c): Supporting Business Intelligence by Providing Ontology-based End-User Information Self-Service. In: 1st International Workshop on Ontology-supported Business Intelligence (OBI 2008). ACM (2008)

Spahn, M.; Wulf, V. (2009): End-User Development of Enterprise Widgets. In: de Ruyter, B.; Pipek, V.; Rosson, M. B.; Wulf, V. (eds.): Proceedings of the Second International Symposium on End User Development (IS-EUD 2009), Springer, LNCS, Heidelberg

Spahn, M. (2010). Flexibilisierung und Individualisierung des betrieblichen Informationsmanagements durch End-User Development, PhD Thesis,, University of Siegen, Department of Information Systems

Stevens, G.; Pipek, V.; Wulf, V. (2010): Appropriation Infrastructure: Mediating Appropriation and Production Work, in: Journal of Organizational and End User Computing (JOEUC), Vol. 22, Issue 2, 58-81

Tuchinda, R., Szekely, P., & Knoblock, C. A. (2008, January). Building mashups by example. In Proceedings of the 13th international conference on Intelligent user interfaces (pp. 139-148). ACM.

Wong, J. (2007, September). Marmite: Towards end-user programming for the web. In IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007) (pp. 270-271). IEEE.

Wulf, V.: “Let's see your Search-Tool!” - Collaborative use of Tailored Artifacts in Groupware. In: Proceedings of GROUP '99, ACM-Press, New York, 1999, pp. 50-60.

Wulf, V.; Rohde, M.: Towards an Integrated Organization and Technology Development. In: Proceedings of the Symposium on Designing Interactive Systems, 23. - 25.8.1995, Ann Arbor (Michigan), ACM-Press, New York 1995, pp. 55-64

Wulf, V., Rohde, M., Pipek, V., & Stevens, G. (2011, March). Engaging with practices: design case studies as a research framework in CSCW. In Proceedings of the ACM 2011 conference on Computer supported cooperative work (pp. 505-512). ACM.

Wulf, V.; Jarke, M.: The Economics of End User Development, in: Communications of the ACM (CACM), Vol. 47, No. 9, 2004, S. 41 - 42

Wulf, V., Pipek, V., Won, M. (2005): Component-based Tailorability: Towards highly flexible Software Applications. IJHCS 66, 1--22 (2008)

Yetim, F.; Dörner, C.; Pipek, V. (2010). Unterstützung von Softwareanpassungen in Kleinen und Mittelständischen Unternehmen: Wege zu einer Anpassungskultur. i-Com - Zeitschrift für interaktive und Kooperative Medien 9(2) 2010, S. 31-37.

Yetim, F.; Draxler, S.; Stevens, G.; Wulf, V. (2012). Fostering Continuous User Participation by Embedding a Communication Support Tool in User Interfaces. AIS Transactions on Human-Computer Interaction 4 (2), 152-167.

Yu, J., Benatallah, B., Casati, F., & Daniel, F. (2008). Understanding mashup development. IEEE Internet computing, 12(5), 44-52.