Lightweight Composition of Ad-Hoc Enterprise-Class Applications with Context-aware Enterprise Mashups

Florian Gilles[1], Volker Hoyer[1 2], Till Janner[1 2], and Katarina Stanoevska-Slabeva[2]

[1] SAP Research St. Gallen, Blumenbergplatz 9, 9000 St. Gallen, Switzerland
[2] University of St. Gallen, **=mcm***institute*, Blumenbergplatz 9, 9000 St. Gallen, Switzerland
{florian.gilles, volker.hoyer, till.janner}@sap.com, katarina.stanoevska@unisg.ch

Abstract. The huge demand for ad-hoc and situational enterprise-class applications led to a new kind of Web-based applications, known as enterprise mashups. End users from the business units with no programming skills are empowered to combine and reuse existing company internal and external resources within minutes to new value added applications. In order to handle the growing number of mashable components, we propose a context-aware concept for enterprise mashups that supports users to find relevant components according to their current situation and to compose them automatically. The designed context model which is structured in the three domains agent, computing and environment is implemented in the SAP Research RoofTop Marketplace prototype to demonstrate its applicability and business benefits.

Keywords: Enterprise Mashups, Context-Awareness, End-User Development

1 Introduction

A new trend for software development paradigm, known as enterprise mashups, has gained momentum in the recent years. They have the potential to bridge the gap between the automation transaction and peer production world. At the core of the mashup paradigm are two aspects: First, the empowerment of the end user to cover ad-hoc and long tail needs by reuse and combination of existing software artefacts; and second, broad involvement of users based on the peer production concept. According to Yorchai Benker, who coined the term peer production, *"it refers to production systems that depend on individual action that is self-selected and decentralized rather hierarchically assigned"* [1]. Thereby, the creative energy of a large number of people is used to react flexibly to continuous and dynamic changes of the business environment. Instead of long-winded software development processes, existing and new applications are enhanced with interfaces that are provided as user friendly building blocks. The explosive growth of these mashable components from company internal as well as external resources requires an efficient and agile organization [2]. Existing research efforts focus mostly on the technical composition of these components. However, end users from the business units with no or limited programming skills should be assisted by mashup systems in a passive manner to

retrieve relevant components and to compose them by connecting their input and output ports.

> *By means of a business scenario, we motivate the practical challenge: A sales manager (Max Meier) is usually responsible for maintaining knowledge of his company's products and liaising with customers. Although he has not much technical knowledge and because asking for support from the IT department would cost too much time, he decides to build his own enterprise mashup. Quite fast, he gets confident with the new platform and has success in building his first mashup which combines Customer Data with current Selling Data. Now, he wants to extend his first mashup, but he has no idea what other services would make sense in his context. One option would be to browse in the existing catalogue. But it's hard to find components which best meet his current business needs.*
>
> *Components which are used by his colleagues from the same department (e.g,. Peter Mustermann) or industry would be also relevant for our sales manager Max. Although adding new mashable components is quite easy, connecting components is the next problem occurring. Often the components have a lot of ports and it's difficult to decide how they can be connected with the correct inport and outport. From the viewpoint of Max, the platform should support him in providing relevant mashable components according to his current context. In addition, after selecting a component, the platform should automatically connect the components.*

The goal of this research paper is to fill this gap by designing a concept of how end users can be supported to navigate through the growing enterprise mashup ecosystem in order to adapt their individual working environment according to their context. The general research questions guiding this study are how to model the context space for enterprise mashups as well as how the context information can be organized.

The remainder of this study is organized as follows: After discussing related work in the area of enterprise mashups and context, section three presents the designed context approach. Section four demonstrates the implementation by means of the SAP Research RoofTop Marketplace prototype. A brief evalution and a summary close this present paper.

2. Background and Related Work

2.1. Enterprise Mashups – Definition and Characteristics

"An enterprise mashup is a Web-based resource that combines existing resources, be it content, data or application functionality, from more than one resource by empowering the actual end users to create individual information centric and situational applications" [4]. By simplifying concepts of Service-Oriented Architecture (SOA) and by enhancing them with the peer production philosophy, enterprise mashups focus generally on software integration on the user interface level instead of traditional application or data integration approaches. With the assistance

of a layer concept, the relevant terms can be structured in an Enterprise Mashup Stack which consists of the elements resource, widgets, and mashups.

Resources represent actual contents, data or application functionality. They are encapsulated via well-defined public interfaces (Application Programming Interfaces; i.e., WSDL, RSS, Atom, etc.) allowing the loosely coupling of existing Web-based resources – a major quality of SOA. These resources are created by traditional developers who are familiar with technical development concepts. The layer above contains **widgets** which provide graphical and simple user interaction mechanism abstracting from the underlying technical resources. The creation of the widgets is done by consultants or key users from the business units who understand the business requirements and know basic development concepts. Finally, the end users from the business units are empowered to combine and configure such visual widgets according to their individual needs, which results in a **mashup**. According to the motivated scenario in the first section, we focus on the mashup layer in this paper.

2.2. Context and Context Awareness

Context is an important utilized source of information in interactive computing. In order to use context efficiently and to build context-aware applications, the term context needs to be defined. The common aspect of existing definitions [5, 6] is that they all try to define the term by simply giving examples for context. Often it is associated with location, but [7] pointed out that context is more than location. Besides conditions and infrastructure, *"location is only one aspect of the physical environment"* [7]. However, the above mentioned two types of definitions are too specific and show a strong focus on location which makes them difficult to apply. In order to describe situation, state, surroundings, task, etc., a more general definition of context will be needed. A definition which fulfills all these aspects, is the definition given by [8]: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"*. This definition points out the important aspect, that *"context is any information that can be used to characterize the situation of an entity"*. Systems which make use of context in order to provide context-related information to the user are called context-aware [8]. This definition provides an easy and simple way to conclude whether an application is context-aware or not.

2.3 Enterprise Mashups and Context

In order to interpret context and visualize the definition of context, a flexible and extensible context model is required. As described in the definition of context, an entity is a person, place, or object. [3] stated out, that these context entities can be structured along the following three domains: *Agent Domain* (Who are you?), *Computing Domain* (What resources?), and *Envionment Domain* (Where are you?).

This kind of structure includes interrelation between entities that are within one domain and also interaction between the domains. Another important aspect is that

context information is dynamic by nature. [5] claim that the important aspects of context are where you are, who you are, and what resources are nearby. However, this description only includes location and identifies information. The aspect of time is completely missing. As a consequence of this, time needs to be included in our model, to characterize the situation in the past and future.

Concrete context entities inside those domains can now be expressed as 4-tuples [4]. A tuple is a sequence of specific values which are called the components of the tuple. The components of the 4-tuple result in "entity name", "feature", "value" and "time". Each context entity is identified by its unique name. For the different entities various features can be defined. The two feature categories are internal and external features. Internal features are characteristics inside the entity in its domain. External features describe the context information concerning the interaction of the entity with other entities, i.e. activities. The component *time* is mainly used to record history of context entities and to predict the future situation accordingly. The following list shows how this domain model is applied:

(MaxMeier, isColleagueOf, PeterMustermann)
(MaxMeier, isUsing, CustomerMashup, 2009/03/01)
(MaxMeier, isUsing, GlobalStartPage, 2009/03/01)

The three examples describe concrete context entities for the agent domain. The first example demonstrates an interaction of entities inside the domain. According to the definition of tuples for context representation, "MaxMeier" represents the entity name, "isColleagueOf" the feature, and "PeterMustermann" the value. Furthermore, this tuple has an internal feature, which means an interaction inside its domain. Because of that, the entity's value must be another entity, in this case "PeterMustermann". The item representing the time is missing in this tuple because it's not necessary to know the exact time of this activity. In contrast to the first example, the other two entities describe an interaction between domains. In the first case, it's the computing domain and in the second case it's the environment domain.

3. Context Aware Enterprise Mashups

3.1. Context Model

After obtaining the potential context entities, these entities need to be categorized to the three domains agent, environment, and computing. In the terminology of enterprise mashups, the *agent domain* represents all registered human users, who are eligible to use the enterprise mashup platform. This domain also includes business information like department, industry, country, and position. Due to the fact, that users are using different development platforms for their enterprise mashup applications, the idea of the *environment domain* is to express where the enterprise mashup is built or executed. Besides execution inside a Customer Relationship Management (CRM) page, the prototype can also be executed inside a sales order environment. What actual widgets are used to build enterprise mashups with a special business purpose is shown by the *computing domain*. It mainly includes the widgets

and the connection of the in- and outport. The following table summarizes these findings and the categorization of these context entities into the three context domains.

Agent Domain	
User	Person creating and executing enterprise mashups
Industry	Industry (i.e. Manufacturing)
Department	Specialized division of the large business organization where the user is working at (i.e. Sales Department)
Country	Country where the department is located at
Position	Status which the user can hpld in his department (i.e. Sales Manager)

Computing Domain	
Enterprise Mashup	Web-based application which consists of different connected Widgets
Widget	Visual representation of a resource
Port	Identifies the connection point between widgets
Business Purpose	Purpose that is desired of the enterprise mashup

Environment Domain	
Environment	Environment where the enterprise mashup is created or executed

Table 1: Context Entities of Enterprise Mashups

3.2. Context Organzation and Layers

In contrast to most other typical and traditional applications, which process the input to an according output based on a simple function, context-aware systems need also include and interpret context data to provide relevant information to the user. The process of interpreting raw context data to relevant context information can be described with three context layers. The bottom layer, the *Source Data* layer is sourced with raw data, like sensed or profiled data. This layer considers every bit of information the system gets and strictly excludes interpretation of data which is done in higher layers. Sensed data is obtained with sensors which provide the physical environment in which the enterprise mashup is executed for example. Profiled data can be for example user's preferences about a special situation. With this source of information, profiled or sensed data, it's possible to generate and build *Context Facts* related to the situation of the entities. Context facts are defined as something that actually exist and can be verified as true. It interprets the raw information from the Source Data layer to the first meaningful context information the first time. Examples for context facts are "Max Meier is using the Customer Data Mapping mashup", "The Customer Data Mapping mashup is executed in the CRM Start Page", or "Customer Data Mapping mashup uses the Customer Data Widget".

By describing context at a higher level than facts, new *situations* can be interpreted. Through ordering and weighting with special scores for each context fact,

confidence of situations can be achieved. Concerning to the context facts, an example for a situation would be: "Max Meier recommends the widget Google Maps as the best widget for mapping addresses on a map in the CRM Start Page". This example is based on interpretation of the context facts and can be used in order to express context characterized situations and to extract relevant information, like recommended widgets for users building enterprise mashups in the same CRM Start Page.

3.3. Context Facts

Most context information is highly dynamic by nature and makes it hard to create and apply adequate models. In order to provide an adequate model for further practical exploration, the Context Modeling Language (CML) is applied [9]. This modeling approach reformulates modeling concepts as extensions to the Object Role Language and assist designers with the task of exploring and specifying the context requirements of context aware applications, by providing a graphical notation for describing types of information.

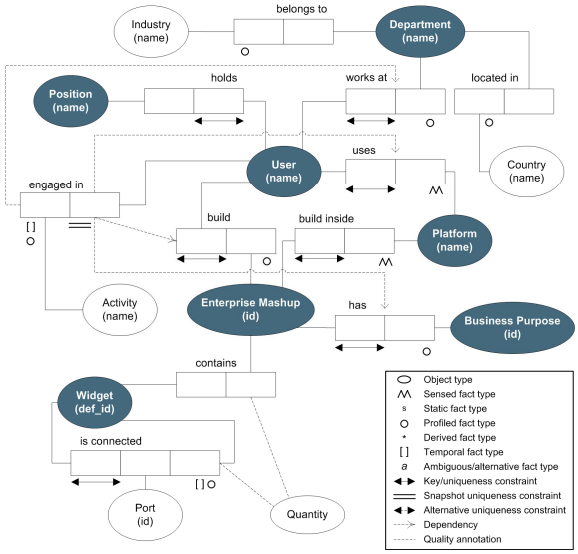

Figure 1: Enterprise Mashup Context Facts

The figure above models context information and types. All necessary context entities are represented by a solid ellipse. The relationship between these entities is defined by roles and an association which is symbolized by connected boxes, one box for each role. Besides trying to visualize the interaction and relationship between the different context entities from table 1, it also includes capturing capabilities for user activities in the form of a fact type that covers historical activities. Besides the historical point of view, user activities also include the used environment, the user's workplace and as well as the business purpose of the executed and built enterprise mashup. Due to the fact, that each role in the above figure also represents a context

fact in form of a relational model, it's possible to easily extract these context facts and use them for the base of the context-aware application.

3.4. Context Situations

Context situations are defined in terms of context facts and logical connectives (and, or, not, exists, forall). They can be unowned, owned by one entity, or owned by multiple entities. Context-aware systems in ubiquitous computing have the ability to determine and compute related context for situated behavior (also known as *Situated Computing*) [10]. According to the determined context facts in the previous section, a situation can be for example that the widget "Google Maps" is a recommended widget to display the customers address on a map. Without interpretation, the according context facts are only describing the quantity of how often the inport of the widget "Google Maps" is sourced with the address from a customer. But through interpretation, it's possible to say that if it's often used in this context, it could also be a recommended widget for other users building an enterprise mashup which includes customer data. The three context situations which are considered in this paper are:

1. **Recommendation of widgets by colleagues**
2. **Recommendation of widgets by environment**
3. **Recommendation of widgets by business purpose**

Each of the three recommendations is represented by a list and has a different focus on the used and processed entities. The first situation contains recommended widgets by colleagues and focuses on the agent domain. Besides the users social network, the users department, industry, country, and the position held by the users are mainly processed in order to generate the list. "Where" the enterprise mashup is built and executed is the main focus for the calculation on the second situation. It considers the different development platforms and mainly recommends different widgets for each environment. The last context situation focuses on the computing domain and considers all widgets which already have been used for the same business purpose.

For each list of the context situation, the widgets need to have a special ordering which should depend on the importance of the widget, as well as how many times the widget has been recently used in enterprise mashups. The widgets quantifier and context related importance is achieved by calculating a scoring function. The scoring considers different parameters, like the users position in the department, the department itself, its industry, country, and as well as the business purpose and the development environment currently used. The following pseudocode tries to state out the main aspect of the used scoring algorithm:

```
For each connection in Context Facts where Widget is involved
    Calculate percentage of criteria multiplied with according weight
```

The algorithm tries to consider all connections where the queried widget is involved. For each of those connections, a score is calculated based on the percentage of occurrence in a special context. After that, the list is sorted by that score, in order to have more context-related widgets listed at the top and less context-related widgets at the bottom.

4. Demonstration: SAP Research RoofTop Marketplace

The SAP Research RoofTop Marketplace prototype is an intuitive Web-based application based on AJAX that enables end users to create ad-hoc enterprise-class applications. To demonstrate the applicability of the context model, we have implemented our concept in the SAP Research RoofTop Marketplace prototype. In order to make use of context related information, the RoofTop Marketplace client is linked to a service, called "Lightweight Context Browsing". This service is responsible to gather and connect important context-related information from the repository which manages all widget or enterprise mashup information.

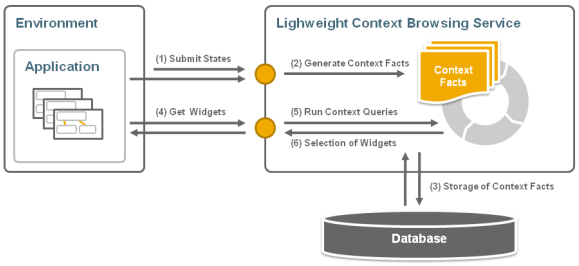

Figure 2: Lightweight Context Browsing

The Lightweight Context Browsing service considers already stated out important issues in context organization and context interpretation from raw source data to context relevant information. The following six steps describe how this service interacts with the client and the prototype:

Step 1: Submit States: Both, the application and the environment are submitting their states to the Lightweight Context Browsing service whenever an enterprise mashup is saved by the user. These states include information about the connection of widgets as well as information about the environment, industry, country, position, and business purpose of the enterprise mashup.

Step 2: Generate Context Facts: The submitted context-related information about the widget connection is then used to generate a context fact object.

Step 3: Storage of Context Facts: For persistent storage and later retrieval, the generated context fact is saved in a database.

Step 4: Get Widgets: Whenever the application needs a list of follow up widgets, it can submit requests to the according Lightweight Context Browsing service. Due to the fact, that the recommendation of widgets is split in three categories, three requests are sent to the service, after the user added the widget to the design view of the enterprise mashup platform.

Step 5: Run Context Queries: After submitting these requests, the context queries are executed on the Context Facts in order to extract recommended and relevant widgets. Extraction is based on relevance of the widget to the current context. To list most important widgets at the top of each list, the ordering of widgets is done by applying the previously described scoring algorithm.

Step 6: Selection of Widgets: At the end, the selection of recommended widgets is passed back to the application where they are presented to the user and from where the user can decide, what widget he wants to add for extending his enterprise mashup.

5. Evaluation and Discussion

As motivated in the introduction, the sales manager in our scenario will usually be responsible for maintaining knowledge of his company's products and liaising with customers. Before starting to build the first enterprise mashup, he has to maintain his user profile which includes his business details like the position, the industry, the country, or the department he is currently working at.

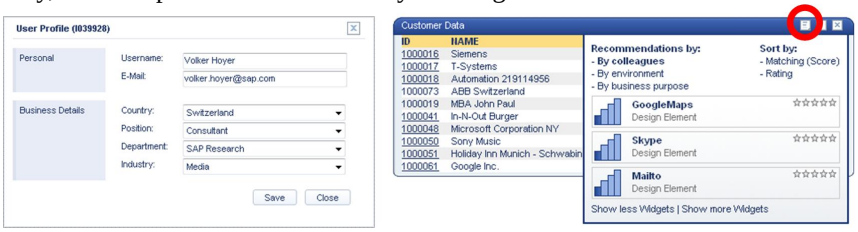

Figure 3: User Context and Lightweight Context Browsing Menu

Quite fast, the sales manager added the Customer Data widget to the design view, to get a list of all customers of his company. However, this step was quite easy, but to extend his enterprise mashup he has not much and detailed knowledge about other widgets which would make sense for his context. Because of that, the sales manager clicks on a little button at the top of the Customer Data widget. This button is available at all widgets. The upcoming popup shows lists of recommended widgets from which the user can choose may it “recommendation by colleagues”, “recommendation by environment”, or “recommendation by business purpose”.

The sales manager is interested in widgets which have been often used by his colleagues in the same department (“recommendation by colleagues”). The presented list of recommended widgets includes “Google News”, “Flickr Photos”, and “Google Maps”. However, these widgets can also be found in the widget catalogue, where they are categorized in different categories. The great benefit of this small presented list is that if the user decides for one of the widgets, it is automatically added to the enterprise mashups. If the user had decided to add the widget from the widget catalogue, he then had the whole responsibility in connecting the widget ports which can also be a nightmare according to the number of the technically focused ports. Instead, if the user decides to add a widget from the lightweight context browsing service presented above, the widget is added to the editor and afterwards connected automatically. This saves time dramatically and assists the end user in a passive manner without dealing of finding and connecting widgets.

The sales manager decided to add the “Google Maps” widget. With one click, the widget is automatically added. Also the outport of the customer data widget is automatically connected to the inport of the “Google Maps” widget. In this case, the “Google Maps” widget is sourced with the address from a customer. As a result, the customers address is automatically displayed on “Google Maps”.

6. Conclusion and Outlook

The aim of this paper was the design of a model for context-aware enterprise mashups. After defining the main terms related to enterprise mashups and context, we presented a context domain model for enterprise mashups and a layer concept of how the context can be organized. An implementation of the concept in the SAP Research RoofTop Marketplace prototype demonstrated its applicability. By means of a typical business scenario, we evaluated and discussed the first benefits of the solution.

What is still missing is a broader application of the context model in various application domains. Further research will deal with a user evaluation in a field study to measure the benefits. In addition, there is no privacy enforcement in the current concept. The base of the described concept is the mass number of enterprise mashups which has been saved by the users and later processed. However, while saving enterprise mashups the end users agree that their sensitive data is stored and later be used to support others.

References

[1] Y. Benkler, *The Wealth of Networks. How Social Production Transforms Markets and Freedom*, Yale University Press, New Haven and London, 2006.

[2] S. Yu, *"Innovation in the Programmable Web: Characterizing the Mashup Ecosystem"*, Proceedings of the 2nd International Workshop on Web APIs and Service Mashups, Sydney, Australia, 2008.

[3] J.-Z Sun, J. Sauvola. *"Towards a conceptual model for context-aware adaptive services"*. Proceedings of *4th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 90-41, Chengdu, China, 2003.

[4] V. Hoyer and K. Stanoesvka-Slabeva, "Towards a Reference Model for grassroots Enterprise Mashup Environments". In *Proceedings of the 17th European Conference on Information Systems*, Verona, Italy, 2009.

[5] B. N. Schilit, M. M. Theimer. *"Disseminating Active Map Information to Mobile Hosts"*. IEEE Network, 8(5):22-32, 1994.

[6] N. Ryan, J. Pascoe, D. Morse. "Enhanced Reality Fieldwork: the Context Aware Archaeological Assistant". In *Archaeology in the Age of the Internet: CAA 97: Computer Applications and Quantitative Methods in Archaeology*: Proceedings of the 25th Anniversary Conference, University of Birmingham, April 1997. British Archaeological Reports.

[7] A. Schmidt, M. Beigl, H. Gellersen. *"There is more to Context than Location"*, Computers & Graphics, 23(6):893-901, 1999.

[8] A. Dey, G. Abowd. *"Towards a better understanding of context and context-awareness"*. Technical report, GIT-GVU-99-22, 1999.

[9] K. Henricksen, J. Indulska. *"A Software Engineering Framework for Context-Aware Pervasive Computing"*. In Proc. of the 2nd IEEE Conference on Pervasive Computing and Communications (Percom2004), Orlando USA, 67-77, 2004.

[10] Hull, R.; Neaves, P.; Bedford-Roberts, J., "Towards situated computing," *Wearable Computers, 1997. Digest of Papers., First International Symposium on* , 146-153, 1999.