Building a Person-Centric Mashup System. CommunityMashup: A Service Oriented Approach.

Peter Lachenmaier[1], Florian Ott[1],

[1] Bundeswehr University Munich, Cooperation Systems Center Munich,
Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
{Peter.Lachenmaier, Florian.Ott}@kooperationssysteme.de

Abstract. Based on the success of the Web 2.0, today's IT systems are continuously moving from a solely information-centric data perspective to a more person-centric model and are thereby becoming more social. In this paper we discuss the challenges within the redesign of established data models resulting from that shift. Our aim is to derive requirements for a flexible social (person-centric) data integration layer which enables us to aggregate data from several distributed services while retaining the assignment to the individual (personal) identities. In addition to these theoretical considerations we describe how a "CommunityMashup" could be developed, easily maintained and adapted to frequently changing APIs with a service-oriented approach.

Keywords: CSCW, Social Software, Mashup, CommunityMashup, Service Oriented Architecture, Model Driven Development

1 Motivation

A study of different commercial and open-source Enterprise 2.0 tools in 2009 showed that only two out of the seven systems supported tracking of user activities. But all of them enabled their users to trace content changes [1]. For example one of the most commonly used systems in that survey, Atlassian Confluence, did not support tracking users or their activities in 2008 (year of survey), but has recently (2010) switched to a more person-centric approach. Another study analyzed seven tools in 2006 and no more than two of them maintained "user centered" functionalities [2]. Compared to former CSCW[1] research (e.g. [3]) with the success of public Social Networking Services like e.g. Facebook particularly personal information has become more and more important. Atlassian and other global players like Microsoft have adjusted their strategy between 2007 and 2010 to a better support of community features [4]. This development can be put down to the peer-to-peer principle of the Web 2.0 where sharing and collaboration are the most important activities [5].

Beside increased significance of personal data today's IT services have become more modular and more open (in terms of accessibility to data) during the last decade. Hence we are facing a variety of different data sources and at the same time the wish to access (identical) data in different ways as for example with desktop applications,

[1] CSCW: Computer Supported Cooperative Work

websites or mobile devices. Already in 2005 Beale showed different systems supporting social interaction with smartphones and mobile consumption of content [6]. By now most Internet service platforms offer interfaces to access their data. But there is still no "standardized" way to access all data with all available devices.

Although there are other approaches dealing with data models for Social Software, e.g. the Semantically Interlinked Online Communities [7], most of the them still focus on linking content and are thereby not person-centric enough to fulfill all requirements of Social Software (e.g. handling awareness information). The requirements of person centrism, device independent access, modularity and easy adaptability are still not completely satisfied. So we derive the demand for a more adequate data model that can address the needs of a flexible integration service for Social Software.

As activities of other people / groups are becoming more important and the corresponding data will need to be consumable in various contexts with different devices we are using three representative application scenarios where the integration of person-centric data plays an important role. Adapted from these scenarios we will derive specific challenges and requirements for a data model and a technical solution.

Scenario 1: Elderly Interaction & Service Assistant (elisa)
For enabling elderly people to access awareness information from Social Networks without being a direct member of every online service, we currently construct a mobile application showing aggregated awareness information. The displayed information consists of data provided by several people of interest in different social services. In this application scenario information has to be delivered from several people through different services and networks to one single person adapted to his or her individual needs.

Scenario 2: CommunityMirrors
CommunityMirrors are large screens presenting information that is otherwise hidden in IT systems in (semi-) public places, like coffee corners, lobbies or beside the elevator as described e.g. in [8]. In this application scenario information from different sources is shown in an aggregated and unified way on the large screens. Data created by many people has to be delivered to many people without knowing their individual preferences in advance. The aggregation is driven by an organizational context.

Scenario 3: Decentral Federated Research Database (DFRD)
As third application scenario we use the decentral federated research database (DFRD) presented in [9]. Researchers are able to maintain their articles and projects in services of their choice and present them on several web sites by using aggregation mechanisms of the DFRD. Possible data targets are for example a private web site or an aggregated version together with works of other researchers on the university portal as well as presentations filtered for different research groups. In this setting data from the same origin has to be presented with different aggregation levels in various places.

Table 1. Scenario overview

Scenario	elisa	CommunityMirrors	DFRD
Goal	aggregation, filtering, personalization	aggregation, offline setting with synchronization	aggregation, individual / context-related presentation
Data sources	different online Social Networking Services (SNS)	Enterprise 2.0 services like wikis, blogs, microblogs or SNS	individual research services and project portals, Research SNS
Data targets	personalized presentation with mobile application	(semi-) public presentation in different social places	online presentations, e.g. personal or organizational websites
Data type	mainly awareness streams and events	organization specific content with corresponding awareness information	static information like research papers or projects, corresponding awareness information
Target device	mobile device, e. g. Tablet	rich client, large interactive screen	browser (web applications)

Table 1 shows an overview of the application scenarios. Based on this first overview we summarize the challenges during the conceptual design (Chapter 2) and then give a brief overview of the technical solution (Chapter 3). Chapter 4 outlines the mashup possibilities of the technical base. The paper is completed by a conclusion and an outlook to the further development plans of the "CommunityMashup" (Chapter 5).

2 Challenges

The term mashup gained more and more attention in the last years and is used in different meanings. An overview of different definitions is given in [10]. We are using the term mashup as a technical solution that combines data from more than one source, enriches it and provides a unified version for further usage. Based on the different application scenarios introduced in the previous chapter we derived the following four main challenges.

Heterogeneity of services
When two or more people are using different online services the heterogeneity of data formats and interfaces makes the data delivery from one person to another very difficult. Main challenges are the handling of differing data models, the availability of various formats and the use of frequently changing interfaces. In addition to that several authentication and authorization methods as well as questions concerning licenses and laws about how data may be reused or stored have to be considered.

Heterogeneity of clients
As ubiquitous devices in all thinkable shapes and colors have become more and more important during the last decade we are facing heterogeneity on the client side, too. Derived from the application scenarios (see Table 1) we distinguish between three different client classes that can consume the aggregated data. The main differences between these classes are the availability of computing power, memory and network connections.

1. Web applications are the most commonly discussed class in the context of mashup development. They are typically executed on a web server and can be accessed with web browsers.
2. Rich client applications execute most of the required calculations on the client device and therefore require a high amount of computing power and memory.
3. Applications for mobile devices have to deal with much less computing power and memory. Most of the available network connections have unreliable bandwidth.

While mobile and web applications need permanent network connections for data access there are certain settings for rich client applications without the availability of a continuous network connection. But there's still the need of permanent and fast access to required data. The data aggregation itself is independent of the application class. But the different classes require special interfaces to access the data.

Privacy and data protection

In the Web 2.0 era every user provides a huge amount of personal data. Thus, there is a demand for preventing third persons from seeing private information. For that reason most Social Networking Services offer the possibility to grant access only to specified persons or explicitly to external services. These mechanisms must not be bypassed by mashup solutions. So we are facing the requirement to integrate authentication and authorization mechanisms on the mashup side.

Aggregation of data

Another challenge is the combination of data coming from different sources. With the person-centric approach this mainly requires the decision if two profiles belong to the same human being. Social Software in general relies on users being able to manage their personal data by themselves, for example by providing links to their various profiles. This process can be supported but must not be restricted by any technological automatism, since it might not be sufficiently transparent for the user.

3 Technical Solution

To meet the challenges described above the most important requirement besides a person-centric approach is a highly configurable mashup system built with flexible service components reusing existing frameworks. Model driven development for components affected by data model changes is a major claim.

We propose an object-oriented solution to enable direct high-level access to and modification of aggregated data. The introduced application scenarios and the existence of flexible tool chains for application development are the main reasons for this approach. In addition to that generated application frameworks allowing easy development of similar community applications in future versions.

3.1 Person-centric data model

Figure 1 shows the core elements of the current version of the CommunityMashup data model. The main objects person, content and organization are derived from

existing models like SIOC[2] [7] or FOAF[3] [11]. The central element is the person, which can be grouped in organizations and can author or contribute to content. Organizations and contents can be modeled hierarchically (parent relation).

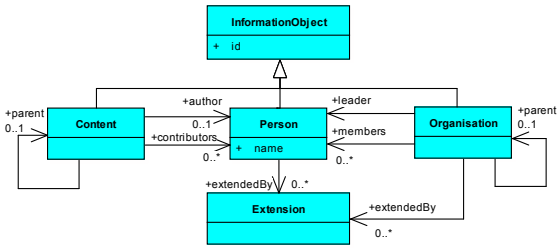

Fig. 1. Core elements of the CommunityMashup data model

In contrast to other models, one important design principle is to include less meta data but instead allow tagging and categorization of information. By this approach a lot of meta data coming from different data sources is transformed to categories and tags. Only identity preserving characteristics like e.g. the name of a person are modeled directly as attributes of the core objects.

Extensions for easier combination
A person is represented by one single object independently of how often he or she appears within the different sources. This means that persons as well as organizations only have one consolidated identity. With every new data source the aggregated information will just be extended. An extension object for the information coming from an additional source will be created, tagged and referenced in the dataset. This allows keeping track of the origin and reintegrating changes back into the source. For example specially tagged extension objects can easily handle complex relations between persons.

Model driven development enables data model evolution
There is no way to determine all facets of a data model in advance so that it fits all future needs. Therefore we need a way to make the adaption to model changes as easy as possible, especially without the need of manual changes to the applications based on the model. Using a continuous tool chain offers tracking of data model changes and the regeneration of application code as well as migration of existing data to the newer version. As we are using a central data model and transformation rules for data from external services, these model transformations can be adapted (semi-) automatically.

3.2 Service Oriented Approach

The CommunityMashup consists of smaller independent modules. Each of these modules acts as a service component and can be combined with others to a complex mashup system. This allows flexible integration of existing services and the distribution over different physical machines.

---

[2] SIOC: Semantically Interlinked Online Community
[3] FOAF: Friend of a Friend

The meta model shown in Figure 2 describes the possible system compositions. Basically a mashup system is composed of several sources. The mapping characterizes how the source data is assigned to the target data of the model. Every source has a configuration containing meta data, e. g. authentication parameters. Additionally there can be an explicit adapter used to transform the data from the external service to a representation according to the internal data mode. The data provided by a source is optionally filtered by a chain of filters, e. g. for privacy reasons. Furthermore every mashup itself can act as a source and thereby be reused in more complex compositions. In addition to that the mashup offers the possibility to persist data from every source as well as the internally "mashed-up" data in either a file or a database. Besides caching which is necessary in offline scenarios this can be used to satisfy performance or availability requirements.

Finally every source and every mashup will be an independent service component that can be deployed, enabled and disabled separately. This aspect is important in order to be able to create distributed mashup configurations. Also a direct execution of mashup components on the client device is possible. For example there could be a mashup component running along with its persistency component as a local cache. With this approach local components can be directly integrated in rich client applications. This enables high level access to data objects without the need to deal with data exchange. Together with a local persistency this allows seamless switching between offline and online data usage.

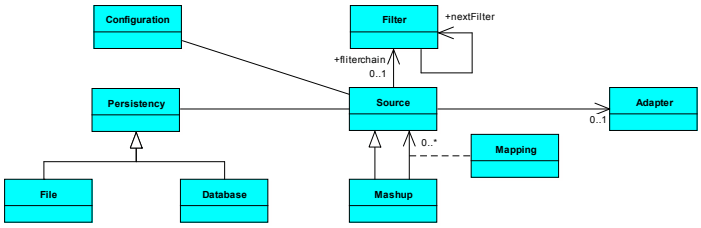

Fig. 2. CommunityMashup meta model

A graphical editor for the creation and change of mashup configurations based on the meta model is planned. Most parts of this editor can be generated with existing tools. Future versions of the editor should be able to interpret these configurations for an automatic deployment and execution of the individual mashup and all dependent components.

3.3 Solution Technologies

For the creation of the data model and the meta model we used the Eclipse Modeling Framework (EMF) [12]. EMF provides a tool chain for the generation of application code and a runtime engine containing persistency and serialization components as well as an integrated event mechanism that allows tracing of data changes. In addition to that there are many useful extensions for EMF like e.g. a tool named COPE[4] [13]

[4] COPE: Coupled Evolution

which helps to track changes of the data model and to migrate existing data to the newest version. This makes it possible to have a continuous tool chain fulfilling the model driven development approach. Because of the multitude of reusable service components we used the OSGi Service Platform with the Eclipse Equinox implementation [14] as service framework.

4 Mashup Possibilities

In the previous chapter we introduced the technical solution of the CommunityMashup. We are continuing with a brief overview of the possibilities of concrete applications based on this technical base.

One of the most important aspects is the discovery of communities that were formerly hidden and distributed over several networks. By combining data from the different networks connections of people will be visible, e. g. knowing the same person or liking the same content. Furthermore the CommunityMashup enables access to an aggregated profile of a person that contains all distributed information. The data model allows keeping track of the data origin, so that changes in the profile can be passed to the original source or delivered to all other profiles.

Many of the content-centric services, e. g. wikis, don't support person-centric awareness streams. With the integration of these services into the CommunityMashup there will be the possibility to automatically create activity streams or similar awareness information. This information can be used in applications based on the CommunityMashup and can be delivered to other source services integrated in the same mashup system. For example a new Wikipedia article can be automatically published as a status update in the Facebook activity stream of the author.

From a more technical point of view, the proposed CommunityMashup solution facilitates the development of applications based on this technical base. The implementation of the three introduced application scenarios (Chapter 1) can be managed without dealing with data integration questions by using a high level API and the integration of CommunityMashup components.

5 Conclusion and Outlook

Based on the motivation for a person-centric data model for Social Software we outlined challenges within the design and development of a flexible mashup solution and presented a person-centric data model. The use of model driven development can help to facilitate model changes. Application source code can be adapted without manual interaction. Furthermore existing data can be migrated without data loss. The CommunityMashup meta model shows the different components of the mashup solution and denotes the possible configuration options. Concerning the architecture of the CommunityMashup we gave a short overview and introduced the technologies that could be used for its implementation.

As stated in Chapter 2 we consider three target application classes: web applications, rich client applications and applications for mobile devices. Currently we are

working on the three usage scenarios of the CommunityMashup described in the motivation:

1. A solution for “best agers” (elderly people) to access awareness information from Social Networking Services with mobile devices.
2. Large semi-public wall-sized screens, our so-called “CommunityMirrors” as a rich client application.
3. A decentral federated research database as a web application with the possibility to present aggregated and filtered data on different web sites.

In all of these scenarios we try to enhance the presented data model and the mashup architecture itself in an iterative incremental way in order to be able to make empirical statements about e.g. the usefulness and efficiency in future examinations.

References

1. Büchner, T., Matthes, F., Neubert, C.: A concept and service based analysis of commercial and open source enterprise 2.0 tools. International Conference on Knowledge Management and Information Sharing (2009)
2. Rama, J., Bishop, J.: A survey and comparison of CSCW groupware applications. In: Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries - SAICSIT ’06, pp. 198--205, ACM Press (2006)
3. Rodden, T.: A survey of CSCW systems. In: Interacting with Computers 3, 3, pp. 319--353 (1991)
4. Harbridge, R.: SharePoint 2007 vs. SharePoint 2010 Comparison, http://www.rharbridge.com/?page_id=103
5. Ganesh, J., Padmanabhuni, S.: Web 2.0: conceptual framework and research directions. In: Proceedings of the 13th Americas Conference on Information Systems (AMCIS 2007), pp. 198--205 (2007)
6. Beale, R.: Supporting social interaction with smart phones. In: IEEE Pervasive Computing 4, 2, pp. 35--41 (2005)
7. Breslin, J., Decker, S.: SIOC: An approach to connect web-based communities. In: International Journal of Web Based Communities (IJWBC) 2, 2, pp. 133--142 (2006)
8. Koch, M., Ott, F., Richter, A.: Community Mirrors - Using Public Shared Displays to Move Information "Out of the Box". In: Supplementary Proceedings of the 11th European Conference on Computer Supported Cooperative Work (ECSCW), pp. 17--18 (2009)
9. Lachenmaier, P., Koch, M., Richter, A.: Supporting Open Research by making research activities visible, Proceedings of the Workshop on Academia 2.0, 11th European Conference on Computer-Supported Collaborative Work (ECSCW) (2009)
10 Hoyer, V., Fischer, M.: Market Overview of Enterprise Mashup Tools, Lecture Notes in Computer Science, Service-Oriented Computing – ICSOC 2008, vol. 5364, pp. 708--721 (2008)
11. The Friend of a Friend (FOAF) project, http://www.foaf-project.org/
12. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework. Addison-Wesley Professional (2008)
13. Herrmannsdoerfer, M., Benz, S., Juergens, E.: COPE-automating coupled evolution of metamodels and models. In: Lecture Notes in Computer Science, ECOOP 2009 – Object-Oriented Programming 5653/2009, pp. 52--76 (2009)
14. Wütherich, G., Nils, H., Berd, K., Lübken, M.: Die OSGi Service Platform. Dpunkt.verlag GmbH (2008)