Architecture for Integrating Desktop and Web 2.0 Data Management

Stefania Leone, Michael Grossniklaus, Moira C. Norrie
Institute for Information Systems
ETH Zurich
CH-8092 Zurich, Switzerland
{leone, grossniklaus, norrie}@inf.ethz.ch

Abstract

A new form of personal information fragmentation is arising due to the rapid growth in Web 2.0 applications and their use for the management of data typically associated with desktop applications. We propose a data management architecture that allows data to be shared between desktop and Web 2.0 applications. The architecture supports a separation of concerns between the management of personal data and its publication on the Web to social networks.

1 Introduction

Web 2.0 is the term often used to describe the Web's evolution from a hypertext publishing system [2] into a platform of participation and collaboration [18]. In Web 2.0, content is not only delivered to users, but users actively participate by augmenting and creating content. As a consequence, content is no longer created locally and then published to the Web, but rather managed entirely online. In essence, Web 2.0, together with Rich Internet Applications (RIA), are taking over some of the functionality of traditional desktop applications.

In particular, several Web 2.0 sites have emerged that allow personal information to be managed collaboratively over the Web. Personal data can be of different types and can be categorised into two main groups. First, there is data which formerly has been managed locally by desktop applications, such as contacts, documents or pictures. Second, there is a lot of personal data which has only emerged because of the Web, such as bookmarks or any kind of metadata associated with media content. Web 2.0 applications that manage such personal information include sites such as Flickr and YouTube for images and videos, Blogs for writing diaries or travel journals, Google Documents for managing and sharing documents or del.icio.us for Web bookmarks. It is important to note that while some of these sites focus on a user's private life, others are closely related to work activities. Thus, friendships might be managed by Facebook, while professional networks and contact information may be managed by sites such as LinkedIn or Xing.

Clearly, this development in terms of how the Web is used for personal information management (PIM), also raises several technical challenges with regard to data management. While the Web is an ideal platform for collaboration and participation, Web applications generally cannot compete with desktop applications in terms of complexity and integration into the local working environment. Further, while a few approaches have been proposed to enable users to work with Web 2.0 applications offline, currently desktop applications still cope better with this requirement. Therefore, it is not to be expected that Web 2.0 applications will replace desktop applications entirely, but rather that the two kinds of applications will be used for different tasks and modes of working. However, they may share data which means that, ideally, there should be some way of seamlessly managing data across desktop applications and Web 2.0 applications.

The distribution of information across desktop applications and Web 2.0 applications introduces further forms of information fragmentation already considered to be one of the main issues of PIM with respect to desktop applications. We believe that new forms of data management architectures are required that can provide an integrated approach to data management for desktop and Web 2.0 applications with a clear separation of concerns between the management of personal data and its publication on the Web to social networks. One of the main technical requirements is to provide ways in which data can be synchronised between desktop applications and Web 2.0 applications. However, such an approach should also incorporate concepts that have proven useful in Web 2.0 applications. One such feature that can be witnessed on sites such as Facebook is the ability to extend these platforms based on plug-ins or modules. In order to successfully manage data for Web 2.0, we believe that the data management architecture should reflect this notion of components.

In this paper, we first motivate the need for a data management architecture that bridges the gap between the desktop and Web 2.0 and then present our approach that is intended for the development of data-centric applications. We begin in Section 2 with a background discussion of the state of the art in developing Web 2.0 applications as well as current solutions in the domain of personal information management. Section 3 provides an in-depth analysis of the challenges and requirements that must be met by any integrated approach to data management for Web 2.0 applications. In Section 4, we then propose an architecture that addresses the requirements defined in the previous section. Preliminary results and future work is discussed in Section 5. Finally, concluding remarks are given in Section 6.

2 Background

In this section, we first review the data management features of Web 2.0 applications and then go on to describe approaches that have been adopted in the PIM research community to address the problems of information fragmentation on the desktop.

As outlined in [18], Web 2.0 is a term used to refer to a category of Web-based applications that have become extremely popular in recent years by defining a set of features that they have in common. These features can be summarised in terms of two main trends. On the one hand, Web 2.0 applications are characterised by rich and responsive user interfaces, and, on the other hand, the user becomes a participant who not only consumes content, but actively takes part in content creation.

There has been a lot of attention given to the design of application development frameworks for the implementation of Rich Internet Applications in both industry e.g. [23, 22, 17] and academic environments e.g. [3, 21]. However, to date, little attention has been paid to data management issues in terms of both detailed studies on the use of Web 2.0 applications for PIM and also tools and infrastructure for managing and sharing user-generated content.

An additional feature of many Web 2.0 platforms and applications such as Facebook[1] and Flickr is the provision of an application development interface (API) which allows an application developer to access and retrieve platform data as well as to build applications for that platform. The Google Open Social API[2] is an effort to standardise interfaces to social networking applications by defining a platform independent interface which can be implemented by any social networking application. It can be assumed that in the near future more and more web platforms will export their data and functionality via such an API.

[1]http://developers.facebook.com/documentation.php

[2]http://code.google.com/apis/opensocial/

Although there is a lack of detailed studies on how Web 2.0 applications are being used to manage personal data, general surveys such as [14] show that social networking sites such as Facebook are heavily used for managing and sharing personal data such as photos in addition to keeping in touch with friends. It should be emphasised that personal data includes work-related data and not just data related to one's private life. For example, increasingly researchers are managing personal data related to their work such as contacts, bookmarks and information about their publications on the Web.

This shift from desktop data management to Web 2.0 data management raises a set of new issues for personal data management. For example, it has become popular for users to publish collections of photos on various Web sites such as Flickr and Facebook, in order that they can share this data with friends. But, typically, users also keep local copies of this data. Sometimes they even publish different or the same photos to different Web 2.0 sites in order that they can share it with different communities. As a result, the data objects may be replicated and users need to manually keep track of which photos and which versions of these photos have been published to particular sites. Sometimes users will make changes to the photos already published in which case some form of synchronisation between the published and the desktop data is required.

Other types of personal data may be produced directly on-line. For example, Blog posts or bookmark collections may be created using a Web 2.0 application which means that users lose control over that data and they cannot access it offline. Also, one of the features of Web 2.0 is that users author content collaboratively which means that other users may tag photos, add comments to posted articles etc. And rather than each user having to manually maintain an entire address book, each user only has to maintain their own contact details and they are automatically shared by other users.

As a result, nowadays, personal information is not just fragmented across desktop applications but also between desktop applications and one or more Web 2.0 applications. A user typically maintains several social networking profiles, for example one for friends and family on Facebook and a Xing or LinkedIn profile for managing business contacts. Research efforts have already been made to allow the aggregation and integration of social networking information from various Web 2.0 applications [13]. In [11] personal user data scattered over various Web 2.0 applications is integrated by providing an infrastructure for integrating web data services as well as a set of web-based tools that support a unified view and file system-like organization of a user's data. However, both approaches only consider data on the Web ignoring the fact that personal data is often stored locally and duplicated on various sites.

The problem of information fragmentation has already received a lot of attention in the PIM research community where they address the problem of fragmentation caused by the fact that different desktop applications manage personal data as well as issues arising due to the hierarchical folder structure, as for example described in [16, 1].

Within the PIM community, two approaches can be identified for solving the problem of personal data being managed by different desktop applications. One approach is to use dedicated search engines for personal data to allow users to find data regardless of the application used to manage it. For example, "Stuff I've seen" [7] as well as commercial desktop search engines such as Windows Desktop Search, Google Desktop Search and Apple Spotlight all help to overcome the problems of data fragmentation. To do so, these approaches create indexes and offer full-text search over all user data or provide support for query refinement to improve search over personal data. An alternative approach is to introduce additional data structures that help to associate and integrate data of different types and from different sources. Some of these data integration systems use predefined PIM domain models [12, 6, 15, 20] while others work according to a "no-schema" or "schema-later" approach [19, 10]. Regardless of the approach, all solutions integrate all of a user's data and provide support for associating data of various types and from various sources, mostly following the vision of trails proposed in [4].

PIM research has not addressed the issue of fragmentation between the desktop and the Web. While most of the systems support the inclusion of Web data sources [6, 15, 12], the problem of data fragmentation caused by the use of online platforms to create and manage data is not addressed directly. Even though newer PIM approaches are drifting away from a fixed PIM schema definition to "no-schema" or "schema-later" approaches as proposed in [9], support for on-the-fly data integration as supported by [19, 5] is not what is required to deal with the problem of data fragmentation outlined above. Further, approaches for solving the problem of traditional data fragmentation on the desktop do not deal with data duplicates, which are very common in a setting where data is managed both locally and on the Web.

Last but not least we note that systems where personal data is exclusively stored on the Web are emerging. Commercial systems that are freely available to users such as the Google Documents suite and Windows Live offer a set of applications for the on-line creation and management of personal information. The main drawback of these systems is the fact that they make a user highly dependent on the service provider and they lose all the control over their private data.

In the rest of the paper, we propose a general data management architecture intended to address the problem of data fragmentation and support the desire for users to be able to keep local copies of their data and publish it in flexible ways on different Web 2.0 sites.

3 Data Management Challenges

As already mentioned, a set of data management challenges are raised by the increased use of Web 2.0 applications as data management solutions. The fragmentation of a user's personal data is caused by three main factors.

First, data that is created using a desktop application is often published to one or more Web 2.0 applications. As a result, it is common for data to be replicated across two or more Web 2.0 sites.

Second, users have to register on each site and user profile data is replicated across sites. To address this particular fragmentation problem, schemes have been proposed where user profile data can be managed in a distributed fashion and used for various applications [8]. However, to date, no one has addressed the more general problem of various forms of personal data being replicated across Web 2.0 applications.

Third, personal data can be generated purely on the Web, leaving the user without any control over that data and fully dependent on the web service provider. As a result, sometimes users will explicitly create local copies of that data to ensure that they can have offline access and a local archive.

In the setting of Web 2.0 data management, means for preventing or handling data fragmentation, as well as for dealing with replication, are very important. Further, users should be able to seamlessly manage data across desktop and Web 2.0 applications. This means that users should be able to manage all of their data in the same way regardless of if and how it is published on the Web. To achieve that there needs to be a clear separation of concerns between *data management* and *data sharing*. Finally, users need to be provided with an infrastructure and tools that can allow them to specify easily which data should be published where and the forms of data synchronisation that should exist between desktop applications and Web 2.0 applications.

This leads to the following list of data management requirements and challenges.

- The data management architecture needs to provide a uniform view of all of a user's personal data regardless of whether it is managed by a desktop application or a Web 2.0 application.
- A user needs to be supported in solving the data fragmentation issue in a controlled and transparent way by providing means to define where data should be published, how the data should be mapped and where the original data resides.
- The data management architecture has to address the issue of having duplicates and provide means for ver-

sion control and synchronisation between the different copies of the same object.

- The data management architecture should be general enough to be applicable to any application domain, letting the application developer choose the appropriate domain model.
- Since the data life cycle of any type of data tends to be rather long, schema evolution should be supported.

When designing a Web 2.0 data management architecture, we believe that these requirements and challenges have to be considered in order to provide the user with a satisfactory data management solution.

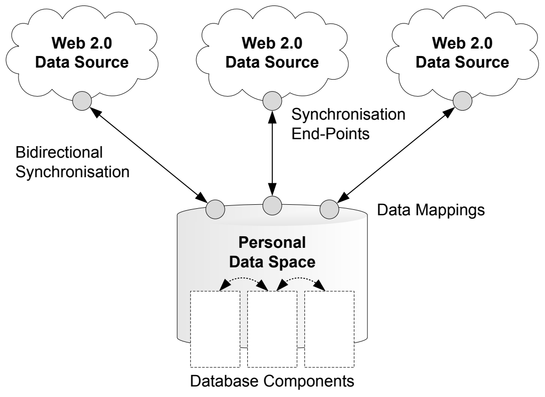

Figure 1. Architecture.

4 Proposed Architecture

The architecture that we propose for integrating desktop and Web 2.0 data management is depicted in Figure 1. The data management architecture consists of three main components. First, there is a personal data space component, where all of a user's data resides. Second, there is an extensible set of Web data sources which represent the Web 2.0 platforms and applications to which the user can publish and synchronise personal data. Third, there is the publishing and synchronisation component, where data publishing and synchronisation strategies can be configured. We will now describe each of these three components in more detail.

4.1 Personal Data Space

The personal data space is the heart of the system, where all user data is stored and which is mainly used by the user to access and manipulate their data. Generally, this would be local to the user to enable them to have full control over all of their personal data and avoid any dependency on specific Web 2.0 applications. Instead of Web 2.0 applications having the sole responsibility for managing personal data, it rather becomes a question of the user controlling what personal data is published to which Web 2.0 applications. In this way, the personal data space has prime responsibility for the management of data while the Web 2.0 applications are responsible for how that data is shared.

A user's data will be heterogeneous and may be created and managed by a set of different desktop applications. We propose an integrated database architecture that allows this data to be viewed and managed through a single interface.

User Interface. We strive for a web-based, pluggable interface architecture of the personal data space component similar to that offered by many Web 2.0 applications. Sites such as Facebook provide an integrated portal-like solution to the management of all sorts of data through a very simple, intuitive style of interface. The standard user interface provides a core set of applications to manage basic information such as contacts, messages, photo albums etc. However, the interface is extensible and customisable in the sense that it is very simple for users to install other applications of interest and even to write their own applications. This plug-and-play style typical of many Web 2.0 applications makes it easy for users to customise their user interface in terms of the types of information stored and published, their own visibility, the level of information sharing and also the layout. A user might replace the standard picture management application with an application that better matches their requirements. We believe that taking this plug-and-play user interface paradigm which has proven to be successful in many Web 2.0 platforms and applying these concepts to personal data management in particular, but also to Web 2.0 data management in general, offers new possibilities and a great flexibility for users.

Database Interface and Components. The pluggable interface architecture is reflected on the data management level by so-called database components. A database component manages data of a specific domain and can be reused and extended by other components and applications. So a user might have a person component where contact information is managed as well as a photo component where photos are stored. The photo application however can make use of both the contacts and the photos components and offer the functionality of tagging people on photos.

Application developers are free to create new components that suit their application model. Note that no predefined data model is imposed, since we believe that users and application developers should be free to design their own models to suit the desired application domain. However, the component architecture supports the combination and extension of existing components when building new applications, which has several advantages. Data common to several applications can be reused and schema evolution is supported in that, for every application, a developer is free to create new components, combine existing components and add additional schema information. Additionally, data of different types, either residing in the same or in different components, can easily be associated. We use an object-oriented database management systems and application developers can implement their applications using a well-defined database API that comprises all functionality common to such systems.

4.2 Web 2.0 Data Sources

The term Web 2.0 data source stands for all kinds of online platforms and applications that manage user data such as Facebook, Flickr etc. and that offer an API to access and synchronise data. In our architecture, these data sources are represented by so-called synchronisation end-points. A synchronisation end-point is the conceptual representation of the Web data source that defines its characteristics. This includes the information about the Web data source's data model as well as the connection configuration that is used to access the data source. Note that, for every Web data source, a synchronisation end-point needs to be implemented, which uses the Web data source's API to access and synchronise data. In order to synchronise user data with a specific Web data source, a user needs to configure a profile which contains all the information needed to connect to and access the data of the Web data source such as the login information.

4.3 Publishing and Synchronisation

Data stored and managed in the personal data space can be published to and synchronised with one or more Web data sources. A user can easily configure to which Web data sources they would like to publish their data. The configuration process includes three steps.

First, a user needs to define the data mapping from the local domain model of the data to be published to the data model of the Web 2.0 data source. Note that data mapping is supported on the attribute level, which allows for a very fine-grained control. A user might decide to synchronise local contacts information with contacts information on Xing and Facebook. For each of these Web 2.0 data sources, a data mapping has to be defined. While Xing is more suitable for managing professional contacts data, Facebook tends to be used for more personal information. A user can, for example, specify that the attribute `work_place` from the local `contact` type should be mapped to the attribute `current_employer` from the Xing `contact` type, while the local attribute `birthday` should be mapped to the Facebook attribute `birthdate` from the Facebook `friends` type.

Second, the synchronisation mode and frequency have to be defined. A user can decide whether data should be synchronised unidirectionally or bidirectionally. While unidirectional synchronisation consists of only publishing content to a Web 2.0 data source, bidirectional synchronisation allows changes made to data through Web 2.0 applications to be synchronised with the personal data space.

Third, in the case of bidirectional synchronisation, there needs to be a mechanism for handling conflicts. If both desktop data and Web 2.0 data are to be synchronised, conflict may arise if both data sources have changed. A user can either decide on automatic conflict handling, defining either the local or online data source as the master copy, or, decide to resolve conflicts manually when they arise.

The proposed architecture supports transparent handling of data duplicates and the configuration of data publishing and synchronisation strategies while managing all of a user's data locally, with a unified view and full control over the data. Additionally, the component-based database architecture allows for data and schema reusability, where components can be combined and linked together with an additional layer on top of the components to support cross-component associative linking and browsing.

5 Discussion

The architecture presented in the previous section addresses the set of requirements presented in Section 3. First of all, the problem of data fragmentation is addressed by managing all of a user's data locally in a personal data space and providing the user with a unified view over all of their data. Second, the user is free to decide where data should be published, how data should be synchronised, either unidirectionally or bidirectionally, and according to what synchronisation strategy. By supporting the configuration of the publishing and synchronisation process, a user is able to let data duplicates be controlled as well as allowing consistency be handled automatically in a transparent and well-defined way. Furthermore, it is up to the user to define the data mapping between the data model of their personal data space and the data model provided by the Web 2.0 data source. Doing so, a user can specify on a very fine-grained level which data items should be published and therefore which are publicly available and which not.

Note that bidirectional synchronisation opens up new aspects of personal data maintenance similar to the approach taken by the FOAF project [8], where personal profile data is managed and maintained in a highly distributed way. Given that users of online networking platforms keep their profile data up-to-date, bidirectional synchronisation can take advantage of this fact and update all contacts of a user's local address book with the most up-to-date data from the corresponding contacts on the social networking platforms that they use. Thus, using bidirectional synchronisation for social networking data enables a user to let the information about all their contacts to be updated automatically without needing to manage and maintain that information.

The proposed architecture is general enough to deal with any kind of application domain, leaving the users and application developers free to decide what kind of data and application should be developed and used.

While data integration approaches mostly operate "on top" of a set of data sources to be integrated, we believe that a bottom-up approach for data creation and publishing is much more intuitive from a user's perspective. In [13], an approach for social networking data aggregation is presented which aggregates social application data "on top" of social networking applications. While this can be a suitable solution for analysing social networks in general, we believe that it is much more natural for personal data management to provide the users with a central point of data management and with means to configure publishing and synchronisation.

6 Conclusion

We have presented the issues of data management that arise in modern settings where personal data is managed by both desktop and Web 2.0 applications. We propose a separation of concerns between data management and data sharing, with Web 2.0 applications having responsibility for the latter while personal data is managed locally and published to Web 2.0 applications under the control of the user. We have presented a data management architecture based on this approach highlighting ways in which data could be synchronised between local data and Web 2.0 applications. Further, for the management of personal data we propose an approach based on the notion of database components influenced by the plug-and-play approach offered by many Web 2.0 applications. As a work in progress, we are currently building a system based on the architecture presented in this paper. This system will serve as a basis to implement an application to manage personal contact information both locally on the desktop and through Web 2.0 applications.

References

[1] O. Bergman, R. Beyth-Marom, and R. Nachmias. The project fragmentation problem in personal information management. In *CHI '06*, pages 271–274, 2006.

[2] T. Berners-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Harper San Francisco, 1999.

[3] A. Bozzon, S. Comai, P. Fraternali, and G. T. Carughi. Conceptual modeling and code generation for rich internet applications. In *ICWE*, pages 353–360, 2006.

[4] V. Bush. As we may think. *The Atlantic Monthly*, 176(1):101–108, 1945.

[5] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD '05*, pages 85–96, 2005.

[6] X. Dong and A. Y. Halevy. A Platform for Personal Information Management and Integration. In *CIDR*, pages 119–130, 2005.

[7] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff i've seen: a system for personal information retrieval and re-use. In *SIGIR '03*, pages 72–79, 2003.

[8] FOAF. Friend of a Friend Project, http://www.foaf-project.org.

[9] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Rec.*, 34(4):27–33, 2005.

[10] E. Freeman and D. Gelernter. Lifestreams: a storage model for personal data. *SIGMOD Rec.*, 25(1):80–86, 1996.

[11] R. Geambasu, C. Cheung, A. Moshchuk, S. D. Gribble, and H. M. Levy. Organizing and sharing distributed personal web-service data. In *WWW '08*, pages 755–764, 2008.

[12] J. Gemmell, G. Bell, and R. Lueder. MyLifeBits: a personal database for everything. *Commun. ACM*, 49(1):88–95, 2006.

[13] I. Guy, M. Jacovi, E. Shahar, N. Meshulam, V. Soroka, and S. Farrell. Harvesting with SONAR: the value of aggregating social network information. In *CHI '08*, pages 1017–1026, 2008.

[14] A. N. Joinson. Looking at, looking up or keeping up with people?: motives and use of facebook. In *CHI '08*, pages 1027–1036, 2008.

[15] D. R. Karger, K. Bakshi, D. Huynh, D. Quan, and V. Sinha. Haystack: A General-Purpose Information Management Tool for End Users Based on Semistructured Data. In *CIDR*, pages 13–26, 2005.

[16] D. R. Karger and W. Jones. Data unification in personal information management. *Commun. ACM*, 49(1):77–82, 2006.

[17] Laszlo Systems. Openlaszlo, an open architecture framework for advanced ajax applications. Technical report, Laszlo Systems (Technology White Paper), 2006.

[18] T. O'Reilly. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. 2005.

[19] M. A. V. Salles, J.-P. Dittrich, S. K. Karakashian, O. R. Girard, and L. Blunschi. iTrails: pay-as-you-go information integration in dataspaces. In *VLDB '07*, pages 663–674, 2007.

[20] K. A. Shoens, A. Luniewski, P. M. Schwarz, J. W. Stamos, and I. Joachim Thomas. The rufus system: Information organization for semi-structured data. In *VLDB '93*, pages 97–107, 1993.

[21] M. L. Trigueros, J. C. Preciado, and F. Snchez-Figueroa. A method for model based design of rich internet application interactive user interfaces. In *ICWE*, volume 4607, pages 226–241, 2007.

[22] J. L. Weaver. *JavaFX Script: Dynamic Java Scripting for Rich Internet/Client-Side Applications*. APress, US, 2007.

[23] C. Wenz. *Essential Silverlight*. O'Reilly Media Inc., 2007.