Semantic-Aware Mashups for Personal Resources in SemanticLIFE and SocialLIFE

Sao-Khue Vo, Amin Anjomshoaa, A. Min Tjoa

Semantic-aware Mashups for Personal Resources in SemanticLIFE and SocialLIFE

Sao-Khue Vo, Amin Anjomshoaa and A Min Tjoa

Institute of Software Technology and Interactive Systems, Vienna University of Technology, Vienna, Austria

{saokhue,anjomshoaa,amin}@ifs.tuwien.ac.at

Abstract. SemanticLIFE is a Semantic Desktop system, which deals with the personal lifetime data. However, SemanticLIFE is limited to local storage, which is an isolated data repository. In the recent years, people have the tendency to share their resources, which are not only stored locally on their personal computers, but also hosted on social networking sites (SNSs). We propose and use the term 'SocialLIFE' to denote one's lifetime information in SNSs, in which personal resources are his/her activities, interests, and related connections. In this paper, we also propose a mashup language and a semantic-based mashup framework. The final goal of this research is to provide a semantic-based way for bridging the gap between SemanticLIFE and SocialLIFE in order to integrate and reuse existing personal resources of existing applications such as information resources of Semantic Desktops and SNSs. The proposed mashup system also aims to supports non-expert users to create data mashups based on semantic-aware mashup dataflow.

Keywords: Semantic Desktop, Semantic Web, Social Networks, Linked Data, Linked Data Services, Semantic Mashup.

1 Background and Motivation

Business objectives are accomplished successfully when human resource management systems are developed and implemented according to organizational goals, particularly if personal information management (PIM) is adopted to utilize all employee information productively [1]. Some PIM systems use Semantic Desktop approach, which create a semantic layer for integrating applications and personal life items, as the means to support users in information management.

Today, an increasing number of organizations/enterprises are taking advantage of mashups, which support users in fast integration of heterogeneous data from multiple sources [2][3]. Furthermore, they also benefit from the collaborative principles of Web 2.0 technologies by using social networking sites (SNSs) to build their social

activities/relations and support their knowledge management. As a result, the amount of heterogeneous data assets, which are distributed among personal and business domains, is increasing. In addition, those PIM systems are limited to isolated data repositories, and do not fulfill most of the requirements for a holistic collaborative environment at the organizational level. Although enterprise knowledge management systems facilitate reusing and accessing of knowledge resources, these systems cannot work effectively unless knowledge workers contribute their knowledge resources and assets to organizations/enterprises.

In order to support users in exploiting the potential of sharing data on the web, the data should be managed in a machine-processable way by applying Semantic Web technologies. During designing and discussing issues around the Semantic Web, Tim Berners-Lee came up with the new term "Linked Data" that describes a method for publishing and interlinking structured data so that it can become more useful for people or machines in exploring the Web of Data [4]. In the meantime, we observe a number of contributions in the Enterprise 2.0 domain [5] that are aiming to apply Web 2.0 and SNSs principles to create an effective and collaborative community, and to explore the role of social computing in achieving the performance goals of enterprises. The potential of Enterprise 2.0 cannot be fully realized without an active support and an increased involvement of human resources [6]. Using SNSs in an enterprise environment, employees can share their data (e.g. skills, interests, or activities, etc.) with their groups or colleagues. From the enterprise perspective, employees can collect business information from customers and partners through SNSs by exploring the relationship of business establishments, professionals, or individuals.

From the above remarks, we can derive the need for a flexible and semantic–aware approach to bridge the gap between internal and external data sources, and especially bringing the relevant personal resources into enterprises. The ultimate goal of this paper is also to find the solution for semantic-aware mashups of personal resources from Semantic Desktops and SNSs following the trends of Enterprise 2.0.

2 Personal Resources in SemanticLIFE and SocialLIFE

SemanticLIFE is a prototype of a PIM system that has been developed in Vienna University of Technology to store, organize and manage various personal life items [7]. The architecture of SemanticLIFE framework is depicted in Fig. 1. It provides a repository of lifetime personal data from varied resources (e.g. email messages, web browser history, images, contacts, phone calls, life events, and other resources). However, the major limitation of Semantic Desktops approaches is that it is restricted to a personal computers and its precious semantic information is not yet effectively used in business processes and tasks that people deal with in their workplace and daily life [8].

With the advent of Web 2.0, many organizations/enterprises have started using the Web 2.0 techniques by applying SNSs in order to increase the effectiveness of their business by means of a tighter collaboration. People have the tendency to share their knowledge or resources, which are not only stored locally on their personal computer

or isolated data repositories, but also hosted on SNSs on the web (e.g. Google documents, LinkedIn profiles, Flickr images, Twitter tweets, etc.).

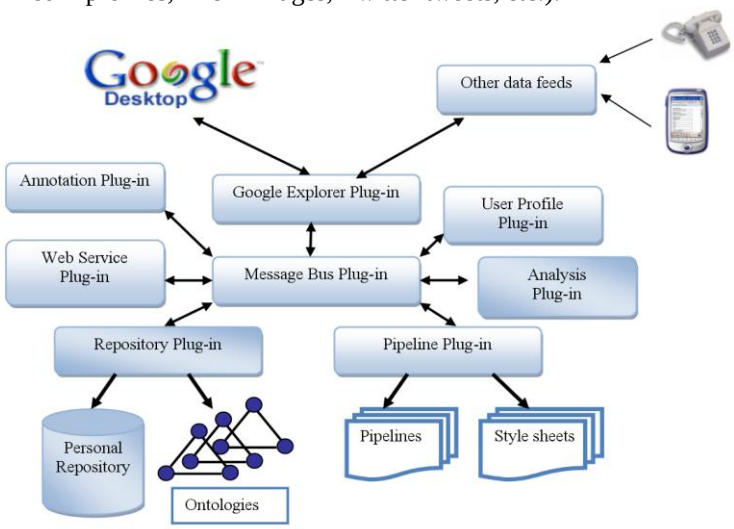

Fig. 1. SemanticLIFE framework [6]

In our research, we propose and use the term 'SocialLIFE' to denote one's lifetime information in SNSs, in which personal resources are user activities (messages, comments, twits, etc.), user interests (books, movies, etc.), and other related connections (friends, colleagues, etc.). To be more specific, one's SocialLIFE consists of interconnections of people relations such as friendship or business/professional relationships on Facebook or LinkedIn; their interests such as video, image or music on YouTube, Flickr, MySpace and so on.

With the aim of interlinking data on the Semantic Web to support people or machines in exploring the Web of Data, Linked Data delivers an important mechanism for information management and information integration [2]. Linked Data extends the standard Web technologies such as HTTP, URIs, and RDF to share information in a machine-readable way. Besides, the term "Linked Open Data" (LOD), refers to a community project that is aiming to encourage people or organizations to follow the Linked Data principle and publish their raw data under open licenses [2]. The Semantic Web community has gained momentum with the widespread publishing of LOD and the big number of promising datasets with a high potential for further exploration. In order to bridge the gap between Semantic Desktops and Linked Data, several research projects are conducted [9], [10]. These researches have the common goals to combine resources from Linked Data and Semantic Desktops by using semantic metadata as a common denominator, and to enrich the data within the personal information as well as the enterprise spaces.

To benefit from the LOD cloud, it is crucial to put personal resources into a context that facilitates the interlinking of resources and enables powerful personal services. These personal resources are not just textual content but also multimedia content such

as music, videos, pictures, etc. There are a number of well-known vocabularies such as FOAF, SIOC, DBPedia, etc., which empower data sharing and linking personal resources with LOD cloud. Furthermore, for the multimedia content, there are some application-specific exporters that allow users to export their data. For instance, Flickr's data can be exported via tools such as Flickr2RDF [11] and FlickrWrappr [12] that parse the user profile and image metadata in order to generate the RDF serialization which reuses the FOAF vocabulary. With a high potential for further exploration, SNSs have served as convincing and useful platforms for linking and reusing heterogeneous data of a user's SocialLIFE for performing further operations or aggregations.

As a result, the existing ontology models and exporters can be reused to extend the SemanticLIFE ontology and support information integration with one's SocialLIFE. Figure 2, depicts the linking of personal resources in SemanticLIFE and SocialLIFE with the LOD cloud.

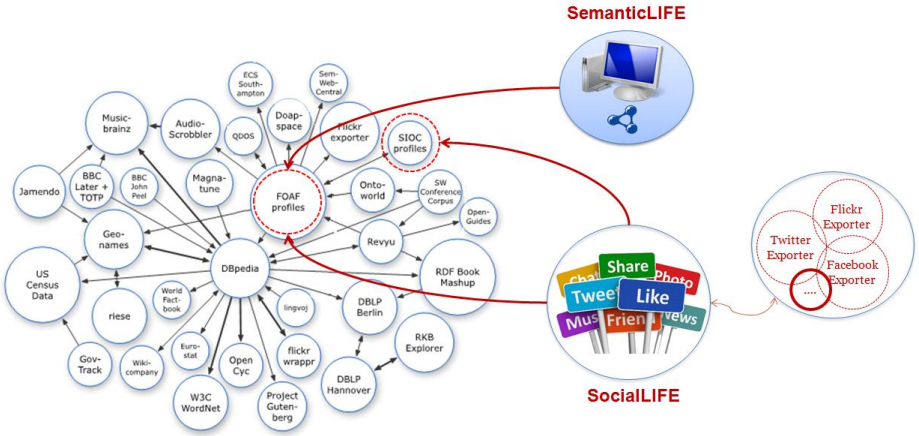

Fig. 2. Linking SemanticLIFE and SocialLIFE with LOD Cloud

To illustrate the potential of this approach, a motivating use case for personal financial data (such as bank statements) will be presented in Section 4.1. A number of financial institutions and banks allow their customers to access their financial statements via APIs or download their statements in standard formats such as Open Financial Exchange format. The Open Financial Exchange (OFX)[1] is a unified specification to exchange electronic financial information between financial institutions and consumers via the Internet. In order to turn personal financial data into mashable artifacts, personal financial data can be converted to RDF/N3 and linked with personal profile.

All gathered information in Semantic Desktops and SNSs platforms have to be parsed into semantic personal services or mashable semantic-based resources that provide a semantic way to express and exchange information from heterogeneous resources. In this context, there are a number of research efforts that try to combine the services and Web of Data. Linked Data Services (LIDS) [13] is one of these ap-

[1] http://www.ofx.net/

proaches that by definition provides HTTP URIs for entities. Dereferencing the LIDS' URIs returns an RDF description of the service input entity, its relation to the service output and the output data itself. This facilitates the transition from data silos to the Web of Data and enables the automatic integration of data resources.

The LIDS is applied as a suitable solution in our approach, in which we use the vocabulary of LIDS to define and describe the relevant services. The advantage of the LIDS is its capability to build semantic models of Web APIs semi-automatically. This includes the lowering and lifting processes that is converting RDF to the required input data of services and constructing semantic data out of a non-semantic API or service response, respectively.

To illustrate this by an example, consider a sample service for searching personal events in SocialLIFE, which has the URL *http://localhost/sociallife/thing*. This service is also used in the latter mashup use cases, which will be presented in Section 0. In the lowering side of this service, the variables are added to the URL of this service as a query string, in which the variables of the corresponding SPARQL query *"SELECT ?thing WHERE {?thing foaf:name ?q}"* are bound to the variables of the service URL. For instance in case of querying the *conference* events, the SPARQL query will be rewritten as *"?events foaf:name ?conference"*. In the lifting side, this service takes the name and the query variables to form the service URL *"http://localhost/sociallife?thing=events&q=conference"*. If the client accepts RDF format, the service returns the results as RDF data. This example is illustrated more details in Figure 3.

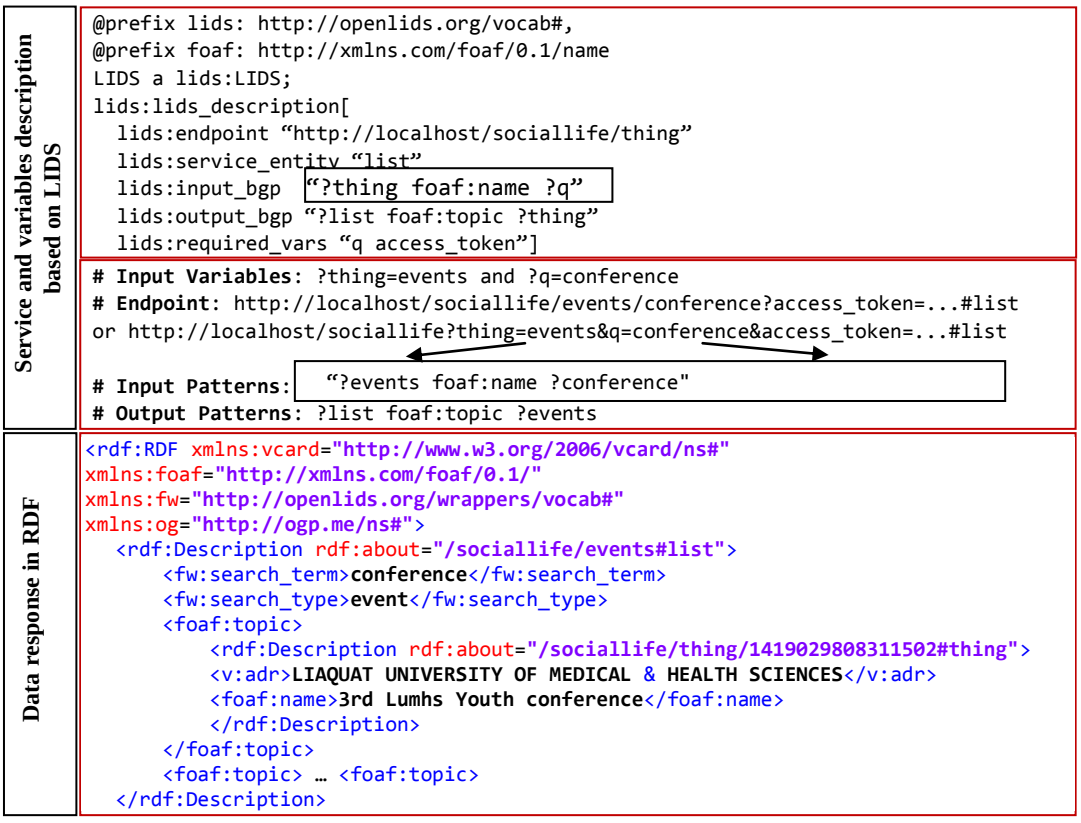

Fig. 3. An example of semantic-enabled personal services based on LIDS

3 Semantic-based Mashup

Mashup is a web application that uses content from different data sources to generate a new service presented in a single graphical interface[2]. Based on the semantic approach, semantic mashup are mashups whose combined services and APIs are supported or annotated by a semantic layer for supporting (semi)-automatic service composition in mashup creation [14]. In this section, a semantic-based mashup architecture is proposed, which aims to: (i) facilitate the integration of heterogeneous personal resources from SemanticLIFE and SocialLIFE within organizations/enterprises; and (ii) improve the mashup usability by supporting the end users to choose the appropriate input data via mapping domain ontologies. This semantic-based mashup framework consists of the following four layers (as illustrated in Fig. 4):

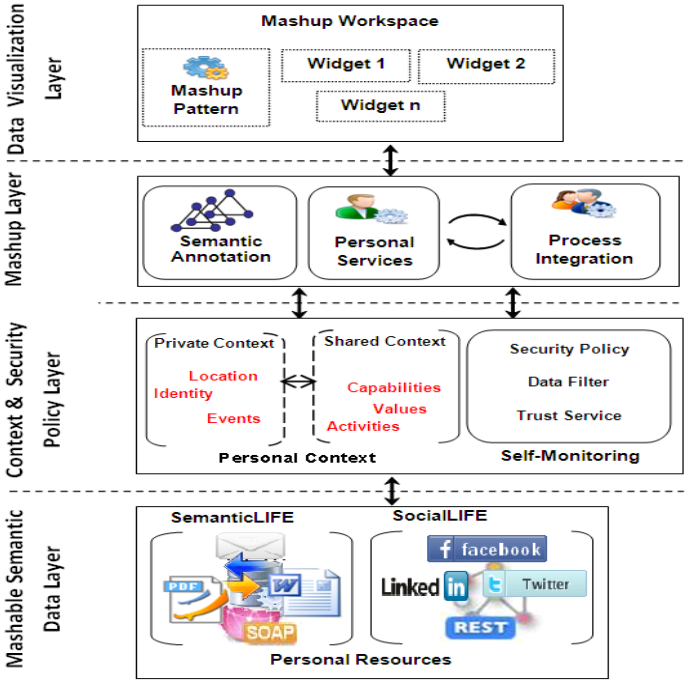

Fig. 4. Semantic-based mashup architecture for SemanticLIFE and SocialLIFE

- *Mashable semantic data layer*: focuses on the retrieval of personal resources from SemanticLIFE and SocialLIFE, and their converting them to RDF based on the relevant domain ontologies.
- *Context & Security Policy layer*: This layer is in charge of the efficient implementation of information security and privacy policies. More importantly, this layer includes context ontologies for describing personal services and the enforcement of

[2] http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)

privacy policies for self-monitoring of shared information in social networks [15]. Further details about this approach are provided in our previous article [16].

- *Mashup layer*: In this layer, the mashable semantic data of lower layers will be integrated and used in various use cases of data integration. To create mashup, the services of SemanticLIFE and SocialLIFE are referred in the mashup language that will be introduced in Section 3.1.
- *Data visualization layer*: This layer enhances the original contents by adding graphical representations like maps or images (e.g., Google Map, Flickr, etc.). This layer constitutes the main workspace that allows end-users to interact with the mashup platform.

In our framework, the following three groups of services are supported:

- *SPARQL-based services:* These services query the semantic data via dedicated SPARQL endpoints such as DBpedia[3], Events[4], etc.
- *Third party services:* These services represent the third party APIs that do not expose RDF data. Examples of such APIs are Flickr, Google Map, etc.
- *Personal services:* These services query the personal resources via custom personal services or data repository of Semantic Desktops.

Our semantic-aware mashup platform is equipped with a resource mashup language, which is used to create widget compositions and dataflows based on the aforementioned services. In the following section, more details about this mashup language is provided.

3.1 Personal Resources Mashup Language

The proposed Personal Resources Mashup Language (PRML) aims to:

- Create a simple mashup language, which uses the domain ontologies and helps developers to create semantic-aware data widgets.
- Formulate the composite solutions based on connectable widgets in order to address the user requirements.

The PRML has four main components, namely mashup, environment, widget, and parameter as described in Fig. 5.

The root element of PRML schema is the *mashup* element and contains an *environment* sub-element, which can be SemanticLIFE, SocialLIFE or any other predefined context.

The third level of PRML contains the required widgets with the following primary attributes:

- *context* and *type*: are the name and type of the required context respectively.
- *source*: indicates the service source of widgets (SPARQL endpoint, third party service, or personal service).
- *role*: indicates the role assigned to a specific user or a group of users who can use the widget.

[3] http://dbpedia.org/snorql/

[4] http://eventmedia.eurecom.fr/sparql/

- *mapping*: supports mapping to ontology resources or properties, for example, *mapping="rdf:type :Place"* is mapping the input/output with the Place concept of the target ontology.
- *parameters*: indicates the additional parameters of the target widget (input and output parameters).

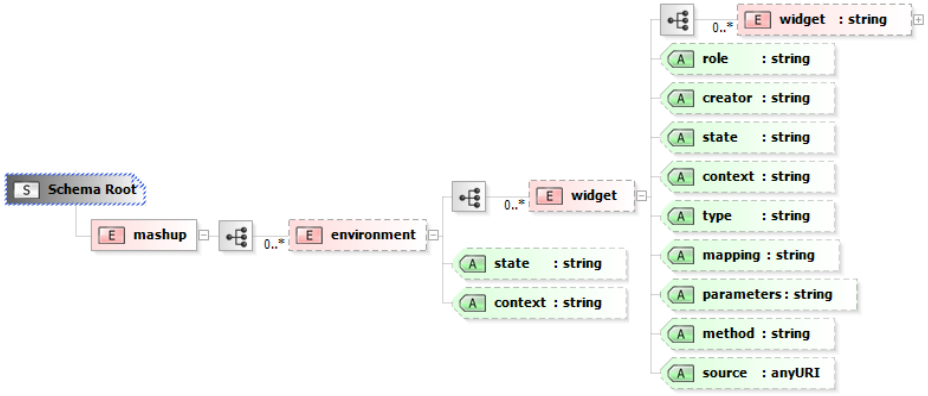

Fig. 5. The schema of personal resources mashup language (PRML)

The fourth level of PRML are *parameters* which contain one or more *input* and *output* elements with the following attributes:

- *type*: is the type of input/output parameters (String, Number, Array, Boolean, Date).
- *mapping*: supports mapping to the ontology resources or properties.
- *acceptedData/dataFormat*: is the type of data format that is accepted by the input ports. The data format can be JSON, XML, SPARQL-results or object.
- *operator*: identifies an operator to construct constraints (for instance: > , < , regular expression, etc.)
- *requireSource*: indicates the input of the target widget if required.

3.2 Semantic Mashup Formalization

In this section, the formal definition of mashup concepts and mashup rules based the proposed mashup language will be defined.

Definition 1. Widget

A widget is a tuple of <I, R, P> where I = $\{i_1, i_2, \ldots, i_p\}$ is a set of Input ports, R is the result set and P is the process (or web service) which is run internally in the widget to consume the inputs I and produce the result R. Each one of these elements has a set of metadata that describes the functions and properties of that element. The input ports and output results will have the previously defined attributes of *input* and *output* parameter of PRML (e.g. *type, mapping, acceptedData/dataFormat, etc.).*

The process may include further metadata such as additional parameters for the widget (e.g. id, name, width, height…), process description, and type of process (e.g.,

SPARQL, Javascript based, Web Service, etc.). In order to use widgets, they should be instantiated and called by the mashup framework.

Definition 2. Mashup

A mashup M is a set of 4-tuple, $M = \{<w_i, c_{ij}, w_j, I_{wj}> \mid \forall i,j=0\ldots n, i \neq j\}$ where

- $w_i, w_j \in W$ are instances of widget (W is the set of available widgets)
- c_{ij} is the connector between two widgets that denotes the dataflow between output port of widget w_i and the I_{wj} input port of widget w_j.

In order to help users to design mashups in a semantic way, some rules for the mashup process are needed that are defined as follows.
Let $D = \{d_1, d_2, ..., d_q\}$ be a set of data formats, and $O = \{o_1, o_2, ..., o_r\}$ be a set of ontology types.

Rule 1: Feasible Connection

A feasible connection between two widgets w_i and w_j is a connection c_{ij} where the data format (d_i) or mapping ontology type (o_i) of the output port on widget w_i side is compatible with d_j or o_j on the input port at widget w_j side.

Rule 2: Avoiding loops

If the Mashup contains a 4-tuple $<w_i, c_{ii+1}, w_{i+1}, I_{wi+1}>$ then there should be no path $<w_{i+1}, c_{i+1\ i+2}, w_{i+2}, I_{wi+2}> \ldots.<w_{i+n}, c_{i+n\ i+n+1}, w_{i+n+1}, I_{wi+n+1}>$ where $w_{i+n+1} = w_i$.

3.3 Widget-based Query generation

To query remote RDF resources, the corresponding query is constructed based on widget parameters as follows:

Suppose $I = \{i_1, i_2, ..., i_n\}$ is a set of input parameters of a given widget and R is the resulting output of this widget. Each input parameter i_j (j=0…n) has also its optional properties (i.e., name, mapping, type, value and operator).

A query Q can be defined as: Q = <S, W, F> where:
- S = <SELECT, R, I>: SELECT statement with relevant parameters
- W = <WHERE, R, I>: WHERE clause with relevant parameters
- F = <FILTER, I>: FILTER constrains with relevant required parameters

Query Q is generated in three steps as described below (depicted in Fig. 6):
- Step 1: The parameters' names are enumerated for creating the SELECT part. The SPARQL variables of SELECT statement are formed as "*SELECT ?R* $?i_{1[name]} \ldots ?i_{n[name]}$*"* by using output R and relevant parameters' names.
- Step 2: The output R and the parameter names and mappings are used to form the triple patterns of WHERE clause as follows: "*WHERE {?R* $?i_{1[mapping]}\ ?i_{1[name]}$ *. … . ?R* $?i_{n[mapping]}\ ?i_{n[name]}$ *}*".
- Step 3: The names, values, and operator of parameters are used to add filter constraints. Depending on relevant parameter's operator, the syntax of the FILTER will be formed differently. For example, if the operator of i_j is *'regex'*, the string

matching syntax must be "*FILTER (regex(?$i_{j[name]}$, ?$i_{j[value]}$))*"; otherwise, the syntax must "*FILTER (?$i_{j[name]}$?$i_{j[operator]}$?$i_{j[value]}$)*".

Defining Widget Parameters in PRML

```
<parameter>
  <input name='i1[name]' mapping='i1[mapping]' type='i1[type]'
  value='i1[value]' operator='i1[operator]'/>
  <input name='i2[name]' mapping='i2[mapping]' type='i2[type]'
  value='i2[value]' operator='i2[operator]'/> … … …
  <input name='in[name]' mapping='in[mapping]' type='in[type]'
  value='in[value]' operator='in[operator]'/>
  <output name='R'/>
</parameter>
```

Parsing parameters into SPARQL

```
# With ij∈P, j=0…n
SELECT ?R ?i1[name] ?i2[name]… ?in[name]
WHERE {?R ?i1[mapping] ?i1[name] .
       ?R ?i2[mapping] ?i2[name] . …
       ?R ?in[mapping] ?in[name] .
<!-- if ii[operator]= 'regex'-->
FILTER (regex(?ij[name], ?ij[value]))
<!-- else -->
FILTER (?ij[name] ?ij[operator] ?ij[value])}
```

Fig. 6. SPARQL query generation based on widget parameters

3.4 Widget UI generation

In the proposed mashup system, UI Widgets are both a graphical user interface and a software component with a specific function. In PRML syntax, the UI Widgets can be described as a user interface with standard web form elements. In this section, a convenient and flexible UI generation mechanism for parsing and rendering the form widgets will be provided. This mechanism and the required steps for the UI Widget rendering process is described in the figure 7:

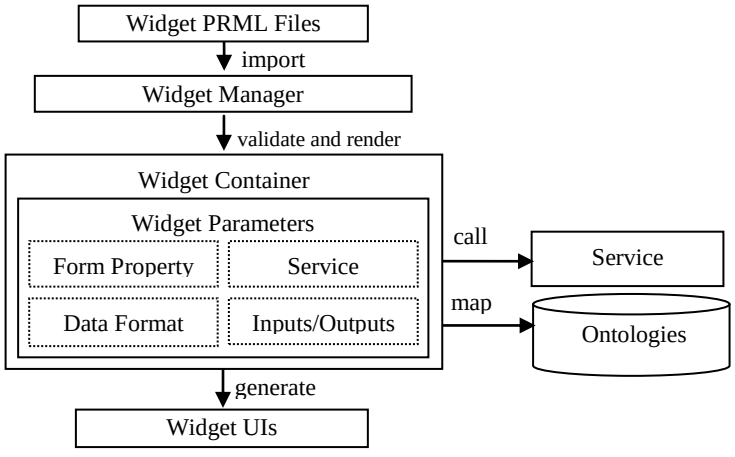

Fig. 7. Widget UI generation mechanism

The following example describes the generation steps of Widget UI with the appropriate ontologies and services from a PRML file (as depicted in Fig. 8):

- The widget's parameters are extracted from the corresponding PRML file: for example, the BankStatements widget defines the following input parameters in the PRML description file: *account, memo, amount, start_date, and end_date*; and output parameter *bank_statements.*
- Widget description files are then imported into the widget manager component.
- Widget container validates and processes the PRML files. In this step, widget parameters are rendered as form elements that are also mapped to the appropriate concepts of domain ontologies. As such, each input parameter is represented as a form element (e.g. textbox, combo box or date field). For example, in case of BankStatements widget, the account input parameter will be represented as a textbox and is mapped to the ontology concept *ofx:ACCTID* and the input parameter *start_date* is rendered as a date field which is mapped to the ontology concept *ofx:DTPOSTED*, and so on.
- The description of UI Widget will be also used to generate the required SPARQL query with relevant variables as described in section 3.3.

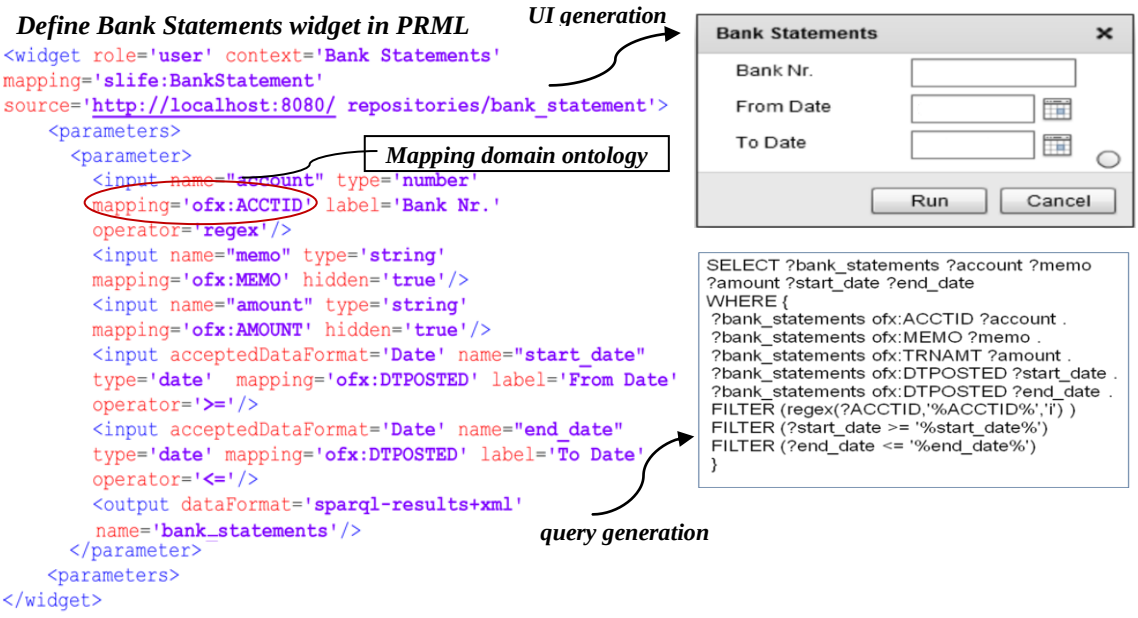

Fig. 8. An example of Widget UI and widget-based query generation

3.5 Mashup Algorithm

Suppose that a set of widgets $\{w_i \ldots w_j\}$ are used in a specific mashup composition M. If w_j is requested to execute, then it is necessary to check whether wj requires output from another widget w_i or not. If w_j has no required input, then it can be executed independently. Otherwise, the requied widget that provides the required input needs to

be executed first. For this reason, a mashup algorithm for possible loop detection is required.

The algorithm for mashup execution is based on the acyclic directed graphs, in which each widget of mashup is considered as a graph vertex, and each connection between two widgets is considered as a graph edge. For iterating through all widgets of a mashup (nodes of the graph), the Depth first-search (DFS) algorithm is used to visit the widgets. In case of mashup, the output value of any selected widget will be the input value of the connected widgets via the relevant port. The mashup M and the visited widget will be handled by a recursive function. Each visited widget will be asked to execute its internal process and return the value via its output port. The returned value will be then considered as the input value for the connected widgets. Instead of checking all edges at once in DFS algorithm, the edges are checked in two phases by differentiating discovery edges and back edges via input ports or output port respectively. Discovery edges are those edges that connect a vertex to another descendant node, and back edges are those edges that connect a vertex to another ancestor node. The mashup execution algorithm is described in the following recursive pseudo code:

```
Algorithm processWidget(M,widget)

Input: mashup M, an instance widget ∈ M
1:  if widget.inputPorts > 0 then
2:         for each inputPort ∈ widget
               //get connected widget
3:             previousWidget =  inputPort.connectedWidget
4:             if previousWidget is executed then
                   //get value of connected widget
5:                 widget.inputPort.value = previousWidget.outputPort.value
6:             else
7:                 //recursive-process connected widget
8:                 widget.inputPort.isVisited = true
9:                 processWidget(M,previousWidget)
10:        end for
11: end if
12: widget.isExecuted = true;
    //executeProcess function: execute process inside widget (SPARQL, service, etc)
13: widgetResult = executeProcess()
14: for each outputPort ∈ widget
15:       nextWidget = outputPort.connectedWidget
16:       if nextWidget.inputPort is visited then
17:          //get value of current widget
18:          nextWidget.inputPort.value = widgetResult
19:       else
              //recursive-process connected widget
20:           nextWidget.inputPort.isVisited = true
21:           processWidget(M,nextWidget)
22:       end if
23: end for
24: return widgetResult
```

Algorithm 1: Mashup algorithm

4 Use Cases

As a proof of concept of the proposed approach, some mashup use cases have been defined to prove the usefulness of our mashup platform and combine the personal resources of SemanticLIFE and SocialLIFE.

4.1 Use Case: Personal Finance Mashup

The first use case is dedicated to the Tim Berners-Lee's vision about combining different data resources [17]. He has formulated this vision as follows: "*The Semantic Web is a web of data. There is lots of data we all use every day, and it is not part of the web. I can see my bank statements on the web, and my photographs, and I can see my appointments in a calendar. But can I see my photos in a calendar to see what I was doing when I took them? Can I see bank statement lines in a calendar? Why not? Because we do not have a web of data. Because data is controlled by applications, and each application keeps it to itself*". Besides, you might have many bank accounts and want to keep a record of different bank transactions from those accounts in a single view (i.e., a calendar view). For realizing this use case, we have created a specific data convertor that converts the financial data from OFX format into RDF triple and stored in the SemanticLIFE repository. Figure 9, depicts for the personal finance mashup. To illustrate and demonstrate the feasibility of this use case, two basic widgets are used:

- *Bank Statements widget*: which retrieves the bank statements of a given bank account in a specific period of time.
- *Calendar widget*: which shows the details of bank statements in a single and flexible view.

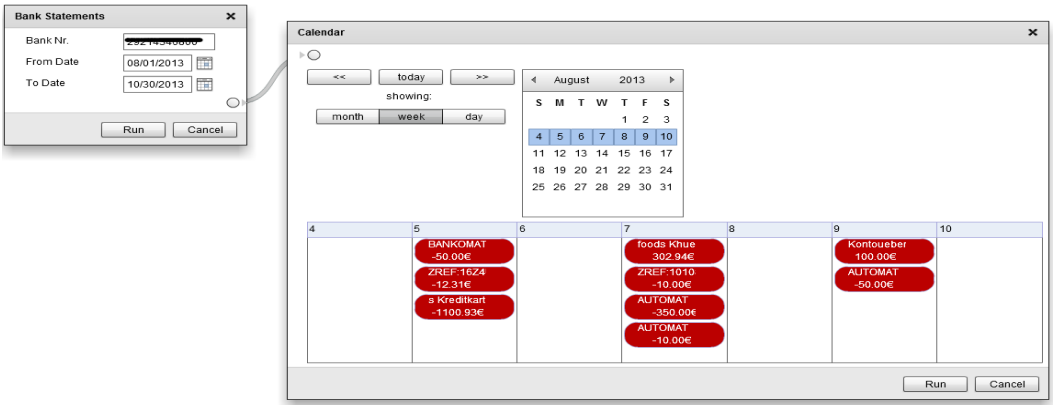

Fig. 9. Personal finance mashup for showing bank statements in calendar view

4.2 Use case: Personalized Mashup

This use case is aiming to combine and enrich some personal data with relevant data from social networks and is formulated as follows: *I want to check the events in my local calendar as well as events from my SNSs in a specific time. For a specific event, show me some famous tourist attractions in the location of that event, including some*

photos (if available), other additional contextual information (weather condition, my friends, etc.).

This use case can be realized via a personalized mashup that uses the following six widgets (as depicted in Fig. 10):

- *Calendar Events widget*: retrieves personal calendar events from SemanticLIFE or SocialLIFE (i.e., Facebook events as mentioned in Section 2). This widget will return the events with relevant information (e.g. locations or organizer).
- *Tourist attractions widget*: calls the third party service to return information about places based on the given input location. This service may be a query to DBpedia via its dedicated SPARQL Endpoint.
- *Geolocation converter*: is a widget that converts the place address into the geolocation format for viewing in Google map.
- *Google map widget*: shows the obtained places in a Google map.
- *Flickr widget*: is a third party service of Flickr to search photos that match some given criteria. In this use case, the matching condition will be the name of the place.
- *Weather widget*: shows the weather forecast of the given place.

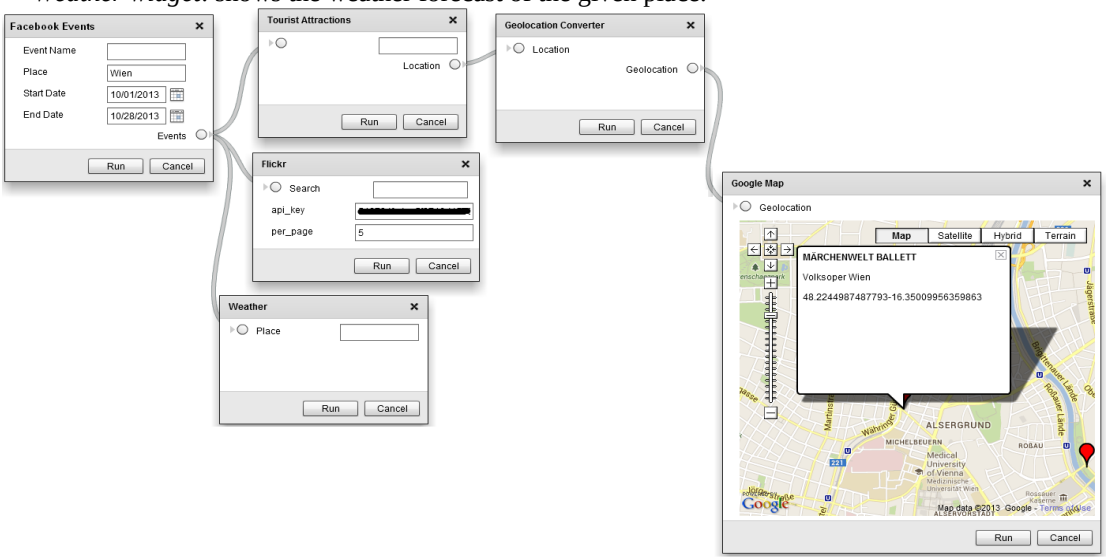

Fig. 10. Design personalized mashup.

5 Related Work

There are a number of research works that deal with part of our proposed approach: (i) expanding the scope of Semantic Desktops with social community aspects, and (ii) building semantic mashup applications to support end-users integrate and reuse their heterogeneous personal life items. Nepomuk [18] is mainly enabling a more meaningful searching on the desktops and bridging the gap between Semantic Desktops and Linked Data, but does not support end-users in data mashups. DERI Pipes [19] have mainly focused on the data aggregation that aims at (semi-) automating mashup development, but end-users are still required to master a basic level of programming knowledge, such as specifying parameters for operators and widgets. Besides, some

mashup languages (e.g. Yahoo Query Language[5], Enterprise Mashup Markup Language[6]) are proposed for the description of programming logic and presentation of data. However, these languages are mostly not easy to understand for end-users without prior knowledge of those languages or extensive programming skills. In addition, those languages does not support ontology or customized ontology mapping yet and support only the construction and formulation of service calls and their execution.

In our semantic-based mashup framework, end-users without programming skills can easily find the right and feasible widget connections on the fly. Fig. 11 depicts the semantic-aware dataflow between widgets and depicts how the mashup platform supports the end-users to find the appropriate input/output connections.

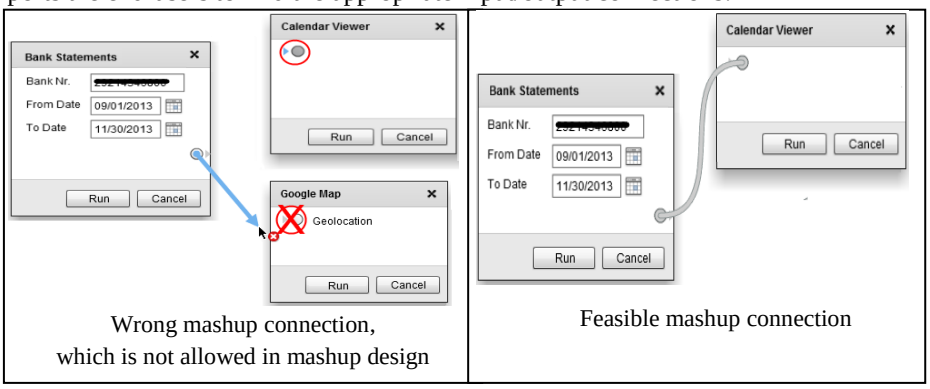

Fig. 11. Semantic-aware dataflow

6 Conclusion and Future Work

In this paper, we have developed a mashup framework where users and mashup developers can collaborate to design personalized mashups on demand. We have also proposed a mashup language that supports mapping ontologies in designing semantic-aware mashup dataflows. The overall objective of this approach is aiming at overcoming the steadily increasing problem of information overload of Semantic Desktops and SNSs (i.e. SemanticLIFE and SocialLIFE in particular). To achieve this goal, the proposed solution expands the scope of our SemanticLIFE in particular and the Semantic Desktops systems in general into the Web of Data.

As future work, the development of the proposed system will be continued to further improve the mashup language and the mashup framework. In this context, we will extend our widget collection to include more third party services of well-known SNSs in order to enrich both personal resources and the mashup repository. Furthermore, the development towards the automatic widget composition will be investigated. Another interesting issue is the mashup sharing which should be supported in order to enable end-users in building data mashups collaboratively in organizations/enterprises. In addition, further experiments for performance and usability evaluation should be conducted.

[5] http://developer.yahoo.com/yql/
[6] http://en.wikipedia.org/wiki/EMML

References

[1] L. Baird and I. Meshoulam, “Managing Two Fits of Strategic Human Resource Management,” *Academy of Management Review*, vol. 13, 1988.

[2] Tim Berners-Lee, “Linked Data,” 2006. [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html. [Accessed: 22-Sep-2012].

[3] A. Anjomshoaa, A M. Tjoa, and A. Hubmer. "Combining and integrating advanced IT-concepts with semantic web technology mashups architecture case study." In Intelligent Information and Database Systems, pp. 13-22. Springer Berlin Heidelberg, 2010.

[4] G. Bader, Wu He, Amin Anjomshoaa, and A. Min Tjoa. "Proposing a context-aware enterprise mashup readiness assessment framework." Information Technology and Management 13, no. 4 (2012): 377-387.

[5] D. Hinchcliffe, “14 Reasons Why Enterprise 2.0 Projects Fail,” 2009. [Online]. Available: http://blogs.zdnet.com/Hinchcliffe/?p=718. [Accessed: 25-Sep-2012].

[6] E. Metter, T. Perrin, V. Gyster, and R. Lamson, “Enterprise 2.0 and HR - Realizing the Potential,” *IHRIM J.*, vol. XII, no. 5, 2008.

[7] M. Ahmed, H. H. Hoang, M. S. Karim, S. Khusro, M. Lanzenberger, K. Latif, E. Michlmayr, K. Mustofa, H. T. Nguyen, A. Rauber, A. Schatten, T. M. Nguyen, and A. M. Tjoa, “‘SemanticLIFE’ - A Framework for Managing Information of A Human Lifetime.,” in *iiWAS*, 2004, vol. 183, no. October 2004.

[8] A. Anjomshoaa, “Integration of Personal Services into Global Business,” Vienna University of Technology, Vienna, Austria, 2009.

[9] L. Drăgan, R. Delbru, T. Groza, S. Handschuh, and S. Decker, “Linking Semantic Desktop Data to the Web of Data,” – *ISWC 2011 SE - 3*, vol. 7032, pp. 33–48, 2011.

[10] T. Groza, L. Drăgan, S. Handschuh, and S. Decker, “Bridging the Gap between Linked Data and the Semantic Desktop,” - *ISWC 2009 SE - 52*, vol. 5823, pp. 827–842, 2009.

[11] Kanzaki, “Flickr photo info to RDF image description,” 2007. [Online]. Available: http://www.kanzaki.com/works/2005/imgdsc/flickr2rdf. [Accessed: 25-Sep-2012].

[12] C. B. C. Bizer, “Flickr wrappr,” 2009. [Online]. Available: http://wifo5-03.informatik.uni-mannheim.de/flickrwrappr/. [Accessed: 11-May-2012].

[13] G. Bader, A. Anjomshoaa, and A M. Tjoa. "Privacy aspects of mashup architecture." In Social Computing (SocialCom), 2010, pp. 1141-1146. IEEE, 2010.

[14] S. Speiser and A. Harth, “Integrating Linked Data and Services with Linked Data Services,” in *The Semantic Web: Research and Applications SE - 12*, vol. 6643, Springer Berlin Heidelberg, 2011, pp. 170–184.

[15] A. Malki and S. M. Benslimane, “Building Semantic Mashup,” in *Proceedings ICWIT 2012*, 2012, pp. 40–49.

[16] A. Anjomshoaa, K. Vo-Sao, A. Tahamtan, A. M. Tjoa, and E. Weippl, “Self-monitoring in social networks,” *International Journal of Intelligent Information and Database Systems*, vol. 6, no. 4. p. 363, 2012.

[17] W3C, “W3C Semantic Web Activity,” 2001. [Online]. Available: http://www.w3.org/2001/sw/. [Accessed: 15-Aug-2013].

[18] T. Groza, S. Handschuh, K. Moeller, G. A. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjónsdottir, “The NEPOMUK Project - On the way to the Social Semantic Desktop,” in *Proceedings of I-MEDIA-2007 and I-SEMANTICS-07*, 2007, pp. 201–210.

[19] D. Le Phuoc, A. Polleres, G. Tummarello, and C. Morbidoni, “DERI Pipes: visual tool for wiring Web data sources,” 2009.