Task Memories & Task Forums: A Foundation for Sharing Service-based Personal Processes

Rosanna Bova[1,2], Hye-Young Paik[3], Boualem Benatallah[3], Liangzhao Zeng[4], Salima Benbernou[1]

[1] LIRIS, University of Lyon 1, France
[2] LIESP, University of Lyon 1, France,
[3] CSE, University of New South Wales, Australia
[4] IBM T. J. Watson Research Center Yorktown Heights, NY 10598

Abstract. The growing number of online accessible services call for effective techniques to support users in discovering, selecting, and aggregating services. We present WS-Advisor, a framework for enabling users to capture and share task memories. A task memory represents knowledge (e.g., context and user rating) about services selection history for a given task. WS-Advisor provides a declarative language that allows users to share task definitions and task memories with other users and communities. The service selection component of this framework enables a user agent to improve its service selection recommendations by leveraging task memories of other user agents with which the user share tasks in addition to the local task memories.

1 Introduction

The recent advances in ICT comprising Web services, pervasive computing, and Web 2.0 promise to enable interactions and efficiencies that have never been experienced before. Users will have ubiquitous access to a network of services along with computing resources, data sources, and user friendly tools [1]. The concerted advances in services oriented computing [2] and pervasive systems [1] provide the foundations for a holistic paradigm in which users, services, and resources can establish on-demand interactions, possibly in real-time, to realize useful and context-aware experiences.

Service oriented architecture and Web services propose abstractions, frameworks, and standards to facilitate integrated access to heterogeneous applications and resources. There has been major progress in terms of services description, interaction protocols, services discovery and composition [2]. More specifically, services composition languages and frameworks foster agile integration by simplifying integration at the communication, data or business logic layers. Furthermore, by leveraging efforts in semantic web services [3], service composition frameworks made a forward step in enabling automated support for service description matching [3].

Although existing composition techniques have produced promising results that are certainly useful, they are primarily targeted to professional programmers (e.g., business process developers). Composition languages are still procedural in nature and composition logic is still hard to specify [4]. Specifying even simple personal processes is still time consuming. Most of the time, if the needed integrated service is not available, users need to access various individual services to become self-supported. For example, a driver might need to use several services including location, travel route computation, traffic information, and road conditions services to get timely information regarding a trip in progress [1].

The emerging wave of innovations in Web 2.0 promotes a new paradigm in which both providers and end users (including non-expert users) can easily and freely share information and services over the Web [5]. For instance, users publish and share information (e.g., URLs, photos, opinions) via personal blogs, social networks and Web communities. Portals such as YouTube, Flickr and Delicious flourished over the years as user-centric information sharing systems. These systems offer more simple and ad-hoc information sharing techniques (e.g., free tagging and flexible organization of content) [6]. We argue that applying such easy-and-free style of information sharing to service oriented paradigm would offer tremendous opportunities for automating information management, in particular for the area of managing data from a variety of end-user processes (or referred to as *personal processes* [16]). Such processes are apparent in office, travel, media, or e-government. For example, travellers will script personal travel arrangement tasks and office workers will script purchasing processes, etc.

In this direction, we propose the WS-Advisor framework. WS-Advisor provides a declarative language and an agent-based architecture for sharing task definitions and services selection recommendations among users.

In our previous work [7, 8], we proposed *task memories*. In a nutshell, a task memory represents the knowledge about services that have been selected in past executions of the task and contexts in which these services have been selected (i.e, the information about the contexts in which certain combination of services were considered most appropriate by users). Task memories are used during service selection to recommend most relevant candidate services. By applying continuous feedback on the on-going usage of services, the system is able to maintain and evolve the task memories, resulting in more fine-tuned service selection.

In this paper, we focus on task definitions and task memories sharing in the WS-Advisor framework. More specifically, we make the following contributions:

- To simplify task information sharing, WS-Advisor provides a metadata model and a declarative (SQL-like) language. Together, they provide an effective management platform for a task repository. The language is used for specifying *task sharing policies.* which includes what a user is willing to share about a task repository and with whom.
- We propose the concept of *task forums* to maintain task repositories (e.g., task definitions and task memories) that may be of interests to several individuals or communities. Essentially a task forum provides a means to collect and share domain specific task definitions and task memories among users. This allows users to reuse and customize shared definitions instead of de-

veloping definitions of tasks from scratch. In addition a task forum uses publish/subscribe interaction model to support service selection recommendations for the "masses".

We briefly overview the basic concepts in Section 2. Section 3 introduces the meta-data model and task sharing policies. Section 4 presents task forums. Section 5 describes the implementation architecture and finally discussion and concluding remarks are presented in Section 6.

2 Preliminaries

In this section, we summarize some background concepts used in the remainder of this paper, namely *task, task memories* and *context summary queries*, to keep the paper self-contained. Details about these concepts are presented in [7, 8].

Task Definition. A task in WS-Advisor represents a set of coordinated *activities* that realize *recurrent needs*. For example, a frequent business travel task may include activities such as hotel booking, car rental, flight reservation, meeting scheduling and attendee's notification. We assume a service ontology in which service categories, service attributes and operations for each service category are defined. A task is described in terms of services ontologies and is defined using UML state charts. A state can be basic or composite. Each basic state is labeled with an activity that refers to either an operation of a service. A task definition can be translated to executable processes such BPEL. Also, the administrator associates each task with its relevant contexts (e.g., for a travel booking task, the user's timezone, local currency, smoking preferences, type of Web browser, may be relevant). Therefore, when a user chooses a task, any relevant contexts can be determined. This enables WS-Advisor to consider context information during service selection and such context forms the basis for making recommendations.

Context Summary Queries. A context summary query (CSQ) represents a context to be considered by the service selection process. It is specified using a conjunctive query of atomic comparisons involving context attributes, service attributes, service operation inputs/outputs, and constants. How the context summary queries are generated and associated with a task is explained in [7]. For example, for a flight booking task, relevant context attributes may be `origin`, `destination`, `price`, `seat class`, `currency`, etc. and a CSQ for a task could be expressed as a set of pairs `(c,v)` where `c` is an context attribute and `v` is the value (e.g., {(origin, "Sydney"),(destination,"Paris"),(seat class, "Economy")}).

Task Memory. A *task memory* is a data structure which captures the information about the contexts and combinations of services that have been successfully used in the past to execute a given task. The service selection process considers task memories so that the selection is not only based on the description or content of services but also on how likely they will be relevant in a given context. The design of task memories is independent of the different service selection strategies as presented in [8]. A task memory is associated with each task and is gradually built overtime using the context and service selection history.

3 Metadata Model for Sharing Tasks and Task Memories

We first present a set of metadata abstractions designed for sharing tasks, task memories and task categories, as well as a simple declarative SQL-like language for manipulating them. Most attributes of metadata are self-explanatory. Hence, we will mainly elaborate the ones that need clarifications.

3.1 Representing Task Categories, Tasks and Task Memories

Users: `Users` represent a group of users with whom a single user maintains some relationships. We assume that a user will add/delete his/her contacts from `Users` as appropriate [1].

Users.

user_id	**user_relationship**	**user_type**
Luc	friend	individual
Jazz_music	common_interest	group

user _relationship specifies the nature of relationships. Possible values are `business`, `family`, `friend`, `common_interest`. user_type specifies the type of user_id which can be either `individual` or `group` (i.e, a community of users with special interests).

Tasks: As metadata for managing task definitions, we define `Tasks` with the attributes shown below. task_annotation_tags represents a collection of keywords that characterize a task. The attribute task_query_schema is a collection of attributes that could be used to query a task. For example, attributes origin, destination, travel_start, travel_end could be the ones for the Sydney trip plan task.

Tasks.

task_name	**task_annotation_tags**	**task_query_schema**
Sydney_Trip	sydneytrip, australia, 2007trip, holiday	origin,destination, travel_start, travel_end
Visa_Trip	international, 2006trip, business	origin, destination, period, VISA_number

Task Memories: We represent task memories with the attributes: tm_name, context_summary and recommendations.

Task Memory.

tm_name	**context summary**	**recommendations**
biz_tripAug06	{(origin,“Lyon”), (destination,“London”), (price, <500)}	{[(AustrianAirline, Ibis), 0.7, Luc]; [(Qantas, Hilton),0.5, Moby]}
hol_tripApr07	{(origin,“Lyon”), (destination, “Sydney”), (price, >1000 && <2000)}	{[(VirginAirline, Stamford), 0.8, Luc]; [(Qantas, Ibis),0.4, Luc]}

[1] This is very much like the way Internet users are finding their “friends” in social networks.

We have explained context summary earlier. The attribute recommendations represents the most preferred combinations of services to execute a given task. In fact, for each context summary query, the task memory maintains the $K(K >= 0)$ most preferred services to execute for the given activity. Each service is associated with a positive weight value, called `Global Affinity` (`GA`), exceeding a predefined threshold [5]. The global affinity of a service measures the relevance of the service in performing the activity in the given context in the task. More precisely, this value represents a weighted average of the values that measures the level of satisfaction of users, about the service, with respect to all the possible services that have been selected in that context[6].

We represent recommendations as a set of triple (sc, score, p), where sc is itself a combination of web services, score represents GA for the combination and p denotes the provenance of the recommendation which references the user or group `Users`. The issue of trusting the provenance of a recommendation and whether a user is allowed to share outsourced recommendations is interesting by itself and is outside the scope of this paper.

Task Categories: For intuitive manipulation, browsing, and querying of tasks we provide the notion of task category (similar to folders and files abstractions desktop user interfaces). A task category is defined as a view (in the database terminology) over tasks and other task categories.

Task_Category.

category name	**category_tags**	**sub_categories**	**tasks**
Travel	Q2, Q3, trip, tourism, travelogue, lodging	International_Travel	Visa_Trip, Sydney_Trip

```
Q2: SELECT  task_annotation_tags FROM Task
WHERE task_name = 'Sydney_Trip'
Q3: SELECT  task_annotation_tags FROM Task
WHERE task_name = 'Visa_Trip'
```

The attribute category_tags represents a collection of keywords that characterize a category (e.g. for the category *Travel*, the keywords can be, trip, lodging etc). The attribute sub_categories represents a collection of categories that are linked to a category via the specialization relationship (e.g., for the category *Travel*, one sub_category can be International_Travel). Each tuple of this relation is a container of tasks (and categories) of a specific domain. Some attributes of this table may be explicitly provided by the user of specified using relational views (e.g., `Q2` and `Q3`).

Task Repositories: With the metadata described aboved, WS-Advisor can provide what we refer to as *Task Repository*, that is, a repository of tasks, task memories and task categories. Users can interact with the task repository in a number of ways. This includes:

[5] This could be a parameter set by a system administrator.

[6] How to compute GA is presented in [9].

- *Task Repository Browsing*: The task query language of WS-Advisor allows a user to browse a task repository. Users can navigate through the task categories hierarchy, select, a display information about specific categories and tasks.
- *Task Repository Querying*: The task query language of WS-Advisor supports both SQL-like querying over the schema of the task repository (i.e, Tasks, Task_Categories, Task_Memories, and Users relations) and keywords based querying. It also supports task signature queries. Task signature queries are useful to find tasks that can accept given input or output parameters (i.e, queries over task schemas).
- *Task Repository Sharing*: The task definition language of WS-Advisor supports the definition of views for sharing information about tasks (e.g, task categories, task definitions, and task memories) of a given task repository with users in social networks. It also supports the definition views for outsourcing information from other task repositories which are accessible through user social networks. We will illustrate the main features of this language through examples in the next Section.

3.2 Representing Sharing Policies

Besides the metadata representation needed for the sharing, there is a need for a simple and effective language for specifying the “sharing policies” for a task repository. We propose the following set of metadata for the purpose: Shared_Categories, Shared_Tasks, and Shared_Recommendations. One thing to note is that the values of the attributes in the metadata are both conventional data types and “query-types”. That is, the values of the attributes may be queries which are to be evaluated[7].

Shared Categories: This includes the attributes C_To_Share and C_With. A tuple (cq,uq) of Shared_Categories is created by two queries:

- cq is a query over Task_Category and selects a collection of categories to be shared
- uq is a query over Users and selects a set of users who will have access to the categories selected by cq.

Shared Tasks: It is defined to define tasks to be shared. It has attributes T_To_Share and T_With. A tuple (tq,uq) of Shared_Tasks is created by tq which is a query over Tasks and selects a collection of tasks which are to be shared. Same as Shared_Categories, uq represents a query over Users.

In fact, Shared_Tasks enables the user to refine the sharing policies in the sense that s/he can identify a subset ot tasks within a category for sharing instead of all tasks in a shared category.

[7] This is similar to the work proposed in [10].

Shared Recommendations: To share task memories, we use the attributes R_To_Share and R_With. A tuple (rq,uq) in Shared_Recommendations is created by rq is a query over Task_Memories and selects task memories to be shared and uq is a query over Users.

3.3 Importing and Mapping Views

Similar to browsing task repositories, a user can browse and search the views created by other users. Once a user finds a view that is useful to fulfilling her task, she can do one of the followings:

Remote View Importation: View importation provides a means for users to copy task categories and task definitions from the shared views of other users. Definitions can be imported and stored in Imported_Categories and Imported_Tasks tables. The schema of the Imported_Categories (respectively, Imported_Tasks) includes the following attributes: I_Categories (respectively, I_Tasks) and I_User. The attribute I_Categories (I_Tasks) represents a collection of categories (respectively, tasks) that are imported from thats repository of the user identified by the attribute I_User. Defining new tasks could be demanding to end users. This allows users to share, not only recommendations, but also definitions of tasks. More, importantly the importation provides a practical means for easy definitions of tasks.

View Mapping and Query Forwarding: In addition to directly importing a shared view, users can also create a mapping between a local repository and a remote repository. Via the mapping, a query (i.e., query over task categories, tasks or task memories) can be forwarded from one to another. In order to forward queries, there is a need to align the terminologies used in these repositories so that queries expressed over one repository could be translated to queries expressed over the other repository. For such mapping, we presented a peer-to-peer schema mapping approach in [11]. The mapping can be complete (that is, all attributes in shared categories, tasks and task memories views are mapped) or partial (that is, only some of the attributes in the views are mapped. For the ones without mapping, synonyms are to translate the terms). The complete description of the mapping and query forwarding process is outside the scope of this paper.

4 Task Forums

In WS-Advisor, a user forms "links" with other users based on the sharing policies they create (i.e., sharing tasks and task memories). These links can be considered as *views* over remote task repositories and can used to forward queries for the purpose of outsourcing service selection recommendations (i.e., obtaining recommendations from remote task repositories).

Based on the foundation of sharing tasks, in this section, we put forward a concept of *Task Forums* to promote what we call "mass sharing". Creating or importing views provide a mechanism for sharing tasks among a relatively small number of individuals (e.g., friends in your Skype contact list). Task forums aim to take the paradigm to a larger scale.

To explain the idea behind the task forums, we would like to draw an analogy from the Internet user groups (e.g., Google groups). In user groups, users share the same domain of special interests. They come to the group with different levels of skills and expertise in the domain. Inside an active group, we would see a novice user posting questions like "How do I do X", "Where can I find X" or "What is X", etc., and experts providing answers or appropriate recommendations. There would be some feedback mechanism to keep track of the quality of the answers or recommendations. Therefore, overtime, the wealth of high quality knowledge is accumulated and shared by the users in the group.

Task forums operate on the similar idea, but they provide a unique and innovative concept in that their focus is on facilitation of sharing recurrent personal processes (i.e., tasks) and recommendations for services that realise such processes. We envisage that a community of individuals or interest groups will form a task forum, which is specialised in a particular domain and various tasks within it (e.g., task forum of travel plans, task forum of small businesses or task forum of financial plans). Each task forum has a set of peers who are task forums themselves, forming a network of task forums.

In each task forum, there are task definitions, task memories and a set of metadata for storing necessary mapping data for querying peer task forums. Expert users can provide various task definitions and even bind them to a specific execution language such as BPEL. For example, a task forum for small businesses may have task definitions such as *issuing a business registration number*, *filing an insurance claim*, *search for a tax agent near your area*, etc. A novice user can easily import such task definitions and execute them in her own task execution engine. The users can discover and import tasks from other forums and provide mappings so that queries can be forwarded.

Although users can browse and search task definitions and task memories for recommendations, there is a need for effectively managing the communication of interests (similar to posting a question and providing an answer). Inside a task forum, users can use a query publication/subscription mechanism to manage the communication.

Query Publication and Subscription: The concept of shared views allows peers (individuals or communities) to publish information that they are willing to share and with whom. This mechanism allows for importation and query forwarding as discussed earlier. In addition, we use the concept of *query subscription* to allow a peer to receive relevant recommendations from other peers in a proactive manner. This is similar to the concept of continuous queries in publish/subscribe systems [12]. While, any peer can use the mechanism of query subscriptions, we will focus on how a community exploit this mechanism to provide a kind of a mass sharing of service selection recommendations. In order to facilitate such sharing, WS-Advisor models subscriptions using a relation called Query_Subscriptions. This relation includes the following attributes: S_Query, Publishers, S_Mode. The S_Query attribute represents a query over a task schema or a context summary query as in the Task_Memories relation. The attribute Publishers represents a collection of peers (e.g.,community members) with whom the query is subscribed. The S_Mode attribute represents the subscription mode. Currently, our approach supports two subscription modes: push mode and pull mode. In a push mode, a peer identifies the publishers and explicitly subscribe by specifying the relevant subscription queries. In a push mode, a peer in fact publish a subscription and other members register for it. For instance, a community may use an internal monitoring mechanism to identify relevant query or context summary query and publish them. Members of a community may subscribe to provide recommendations about these queries.

In a nutshell, a pull subscription query has the same meaning as a subscription in traditional publish-subscribe systems. A push subscription query is in fact a publication of subscription query. We assume that peers also forward feedback to each other and this especially important for communities, but this issue is outside the scope of this paper.

5 Implementation Aspects

It should be noted, that a detailed description of the WS-Advisor framework and the supporting platform is outside the scope of this paper. Here, we briefly overview the system architecture and describe components that support the concepts and techniques presented in this paper. We adopt a layered architecture for the design and implementation WS-Advisor system. Figure 1 shows the elements of this architecture. The user layer consists of three components. The task manager allows expert users to create task definitions using a state-chart based modelling notation or directly using BPEL. The implementation of this component relies on the services composition editor of the Self-Serv platform [13]. The view manager allows both expert users and non experts users to share information about their task repositories. It also allows users to reuse both task definitions and task memories from task repositories of other users or task forums. The query manager allows users to browse and query task repositories as well as executing individual tasks. The agent layer consists of three agents, namely; the advisor, builder and social network agents. These agents implements the processes related to providing service selection recommendations, building task memories, and maintaining the relationships that a user may have with other users and task forums. More detailed description of this agents is presented in [8].

The service layer consists a gateway to access underlying meta-data and services from both user and agent layers. It provides a number of infrastructure services that we reuse from our existing work on Web services platforms including service discovery and context management engines [13]. In addition to that, to support the task representation and manipulation facilities presented in this

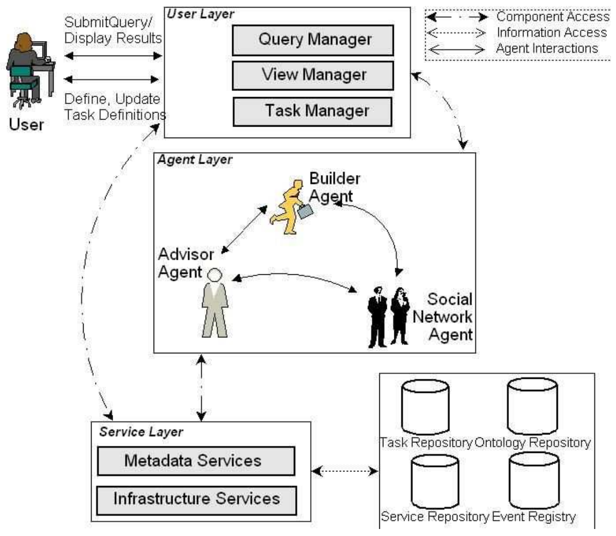

Fig. 1. Implementation Architecture

paper, we propose the use of data services as a foundation to access the information required to manage task repositories. These data services provide operations to query, browse, and update task repositories and event buses. The also provide operations to query and browse service ontologies and service registries. In order to support the publish and subscribe model of task forums, the data services rely on a semantic publish subscribe event bus. The event bus allows social network agents representing individuals or task forums to: (i) publish requests for service selection recommendations, (ii) register with other agents and notify them when relevant service selection recommendations become available, (iii) subscribe with other agents for relevant service recommendation recommendations, (iv) send notifications about relevant service selection recommendation to interested agents. The implementation of the service bus is a work in progress and will rely on the *Semantic Pub/Sub* techniques developed in [12].

6 Discussion and Conclusions

Our work builds upon results in services oriented architectures and semantic Web services to provide a foundation for sharing personal processes. We leverage techniques from the areas of services discovery and composition to cater for the specification and provisioning of user tasks. We leverage results in ontologies and schema mapping in peer-to-peer architectures [11] to support interaction among different task repositories. The above techniques are used in a composition framework called WS-Advisor to allow experienced users to define tasks and build mappings among different and possibly related task repositories. In our previous work [8], we presented an agent-based framework that leverages knowledge about past experiences to improve the effectiveness of Web service selection.

Inspired by advances in Web 2.0 and personal information management and sharing [14, 15, 5], the WS-Advisor framework aims at providing a foundation for easy sharing of users tasks and task memories (i.e, knowledge about past services selection experiences) among individual users and communities. Work in personal data management and sharing [14, 15] focuses on uniform management and sharing of personal data stored in files and databases. As mentioned in the introduction of this paper, systems such such as Flickr and Delicious allow easy and ad-hoc sharing of information such as URLs and photos over the Web [5]. To the best of our knowledge there is no previous work that focuses on sharing personal processes. [6] introduces the concept of service clubs as service-based middleware for accessing e-market places. The Self-Serv framework [13] features the concept of service communities as a mechanism for integrating a large number of possibly dynamic services. Early work on personal processes [16] focuses on providing modelling notations and operators for querying personal processes. This work focuses specifically on catering for the requirements of mobile users when accessing personal processes. Although these efforts produced results that are certainly useful for sharing services and specifying tasks, more advanced techniques that cater for simple and declarative exploration and sharing of task repositories are needed. These techniques are necessary to transition composition systems from the realm a static and elite developer type of business processes to composition systems which are end user-centric.

Our work builds upon these efforts and provides complementary and innovative contributions to facilitate personal processes sharing. We provided a meta-data model and declarative language for sharing task repositories. The meta-data model captures a minimal set of abstractions that are useful for representing tasks, task memories, and user relationships. The proposed language hides the complexity of managing processes by providing an SQL-like language and a number of pre-defined functions for browsing, querying, and sharing task repositories. We proposed the concept of task forums to facilitate the sharing of task definitions and task memories. Task forums act as containers of task definitions and task memories for a specific domain. Non experienced users can outsource task definitions from task forums by using simple importation mechanism (aka file copying). Task forums also provide means for gathering and dissemination of task memories among individual users or communities. We rely on data services to provide uniform access to task repositories. These data services are used to develop applications and interfaces for main functionality of the WS-Advisor (i.e, services selection, personal processes management and sharing). The proposed framework is an important step toward easy and effective sharing of personal processes. Ongoing work includes developing case studies in specific domains such as travel and personal finance to further study the added value of the proposed foundation.

References

1. Coutaz, J, Crowley, J. L., Dobson, S., Garlan, D.: Context is key. CACM 48(3) (2005), pp.49-53
2. Alonso, G., et. al.: Web services: Concepts, Architectures, and Application. Springer (2004)
3. Paolucci, M., Kawamura, T., Payne, T., R., Sycara, K.,P.: Semantic Matching of Web Services Capabilities. ISWC (2002), pp.333-347.
4. Carey, M.J.: Data delivery in a service-oriented world: the BEA aquaLogic data services platform, SIGMOD (2006), pp.695-705
5. Ankolekar, A., Krotzsch, M., Tran, T., Vrandecic, D.: The two cultures: Mashing up Web 2.0 and the Semantic Web. WWW (2007), position paper
6. Tai, S., Desai, N., Mazzoleni, P.: Service communities: applications and middleware. SEM (2006), pp.17-22
7. Bova, R., Paik, H.Y., Hassas, S., Benbernou, S., Benatallah, B.: On Embedding Task Memory in Services Composition Frameworks. ICWE (2007), to appear
8. Bova, R., Paik, H.Y., Hassas, S., Benbernou, S., Benatallah, B.: WS-Advisor: A Task Memory for Service Composition Frameworks. IC3N (2007), to appear
9. Bova, R., Hassas, S., Benbernou, S.: An Immune System-Inspired Approach for Composite Web Service Reuse. Workshop on AI for Service Composition (in conjunction with ECAI 2006)
10. Srivastava, D. and Velegrakis, Y.: Intentional Associations Between Data and Metadata. SIGMOD (2007), to appear
11. Benatallah, B., Hacid, M.S., Paik, H.Y., Rey, C., Toumani, F.: Towards semantic-driven, flexible and scalable framework for peering and querying e-catalog communities. Inf. Syst. 31(4-5) (2006) pp.266-294
12. Zeng, L., Lei, H.:. A Semantic Publish/Subscribe System. In Proc. of the International Conference on E-Commerce Technology For Dynamic E-Business (2004)
13. Sheng, Z., Benatallah, B., Dumas, M., Mak, E: SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment. VLDB (2002), pp.1051-1054
14. Geambasu,R., Balazinska, M., Gribble, S.D., Levy, H.M.: HomeViews: Peer-to-Peer Middleware for Personal Data Sharing Applications. SIGMOD (2007), to appear
15. Dittrich, J.P., Salles, M.A.V.: IMEX: iDM: A Unified and Versatile Data Model for Personal Dataspace Management. VLDB (2006), pp.367-378
16. Hwang, S.Y., Chen, Y.F.: Personal Workflows: Modeling and Management. MDM (2003), pp.141-152