TECHNISCHE UNIVERSITÄT WIEN
VIENNA UNIVERSITY OF TECHNOLOGY

Diese Dissertation haben begutachtet:

…… ……

Dissertation

Integration of Personal Services into Global Business

ausgeführt zum Zwecke der Erlangung des akademischen
Grades eines Doktors der Sozial- und Wirtschaftswissenschaften
unter der Leitung von

O. Univ.-Prof. Dipl.-Ing. Dr. A Min Tjoa
E188
Institute of Software Technology and Interactive Systems
Vienna University of Technology, Austria

eingereicht an der Technischen Universität Wien

Fakultät für Informatik

von

Mag. Dipl.-Ing. Amin Anjomshoaa
0227286

Kammelweg 10/212, A-1210 Wien

Wien, 20. November 2009 _____

Kurzfassung

Das Management und die Organisation einer lebensbegleitenden Speicherung sämtlicher anfallender Informationen stellt eine wissenschaftliche Herausforderung der Informatikforschung seit der bahnbrechenden Publikation von Vannevar Bush im Jahre 1945. Zur Zeit der Veröffentlichung war das vorgestellte Konzept noch hoch spekulativ. Memories for Life bildete eine große wissenschaftliche Herausforderung für die nächsten 50 Jahre nach dieser Veröffentlichung. Die vorgestellte Vision von Vanevar Bush wurde als eine der Grand Challenges der Informatik in den 90er Jahren ausgewählt, die nun als eine multidisziplinäre Herausforderung betrachtet wird, welche Aspekte des Knowledge Management, Information Retrieval, Security/Privacy und der Mensch Machine Interaktion umfasst. Mit dem Aufkommen von Semantic Web wurden neue Möglichkeiten der Organisation von Memories-for-Life-Entitäten und deren Assoziationen durch sogenannte Semantic Desktops eröffnet. In der Literatur existieren jetzt eine Handvoll Semantic Desktop Lösungen und Prototypen. Diese Lösungen beschränken sich allerdings vorrangig auf die Speicherung und Retrieval der Entitäten und der zugehörigen Assoziationen der Semantic Desktops.

Ein nächster wichtiger Schritt zur Erreichung der ambitionierten Ziele einer Memory for Life Grand Challenge ist die Integration einer Semantic Desktop Lösung für betriebliche „Services“ und Soziale Netzwerke. Die immer komplexer werdenden betrieblichen Prozesse machen es notwendig die lebensbegleitenden Informationen der Benutzer in globale betriebliche Prozesse einzubetten. Dieser Aspekt der lebensbegleitenden Berücksichtigung der persönlichen Geschichte der Eigenschaftsausprägungen auf globale und soziale Netzwerkservices wurde bislang kaum untersucht. Um dies zu bewerkstelligen wird eine Serviceebene vorgeschlagen, die eine sichere Verfügbarkeit dieser Schichte sowohl für den Benutzer als auch für globale Prozesse ermöglicht. In dieser Arbeit werden mehrere Herangehensweisen der Informationsintegration vorgeschlagen, wobei Mashup-Architekturen ein besonderes Augenmerk finden. Mashup Architekturen nützen viele der Vorteile von Web 2.0, welches einen Schub der Entwicklung des Internet Computing in der

letzten Dekade bewirkte. Das Web 2.0 bietet neue Möglichkeiten für eine bessere Mensch-Computer-Interaktion mit Hilfe von Anwendungen wie Mashups, welche eine benutzergetriebene Integration von Web-zugänglichen Daten ermöglichen. Mashups sehen den Aufbau effektiver und leichtgewichtiger Informationsverarbeitungslösungen basierend auf den Web Services von Organisationen vor. Solche Web Services können von einfachen Web Services wie RSS und REST-basierten Diensten bis hin zu BPEL-Diensten für komplexere Anwendungsfälle reichen.

Das Ziel dieser Dissertation ist es, die Lücke zwischen persönlichen Informationen und der globalen Prozesse der Geschäftswelt durch die Einführung von semantischen Mashups zu schließen. Dies bewerkstelligt einerseits die sichere und intuitive Erstellung und den Austausch von Informationen und neuen Diensten basierend auf lifetime memories, und fördert andererseits das Internet Computing Paradigma via Geschäftsprozesse, welche die Vorteile von benutzerdefinierten persönlichen Diensten ausnützen.

ABSTRACT

Managing lifetime memories has been a blurry challenge ever since it was originally proposed by Vannevar Bush in his seminal paper (1945). At the time, the concept was extremely speculative and indeed, Memories for Life has remained a challenging problem for the next five decades. Recently this idea was declared one of the grand challenges in computing research where its scope has been expanded and the problem has been outlined as a multidisciplinary problem that should address issues such as knowledge and databases, information retrieval, security/privacy, and human-computer interaction. New possibilities for Memories for Life emerged with the introduction of Semantic Web technologies that make it possible to model a diverse range of memory items and their associations in so-called Semantic Desktops. Today, there are a handful of Semantic Desktops that are to some extent able to organize and manage personal life items. However, their application is limited to the storage and retrieval of various personal items.

The next step towards the goals of the Memories for Life grand challenge is to integrate this organized information in the business and social network services. With the emergence of complex business processes, there is a growing need to automatically map and embed the user's information context into global business services. This aspect of lifetime memories, which may have a great effect on extending the footprint of such memories for interactions with global and social network services, has not yet been explored. In order to realize such use cases, the major obstacle to face, is providing a secure and comprehensible service layer that can be easily used by both end-users and business processes. In this proposal some information integration approaches will be introduced and among them the Mashups architecture will be explored in more details. The Mashup architecture utilizes the advantages of Web 2.0 that has recently hit the mainstream of Internet computing and continues its rapid evolution. Web 2.0 has introduced new possibilities for a better human computer interaction via rich applications such as Mashups that provide a user-driven micro-integration of web-accessible data. Mashup envisions building effective and light-weight information processing solutions based on the exposed

Web Services of organizations. Such Web Services may range from simple Web Services such as RSS and REST based services to complex BPEL services for more serious use cases.

The ultimate goal of this dissertation is to bridge the gap between the personal information world and the global business world by introducing semantically-aware Mashups that on one hand facilitates the secure and intuitive creation and sharing of information and new services based on lifetime personal information, and on the other hand fosters the Internet computing paradigm via business processes that take advantage of user-generated personal services.

ACKNOWLEDGEMENTS

This work is the successive result of some project done in the Institute for Software Technology and Interactive Systems (Vienna University of Technology) under the supervision of Prof. A Min Tjoa. Without his inspiration, advices, guidelines and continuous support this thesis would not have been possible.

I would like also to thank my colleagues at institute, especially the members of SemanticLIFE project and the colleagues in Secure Business Austria that I had the privilege to work with for the fruitful discussions and for creating such a nice work atmosphere.

I am very grateful to Reza Rawassizadeh and Andreas Hubmer for their great work and contribution to the implemented prototypes.

I owe a special thank to my wife Ferial for her patience during past months, and taking care of our little sunshine Kian that allowed me to work long hours in front of computer.

I wish also to thank my mother Zari who bore me, raised me, supported me, taught me, and loved me. Unfortunately my father passed away some months ago and I will not have the chance to share this success with him, but his life's work and lessons will live on in my heart and mind. To him, my Mother, my brother Eiman, and my sister Aida I dedicate this thesis.

Lastly I would like to dedicate this work to all brave Iranians who have sacrificed their lives for my Land and those who are advancing the Green Movement toward a better future for Iran.

Table of Contents

List of Figures

List of Tables

List of Listings

List of Appendixes

Chapter 1

INTRODUCTION AND MOTIVATIONS

The drive to create helper tools is a basic function of human being. We have been doing it since our ancestors made axes during the Stone Age period, and we are still doing it today by inventing modern intelligent devices that help us in different ways.

The idea of "Memories for Life" is the dream of creating an intelligent device that captures and stores the lifetime information of individuals in order to assist human in daily tasks. In 1945, Vannevar Bush has named this intelligent device "Memex" and defined it in his famous article "As we may think" as follows (Bush, 1945):

> *Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and, to coin one at random, "Memex" will do. A Memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.*

The number of atomic facts that the average person knows is astronomical and there are many issues to be addressed before this amount of data can be efficiently used in individuals' daily life and the Memex dreams comes true. The Memex idea has been renewed again in 2004 by the UK Computing Research Committee (UKCRC) who declared the "Memories for Life" as one of the grand challenges in computing research (Fitzgibbon, 2004). The grand challenge has expanded the scope of Memories for Life and outlined it as a multidisciplinary problem that should address issues such as knowledge and databases, information retrieval, security/privacy, and human-computer interaction. New possibilities for Memories for Life emerged with

the introduction of Semantic Web technologies that make it possible to model a diverse range of memory items and their associations in so-called Semantic Desktops.

Today, there are a handful of Semantic Desktops, such as SemanticLIFE (developed at the Vienna University of Technology), Gnowsis (Gnowsis, 2009) and IRIS (IRIS, 2009) that are to some extent able to organize and manage personal life items. The next step towards the goals of the Memories for Life grand challenge is to integrate this organized information with the open world's business and social network services.

It is also important to note that due to recent advances of web and emergence of Web 2.0 and social networks, the concept of “personal life items” is no more limited to the resources that are stored locally on our personal computer or isolated data repositories, but is extended to our social networks and various information that we share on the web such as YouTube videos, Flickr images, weblog entries, Twitter twits, maps, GIS location, etc. Unfortunately, most of the existing Semantic Desktop solutions are now limited to the storage and retrieval of locally stored personal items and the user’s contributed information on social networks is not yet supported.

With the emergence of complex business processes, there is a growing need to automatically map and embed the user's information context into global business processes. This aspect of lifetime memories, which may have a great effect on extending the footprint of such memories for interactions with global and social network services, has not yet been explored. The ultimate goal of this dissertation is to bridge the gap between the personal information world and the global business world by introducing Personal Web Services based on lifetime personal memories and Semantic Web technologies.

1.1 Personal Information Integration

Information Integration is the process of merging information resources that are having different contextual, structural and conceptual representations (Information Integration 2009). It is important to note that information integration process, like many other abstract concepts, has its roots in instinctive human behaviors. Every human is non-consciously engaged in the information processing activities to

acquire, retain, and use the incoming information. After receiving this information, our brain tries to conform them to our pre-existing model of reality. In other words our brain integrates the incoming information into our existing knowledge context and verifies the information consistency. As a result the accuracy of our pre-existing information indicate how close our perceptions, approximate the real world.

From information technology point of view, Information integration is typically done via a mediator schema that maps heterogeneous set of source schemas to each other or to a common target schema. This kind of information integration is a highly moderated process that needs human contribution. In other words both source schema and target schema should be known first and human user will configure and supervise the integration process. The moderated information integration has been implemented successfully in many information exchange scenarios using generic middleware software or dedicated software mediators. Apparently such high-tech solutions cannot be easily adopted and used by non-expert users who need to incorporate the outside-world data in their processes or to feed the external business processes with their personal data.

The personal information integration use cases can be roughly broken into three main categories:

- Outward information integration: In this kind of information integration scenarios, the personal information is utilized to feed external processes that need such information. At the moment, lots of such integrations take place by intensive human intervention. A simple example of these integrations is the online web forms that are filled out by end users who provide their personal data such as name, date of birth, social security number, etc.
- Inward information integration: unlike the outward information integration scenarios that information flow is from user to external processes, the inward information integration deals with consumption of external information resources such as World Wide Web information, to empower internal processes. For instance the information available on web can be used to select the cheapest flight that meets user requirements. Most of such information integration scenarios are currently done via end-user involvement.

- Compound information integration: the information integration scenarios of this kind require both inward and outward information integrations to complete their tasks. An example of complex processes is the trip planning that needs inward information integration to integrate and adopt the available online resources such as flight and hotel information into user's context. This process also needs to use outward information integration to submit the user data to proper services and make the required reservations.

The integration systems are formally defined as a triple <G,S,M> where G is the global (or mediated) schema, S is the heterogeneous set of source schemas, and M is the mapping that maps queries between the source and the global schemas. In other words, the mapping M consists of assertions between queries over G and queries over S. When users pose queries over the data integration system, they pose queries over G and the mapping then asserts connections between the elements in the global schema and the source schemas (Data Integration, 2009). Generally building and applying, integration triples is not feasible without intensive human contribution who uses his/her pre-existing knowledge of the source and target entities to formulate mappings between them. In this context, the Semantic Web technologies might be helpful to facilitate data integration by introducing a computer-readable description of schemas that supports a better computer to computer and computer to human interactions. In other words, the Semantic Web aims to align the content of current web to an explicit specification of conceptualization which is known as ontology (Gruber, 1995). This idea can be also seen as a data integration process that maps the global web schema (outside-world) to user's world and processes. Semantic Web technology has been widely accepted and used to capture and document context information in many domains. It plays a significant role in information sharing scenarios and interoperability across applications and organizations (Anjomshoaa, 2006). This added-value opens the way to integrate huge amount of data and becomes extremely useful when used by many applications that comprehend this information and bring them into play without human interaction.

In Semantic Web based data integration approach, the computer-readable concepts are captured in ontologies that are shared among data integration actors. More importantly ontologies can also afford handling higher-order information by

determining whether source and target entities are semantically map-able to each other. Answering such questions requires the precise definition of metadata that is theoretically feasible by means of appropriate ontologies.

Even though the Semantic Web technologies have flourished consistently in the past few years, it is unlikely to achieve the Semantic Web goals on global Web in near future and the initial expectations such as turning the World Wide Web to a machine-comprehendible medium is far away from current state. Likewise the dream of "turning the World Wide Web to an environment in which information is given well-defined meaning, better enabling computers and people to work in cooperation" (Berners-Lee, 2001), has not yet come true. The best proof of this is a look at the current status of World Wide Web and few websites and services that are Semantically-enabled. Furthermore according to Gartner's report, the Semantic Web which had been among the emerging technologies in the past few years is vanished from the emerging technologies hype cycle in 2009 (see Figure 1.1).

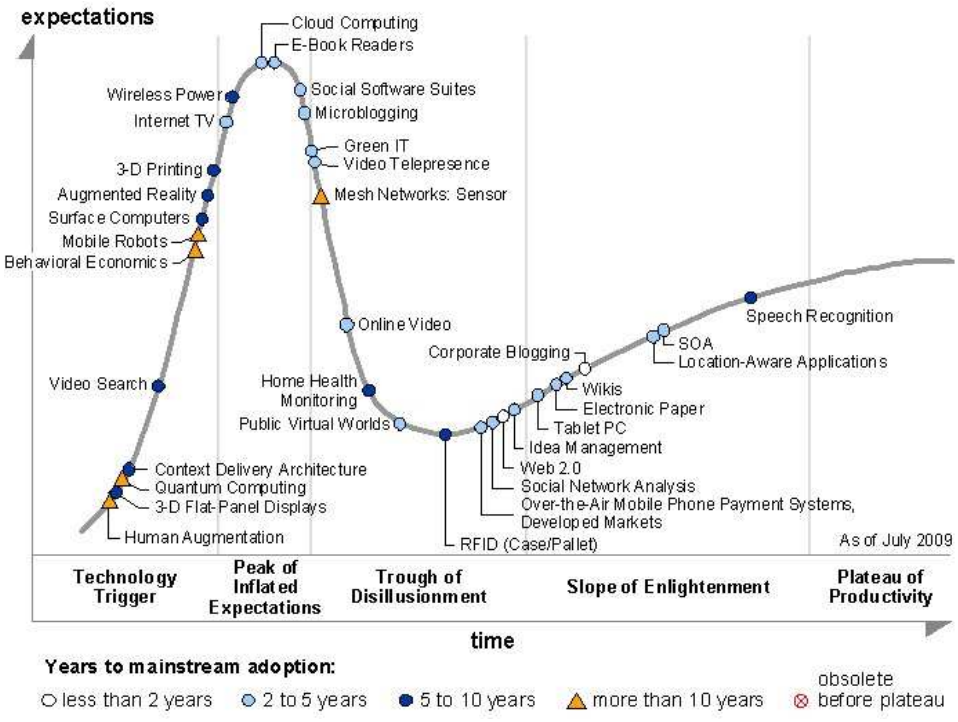

Figure 1.1 : Gartner's emerging technologies hype cycle in 2009

The basic reason for this situation is that so far it is not easy to get people to learn and apply Semantic Web concepts in their Web content and use Semantic Web technologies efficiently in their daily life. The real breakthrough in Semantic Web

implementation happens by emergence of semantic-enabled content authoring and management tools that make the paradigm shift from Traditional Web to Semantic Web feasible.

The proper personal information integration solution should not only be able to share user information with little human intervention, but should also take the security and privacy considerations into account. The need for usable and trusted privacy and security is a critical area in the management of personal information. The Computing Research Association (CRA) Conference on Grand Research Challenges in Information Security and Assurance has identified the ability to "give end-users security controls they can understand and privacy they can control for the dynamic, pervasive computing environments of the future" as a major research challenge (CRA, 2003). This goal demands not only efficient security and privacy policies but also requires improvements to the usability of security aspects. In fact, poor usability can have a negative impact on security, making usability a particularly critical aspect for security and privacy systems.

1.2 Research Questions

There are several specific questions regarding the social, theoretical and technical aspects of realizing personal information integration for closing the gap between user information and open world business processes. Some of the questions that are the main focus of this thesis are:

- How can the semantics of personal information and their associations be modeled accurately for open world interactions?
- How to define personal information sharing services that can interact with global services and share useful information about a specific person in a secure and trustworthy way?
- How can user requirements and preferences be represented and how should they be taken into account in tailoring global services to a particular user?
- How security and privacy policies can be applied to information flow between personal information and global web?

- How information integration solutions can be created and managed intuitively by end users who are not IT experts?

1.3 Thesis Contributions

This thesis is dedicated to different information integration methods that make user interaction with global web easier and more efficient. In this context the Semantic Personal Services are introduced in order to leverage process interoperability, which in turn boosts global business processes. Personal services aim to take over parts of the business processes that are directly related to Semantic Desktop's life items. Furthermore the personal services should also cooperate with global business processes to share the knowledge in a trustworthy and secure way.

More specifically this thesis tackles the information integration problem using different methods such as Semantic Desktop service pipelines, XForms, Web Form Services, and Semantic Mashups. In the rest of this section we will briefly introduce the main contribution of this thesis.

One of the noteworthy contributions of this research work is the introduction of Web Form Services. Nearly all human-computer interactions are happening through Input forms which are responsible for receiving the user input and sending it to appropriate component for further processes. Particularly, a large portion of Internet advances owes the human-computer interaction and data exchange via web forms and we are using them extensively in our daily activities. Current complex Internet applications demand a significant amount of time for development and maintenance of web forms which are solely designed for human users.

To extend the footprint of Web Forms, a novel component has been implemented to translate the Web Forms to a plain Web Service that can be described semantically and integrated in more complex solutions. This Web Services are referred to as Web Form Services in the rest of this paper. Similar to normal Web Services, the Web Form Services present themselves to the end users by means of WSDL conventions.

The Web Form Services accept the web form entries as their input element and the expected response of web form as output. On the backend, Web Form Services are supported by a user imitation service that reads a Web Form configuration and imitates the Web Service functionality.

The implemented Web Form Service prototype shows the feasibility of this approach to turn web sites to web services. The Web Form Server needs just a simple XML configuration which describes the target website and the interactions with corresponding forms. To handle more complex scenarios, the user authentication, caching, and multistep user inputs (wizard-like forms) are supported.

The Semantic Mashup approach which is the main contribution of this thesis will be explored with more details. Mashup Architecture which is one of the outcomes of Web 2.0 paradigm is currently being used for user-centric information processing. At the moment mashups are mainly used for less fundamental tasks such as customized queries and map-based visualizations; however, it has the potential to be used for more fundamental, complex and sophisticated tasks in combination with Semantic Web technologies.

The bidirectional support of Semantic Web and Mashups can provide a solid basis for many interesting applications. The Semantic Web support for Mashups is very credible and has its roots in Semantic Web Services that are aiming to automate service discovery and composition without human intervention. The basic difference between Semantic Web Services and Semantic Mashups approaches is derived from their different target users. The Semantic Web Services are mainly managed and used by IT experts who are aware of underlying data structures and corresponding services, however the Semantic Mashups' target group is ordinary users who need to combine the Mashup Widgets for their specific purposes.

On the other hand, mashups have the potential to facilitate the transition from traditional Web to Semantic Web era and support this paradigm shift with "zero footprints" on the Web pages. In order to distinguish our proposed approach we introduce the concept of Annotation Mashup. Annotation Mashup is an ad-hoc mashup that on one hand connects to the preconfigured information resources and processes their data and on the other hand maps its context data to the relevant domain ontology. In other words, instead of embedding the semantic meaning in the Web content, the semantic is attached to relevant content via mashups in a dynamic and loosely coupled manner. This approach has the following advantages:

- Unlimited number of ontological mappings can be defined for the same content depending on the context of use. As a result two different users can

extract and use the same data but interpret it differently according to their use cases. So for example an extracted price from a Web page can be mapped to income_amount concept for being used in an accounting system and in another use case the same price is mapped to costs_amount concept for a private user.

- Semantic meaning can be added by community and it is not limited to the content owners. As mentioned before the content owners are reluctant to embed semantic meanings in their Web contents. Using Annotation Mashups the semantics can be added to Web pages on the fly with no need to manipulate the original content.
- Semantic Mashups may also support the "open model" communication between organizations. At the moment the conceptualization of business processes and their relevant objects and entities are limited at organizational level. Semantic Web and ontologies are potential candidates to harmonize the inter-organizational data exchange via open models; however, the implementation progress of such systems is time and money extensive. Annotation Mashups as a flexible light-weight component can facilitate the creation of semantic-enabled "open models" that can be shared and understood by business partners via shared ontologies.

Mashups are very helpful in creating fast solutions for data integration, however a major drawback of mashups is the fact that such solution are fragile and not as stable as formal business processes. As the number of serious Enterprise Mashups increases, there is a growing need to make the mashups more stable. To provide a solid basis for such applications, the proposed approach includes a Mashup-BPEL convertor that transforms the user generated mashups to a formal BPEL process. As a result the end user will benefit from simple service composition of Mashups and at the same time the process will be managed and controlled by well established business process engines. Furthermore all complexities of a business process such as defining the partner links, services, etc will be hidden from the end user point of view.

1.4 Thesis Organization

This thesis is comprised of three major parts: Semantic Desktop (chapter two), Information Personalization and Integration approaches (chapters three and four) and Semantic Mashup prototype (Chapter 5). In the first part the Semantic Desktops in general and the specific case of SemanticLIFE will be discussed and its exclusive method of data integration by means of data pipelines will be explained. The second part explores the different approaches that can be used for information integration and their potential applications in personal information integration. The last part presents the Semantic Mashup prototype as a proof of concepts for the proposed approach of this thesis. The chapters are described in more detail below:

- Chapter 2: provides theoretical background and summarizes relevant research work in the area of information integration.
- Chapter 3: provides an introduction of Semantic Desktops. A main part of this chapter is dedicated to SemanticLIFE prototype and its data integration backbone which is called “Service Oriented Pipeline Architecture (SOPA)”.
- Chapter 4: presents two novel approaches for enabling web forms for better integration into business processes namely Semantic XForms and Web Form Services.
- Chapter 5: is dedicated to Mashups and presents Semantic Mashup architecture as a business enabler. This chapter also includes the different use cases that can be addressed using the proposed architecture.
- Chapter 6: summarizes the research work presented in this thesis and presents the main results that have been concluded from the work. Finally the research questions that were listed previously will be revisited to show how the proposed solution will address the challenging issues

Chapter 2

Information Integration

Today, the business informatics has to deals with a highly interdisciplinary network of knowledge resources that are coming from different domains. In most of the cases the analysis of scattered business information and deploying them in different business solution is not conceivable without huge contribution of human users who take over the complex task of information integration. As a matter of fact, a big portion of IT processes deal with the information integration issues which are necessary to shape the information for their specific use cases. In most of the cases such information integration processes are created by IT experts and presented as a service to end users. One of the bottlenecks of these knowledge integration solutions is that the IT solutions typically focus on 20% of user requirements that affect the most users and the long tale of requirement is usually ignored by IT providers (Figure Figure 2.1) (Hoyer, 2008b).

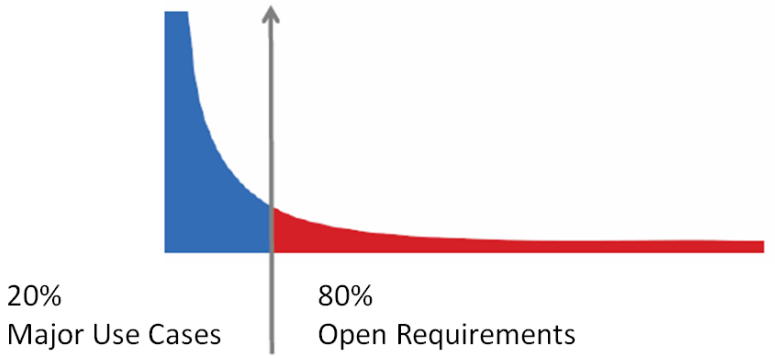

Figure 2.1: Long tale of open requirements

Some field studies has shown that the "Hidden Costs of Information Work" – for covering the long tale of requirements - in an organization with 1,000 employees is

over $10 million annually for reformatting and recreating information (Feldman, 2005).

In this chapter the theoretical background of information integration and the corresponding approaches will be discussed. In particular the information integration between personal information and global internet processes will be explored in more details. It is important to note that the personal information integration scenarios can be easily extended to organizational services for sharing the organizations' business information with trusted partners and other business processes.

2.1 Information Resources on Web

Back in 1967, Doug Engelbart coined the collective IQ of organizations and of society as a grand challenge and the Global Web is an attempt to come a step closer to this vision. It is believed that central principle behind the success of the giants born in the Web 1.0 era who have survived to lead the Web 2.0 era appears to be this, that they have embraced the power of the web to harness collective intelligence (O'Reilly, 2005).

One of the simplest and most common approaches for collective intelligence is the full-text search methods which allow people to search a large data set using some key words. The query results are ranked according to some criteria such as frequency of key words in target resource and/or trustworthiness of the resource publishers. Algorithms for full-text searches are among the most important collective intelligence algorithms and a deciding factor for the success of search engines. For instance the rapid rise of Google from an academic project to the world's most popular search engine was based largely on their specific PageRank algorithm (O'Reilly Media, 2007). Despite all advantages of full-text search engines and their great contribution to information retrieval, in several cases the results are not relevant. For instance if you search any of the famous search engines for terms "Semantic Web" and "Architecture", the top ranked results are mostly about "architecture of Semantic Web technology" which is misleading for a person who was looking for applications of Semantic Web in Architecture, Engineering and Construction (AEC) field. This misunderstanding is because of the different

applications of term architecture in different fields and the fact that computers deal with concepts as string of bytes.

One of the important outcomes of Web 2.0 is a large collection of semi-structured data that are used to empower the collective knowledge systems. The quality of such collective knowledge gets better as more people participate in data creation process and their contributions improve the quality of information. A good example for such collective knowledge system is the Wikipedia project. The database size of Wikipedia is increasing exponentially and many pages are being added each day.

The Web 2.0 approach for creating data has made the created data more structured in comparison to traditional web contents. For instance the taxonomy of weblog entries includes title, description, creation date, author name, tags, and some comments; however this structure is not yet enough for making advanced use of data for more complex use cases that require a richer semantic of web content. To address this problem there are two major research areas namely:

- deep syntactic and semantic analysis of human language
- automatic extraction of semantic relations from the text

The first method which has been a fundamental challenge for many years is still a complex task and many applied computational linguists have now switched to easier challenges such as text classifiers, text-to-speech converters, grammar checking and statistical machine translation. The second group of research activities is focusing on extraction of semantic relation of text and its context. The semantic relations can be events, properties of objects, or geo temporal information that will be used to make a semantic understanding of the texts.

The proposed project is aiming to benefit the advantages of the both text analysis methods and combine the results with the taxonomy of Web 2.0 applications. So on one hand there will be a rather comprehensive understanding of the text and on the other hand taxonomy of different applications of Web 2.0 will help to join and unify this result for the sake of more complex processes such as information integration scenarios.

There are several methods for classification and analysis of Web contents. One of the classification methods, that has been widely used is tagging. In this method users add

an atomic token about what they think about a specific item on the web. For instance a blog entry that explains the recipe of "Wiener schnitzel" might be tagged with cooking, recipe, and Vienna. The tag's quality improves as number of people that are tagging the same item increases. Some Web 2.0 systems use the tag statistics and suggest the top ranked tags to the end users. It is important to note that such recommendation systems have no idea about meaning of tags. The tag count and matching is done by recommendation system, but the tag semantics are in the users' mind. In other words the linguistic meanings of the items are not injected into the machine via tagging. One possible approach for capturing the meaning of entries is mapping the tag tokens to an upper ontology that defines and connects the domain concepts. This can be done either by the text analysis or text matching techniques over the assigned tags.

A more elegant and more precise approach for enriching the Web content is to embed the semantic information in the web content at creation time so that machines can read and interpret the content without the overhead of natural language processing methods. One such solution for making the web contents more intelligent is the W3C's initiative RDFa (RDFa, 2009) which provides a set of HTML attributes to augment visual data with machine-readable hints. It is highly beneficial to express the structure of web data in context; as users often want to transfer structured data from one application to another, sometimes to or from a non-web-based application, the user experience can be enhanced. For example, information about specific rendered data could be presented to the user via right-clicks on an item of interest (RDFa, 2009). The rules for interpreting the RDFa are generic, so that there is no need to define different rules for different formats; this allows authors and publishers of data to define their own formats without having to update software, register formats via a central authority, or worry that two formats may interfere with each other. There are many major use cases where embedding structured data in HTML using RDFa provides significant benefit. For example people's contact information, events and content's license (for example creative commons) can be included in web contents using RDFa syntax and relevant namespaces.

The RDFa is not the only solution for providing more intelligent data on the web. A similar approach for embedding machine-readable data in web content is

microformats (microformats, 2009) which is supposed to be coinciding with the design principles of "reduce, reuse, and recycle". The main difference between these approaches has historical background. The microformats has grown out of the work of blog developer community as an easy and ad-hoc response to common applications, but RDFa, on the other hand, is built with a more systematic vision of the W3 Semantic Web group and its associated thinkers.

2.2 Web Services & SOA

The fast growth of the World Wide Web and the emerging pervasiveness of digital technologies within our information society have significantly revolutionized business transactions, trade and communication between people and organizations. (Medjahed, 2003). Besides the augmentation effect, business-related information is characterised by the fact that it also originates from heterogeneous sources and get more and more complex in structure, semantic and communication standard. Therefore, mastering heterogeneity becomes a more and more challenging issue for research in the area of Business Process Management. This challenge involves all facets of process integration, composition, orchestration, and automation amongst heterogeneous systems. Fortunately, Web Services (Web Services, 2009) which are built on top of the existing Web protocols and open XML standards are considered as a systematic and extensible framework for application-to-application interaction. Web services which are the fundamental block of Service Oriented Architecture (SOA), allow automatic and dynamic interoperability between systems to accomplish business tasks. SOA separates functions into distinct units, or services, which developers make accessible over a network in order to allow users to combine and reuse them in the production of applications (Bell, 2008).

The business processes are then built on top of the existing Web Services and are formulated by the Business Process Execution Language for Web Services (WS-BPEL, 2009) which provides a mean to formally specify business processes and their interactions. By doing so, WS-BPEL extends the Web Services interaction model and enables it to support business transactions and defines an interoperable integration model that should facilitate the expansion of automated process integration in both the intra-corporate and the business-to-business spaces (WS-

BPEL, 2009). In this context the process of assembling the pieces of functionalities into a complex business process is often done with significant human involvement and the main reason for this is the fact that the basic SOA components describe the services at the syntax level and do provide an explicit semantic context for those services. Such semantic context can benefit various service discovery and composition use cases which have been centre of attention in SOA during the recent years.

To address the semantic requirements of SOA, the Semantic Web technologies have come to help by introduction of Semantic Web Services that are aiming to automate service discovery and composition without human intervention. At the moment there is a handful of successful Semantic Web Service frameworks such as WSMF (WSMF, 2002), OWL-S (OWL-S, 2009), WSMO (WSMO, 2009), and METEOR-S (METEOR-S, 2009) that are trying to take the semantic concepts to SOA world.

The service registry, as a standard part of the Service Oriented Architecture (SOA) supplies the business processes with the description, discovery and integration services. However as explained above traditional service registries support solely the syntax of service interface specifications and do not capture service semantics. Capturing the service semantics is one of the most important plug-in tasks and will be used in both “locating appropriate services” and “ranking the competitor services”. The W3C’s “Semantic Mark-up for Web Services” (OWL-S) defines standards to capture the functional description of the service in terms of the transformation effected by the accordant service. Specifically, it specifies the inputs required by the service and the generated outputs. Furthermore, a service may require external conditions to be satisfied, and it has the effect of changing such conditions, the profile thus describes the preconditions required by the service and the expected effects that result from the execution of the service. For example, a selling service may require a valid credit card as a precondition. It then requires the credit card number and the expiration date as input, and generates a receipt as the output.

Despite all advantages of SOA and Semantic Web Services, they are complex and cost-intensive technologies and not suitable for non-IT professionals. They are complex technologies which need a thorough understanding of these technologies, their underlying standards and programming capabilities. In other words, there are

not suitable for end-users who lack these capabilities. Another shortcoming of SOA-based approaches is their inability to react rapidly to the changes in the environment, as implementation of such systems are cost and time intensive and any changes in the environment may require some modifications in the system. An important characteristic of systems for end-users is their capability for personalization and customization. Most of today's business processes and SOA-based solutions are still mainly designed to satisfy the mass of users. Unfortunately customization and mass generic production are at odd with each other. As mentioned before, IT solutions typically focus on 20% of user requirements that affect the most users and the long tale of requirement is usually ignored by IT providers. The issue is even more critical in case of Semantic Desktops and variety of users with different requirements. It is too difficult for an ordinary user to take benefit of the available services by composing the appropriate services together. For this reasons, SOA-based approaches are mainly in the hands of IT-departments of big firms who have a complex stack of technologies to realize SOA-based scenarios. End users require simple, cost-effective techniques which enable them to design solutions in an ad-hoc "quick and dirty" manner.

2.3 Web 2.0 and Rich Internet Applications

Recently the Web 2.0 has set a new trend in Rich Internet Application (RIA) world. It makes better use of the client machine's processing power and at the same time pushes the SOA paradigm to its limits. At the moment most SOAs are conceptually trapped inside an organizations' intranet and Web 2.0 envisions building collective intelligence and mashed up functionality based on web services. In this environment, Internet will play the role of a global operating system that hosts the web services. In other words the Web 2.0 is a step toward the global cloud computing idea where business services are presented on Internet and developers should select and weave them together to create new compound services.

Thus Web 2.0 is much more than adding a nice facade to old web applications, rather it is a new way of thinking about software architecture of RIAs. In comparison to traditional web applications, the application logic of modern Web 2.0 applications tends to push the interactive user interface tasks to the client side. The client

components on the other hand negotiate with remote services that deal with user events.

A set of six key business applications are motivating overall RIA spending, consisting of enhancement of existing web applications, high-transaction and event-driven Internet applications, next-generation portals, enhanced business intelligence solutions, application modernization, and peer-to-peer or mashup solutions. Market analyzers expect spending on each of these areas to increase rapidly over the next three years, exceeding $500 million by 2011 (Schmelzer, 2006). On the other hand information sharing in RIA collaboration environment adds new dimension to web application security. The most "Web 2.0"-oriented, exist only on the Internet, deriving their effectiveness from the inter-human connections and from the network effects that Web 2.0 makes possible, and growing in effectiveness in proportion as people make more use of them (O'Reilly, 2006). As a result a huge amount of information is created by this group of Web 2.0 applications that needs to be managed in a machine-processable way. An interesting use case of this approach is inter-organization trust where lots of entities such as social networks and weblog entries are analyzed and enriched semantically to assess a trust indicator for organizational cooperation. The same method can be also used by individuals to make a self-test of their web 2.0 contributions and find out what inferences will be derived from their web presence.

Unfortunately the content description methods are not being used by all content owners and the Web 2.0's Achilles' heel in my belief is the lack of semantic information that can be used to link this huge amount of information efficiently and this is the reason that some web specialists are expecting another web which is called Web 3.0 to complete the deficiencies of current web. Without the explicit semantic context, the process of data analysis and putting the data to work in business process safely is still unthinkable without significant human involvement and exactly at this point, Semantic Web can fit, to make the data machine-processable.

2.3.1 Mashups

Web 2.0 has also introduced new possibilities for a better human computer interaction via rich applications such as Mashups that provide a user-driven micro-

integration of web-accessible data (Reliability, 2009). The term mashup originates from music industry where a song or composition is created by blending two or more songs, usually by overlaying the vocal track of one song seamlessly over the music track of another (Music Mashups, 2009). Mashups owes its popularity and fast improvements to its two basic blocks namely Web 2.0 and SOA. Mashup envisions building effective and light-weight information processing solutions based on the exposed Web Services of organizations. Such Web Services may range from simple services such as RSS (RSS, 2009) and REST (REST, 2000) based services to complex BPEL services for more serious use cases; however the Mashups benefits for the latter use cases is not yet known to IT decision makers (Anjomshoaa, 2009). According to market research reports, this situation is going to change quickly in the coming years. Mashups are identified among top 10 strategic technologies for 2009 (Gartner, 2008) and it is expected that by 2012, one-third of analytic applications applied to business processes will be delivered through coarse-grained application mashups" (Gartner, 2009). The power of mashups is also being examined in real world information management scenarios and has attracted many attentions (Hoyer, 2009). Mashups can be applied to a broad spectrum of use cases ranging from simple data widgets to more complex use cases such as task automation and system integration. The mashup applications are roughly categorized under the following groups:

- Consumer Mashups (Presentation Mashups): are the simplest group of Mashups that are used to facilitate the creation of information portals from different resources for presentation purpose. This group of mashups have the lowest degree of customization and are usually implemented as pre-built widgets that can be added to user interfaces. A well-known example of consumer Mashups is iGoogle (iGoogle, 2009) which is a personal web portal with capability of adding Web feeds and Google Gadgets such as email, scratch pad, news, weather, etc.
- Data Mashups: which are used to integrate data and services from different resources such as Web Services, RESTful APIs, Web Extractors, RSS Feeds, etc. These kind of mashups aim to facilitate the data access and cross-referencing between resources. They may also benefit from presentation

gadgets or visualization APIs to deliver the results. Nowadays a handful of mashable resources are available on the Web and plenty of useful Mashups have been created using them. A good example of data mashups is geo-based Mashups that integrate the information from different resources into Web-based maps such as Google Maps (Google Maps, 2009) or Yahoo Maps (Yahoo Maps, 2009). Figure 2.2 shows the top APIs for Mashups and their percentage of utilizations in community-created mashups (Programmable Web, 2009).

Figure 2.2: Top APIs for creating Mashups

- Enterprise Mashups (Logic Mashups): always involve programming and therefore are the most complex mashup category. They connect two or more applications, automate certain tasks and include awareness of workflow (Mashup Basics, 2009). Enterprise mashups usually depend on some server-side components and compete with data integration and service orchestration technologies such as BPEL and Enterprise Service Buses (ESBs).

Among the introduced categories of mashups, Enterprise Mashups have gained momentum during the last years. By empowering actual business end-users to create and adapt individual enterprise applications, Enterprise Mashups implicate a shift concerning a collaborative software development and consumption process. Upcoming Mashup tools prove the growing relevance of this paradigm in the industry, both in the consumer and enterprise-oriented market. Market research companies like Gartner (Gartner, 2008), Forester (Forester, 2008) or the Economic Intelligence Unit (EIU, 2007) forecast a growing relevance of this paradigm in the next years. (Hoyer, 2008a).

To effectively adopt the Enterprise Mashups in organizations the following “Five Cs” principle is used to identify the different enterprise requirements (Mashup Guide, 2008):

- Consume: Users need to consume public and private Web-enabled SOA-style services and mashups on demand.
- Create: Users need to create new enterprise mashups from existing SOA-style services and mashups, preferably in a visual editor.
- Customized: Users need to customize (e.g. filter, annotate, etc.) existing mashups and create variants which can be published visually in standardized user-interfaces such as portals.
- Collaborate: Users need to tag, describe, publish, and share their mashups with others in their community.
- Confidence: Users need to be confident that all mashup consumption, creation, customization and collaboration occur in a secured and governed environment.

Today organizations are confronted by business pressures to decrease costs, reduce workforce and transform their business from an internally focused organization into a service oriented and customer centric organization, which target the need of their customers. Therefore organizations have to deal with different domains and bring them together which in turn means also dealing with lots of common business process concepts and architectures.

SOA (Service Oriented Architecture) and RIA (Rich Internet Application) are leading the standardized access to business functionality and data with desktop-like interaction over Web. But for users like non-IT experts, it's really difficult to use these new technologies to improve their daily business. Companies, who are using and "living" the SOA approach to get faster business responds and cost reducing effects, have the problem to manage and provide information on how the services interact and whether they are used in a right way. Mashups enable users to get access to data sources through SOA by a user-driven Process with short development cycles for new Mashup applications.

As the main contribution of this thesis, Semantic Mashups will be introduced in the next chapters. This novel concept is considered to be a key enabler for information integration solutions in business use cases.

2.3.2 Mashups Providers

The power of mashups is also being examined in real world information management scenarios and has attracted many attentions. The best proof for this assertion is the various Mashups solutions that are released by main IT players such as Google, IBM, and Microsoft. Mashups are also in focus of different research programs such as FAST EU FP7 project (FAST, 2008) that aims at providing an innovative visual programming environment that will facilitate the development of next-generation composite user interfaces. It constitutes a novel approach to application composition and business process definition from a top-down user-centric perspective.

In this section a short survey of main industrial mashup solution-providers will be provided.

One of the Web2.0 pioneers in the last few years is Google. Google started with a native web application – a revolutionary search engine – which was delivered as a service to the public. At this time Google was not focusing on Web2.0 and therefore did not put too much emphasis on topics like web standards, social communities or web services.

In the course of years Google come up with a handful of user-centric Web2.0 projects such as Blogger, online collaboration tools, Google Earth, etc. This mushrooming of projects led also to Mashup solutions with a focus on the web user. Application areas of their Data Mashup solutions are Google Maps and Google Mashup Editor (which is deprecated and will be merged into the Google App engine). Especially Google Maps can be seen as the initiator of Data Mashup trend. Users got easy and intuitive access over their web browser to an amazing application and move around on the map. For this reason users wanted to have also access to company data, data from other sources, feeds, services, etc over web and as easy as Google Maps.

Companies like Yahoo with Yahoo! Pipes (Yahoo Pipes, 2009) and Microsoft with Microsoft Popfly (Popfly, 2009) developed also web based Data Mashups. These Mashup solutions are really easy to use via a visual programming environment to integrate data from different sources like WebPages, RSS feeds or web services. With Microsoft Popfly (in BETA phase) users can in addition to Mashups also create web pages and games and share them with other users. Popfly is based on the Silverlight technology but for presentation reasons, besides Silverlight also AJAX and DHTML are used. With Mashups, Microsoft also tries to expand the Windows personalization features by giving the user the possibility to embed Mashups in the Windows Sidebar. Therefore users can get access to the Popfly Mashups directly through their personalized desktop environment.

A newcomer to the market of Mashup solutions is Intel with the Intel Mash Maker (Mash Maker, 2009) product. Intel has developed an innovative service which allows users to extract content from websites and merge them with the Intel Mash Maker solution in a user friendly and intuitive way. The Intel Mash Maker can be installed as a browser plug-in on all common web browsers in the market. This integration to the web browser brings personalization possibilities directly to the end-user. Especially using the existing websites and enriching them with new and personalized information is a strengths point of Intel's solution.

Compared to Presentation and Data Mashups, Enterprise Mashups have the pretension to deliver content and services with the necessary grade of automation. Enterprise Mashups solutions have to provide also an easy way of collaboration and documentation with adequate guidance. JackBe (JackBe, 2009) is one of the leading Mashup Solution providers in the business market. With their Enterprise Mashup solution Presto, they are focusing on enterprise customers, especially companies with an affinity to Web2.0 and the necessary enablers like SOA and RIA. JackBe’s solution is not considering the personalization and user-centric aspects of mashups. JackBe has a data-driven approach by offering a lot of integrated data connectors. The server-side mashup components facilitate the exchange and reuse of created Mashups for internal and also external use cases in a secure and trustworthy way. Serena, playing in the same league as JackBe, is more focusing on the user-centric aspects of mashup solutions via their Mashup Composer application. Serena’s

solution enables users to design Mashups via a graphical interface to automate business activities.

Another important player in this market of enterprise Mashup solutions is IBM. Recently IBM has developed a number of Web2.0 and Mashup products such as QEDWiki (QEDWiki, 2009), Lotus Mashup, etc. Lotus Mashup is an enterprise-grade Mashup editor developed and distributed by IBM as part of the IBM Mashup Center solution (Mashup Center, 2009). Parts of the suspended QEDWiki are also integrated into the IBM Mashup Center. The main focus of IBM lies on integration of different data sources and creating the so called widgets and sharing them for reuse by other end users.

Some other Mashup providers such as Kapow technologies (Kapow, 2009) have focused on Web Intelligence solutions that cover features from all three kinds of mashups. Kapow Mashup Server offers access to different data sources and services where users can easily access content and analyze data on their desktop. Especially monitoring and management tools can be integrated with the necessary scalability, high availability, security and automation.

The evaluation matrix (Table 2.1), delivers a survey of Mashup products according to the different functionality features of the three mashup categories introduced.

Mashup Type	Feature / Product	IBM Mashup Center	Serena Presto	JackBe	Yahoo pipes	Dapper	Intel Mashup Maker	Microsoft Popfly
Consumer	Visualization Widgets	✓	✓	✓	✓	✓	✓	✓
	Screen Scrapping	✓	✓	✓	✓	✓	✓	×[a]
Data	Visual Data Assembly	✓	✓	✓	✓	✓	✓	✓
	Use Web 2.0 Resources	✓	✓	✓	✓	×	×	×
	Advanced Data Aggregation	✓	✓	✓	✓	×	×	×
	Advanced Data Flow	✓	✓	✓	✓	×	×	×
Enterprise	Workflow	✓	✓	✓	×	×	×	×
	Enterprise Integration	✓	✓	✓	×	×	×	×
	Server-based	✓	✓	✓	×	×	×	×

a. Microsoft Popfly can indirectly support screen scrapping via embedding Dapper artifacts

Table 2.1: Comparison of Mashup Products

The result of the above table is summarized in Figure 2.3 which clearly shows the priority of introduced mashup categories for different mashup providers.

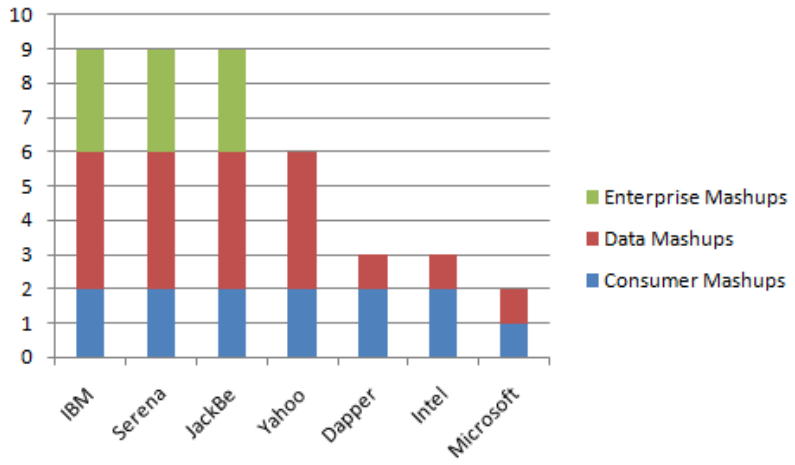

Figure 2.3: Mashup product classification

2.4 Security and Privacy Requirements

The security and privacy use cases have the potential to give a significant boost to the semantic technologies. Combination of Semantic Web technologies and the power of Web 2.0 content can be used for assessing general reliable indicators for security and privacy scenarios. It is worthy to note that the “general trust” is a key factor for inter-organizational trust scenarios. While there are efficient ways of measuring trust in individual systems, it is still a big challenge to aggregate such

information across systems and make an overall inference. For instance number of edits that are made by members of a specific organization in Wikipedia and the further corrections to the contributed data can be considered as a good indicator for reliability of that organization in Wikipedia, however there is no straightforward way to aggregate this information with the contributions on Flicker and YouTube to make a general inference about organization's trustworthiness.

On the other hand, the disclosure of personal information is an emerging demand of today's industrialized world where sharing information for different purposes with organizations around the world is efficient, productive and nearly essential. Unfortunately, there are various different, unstructured ways to develop privacy and security policies, and as a result there is no single standard solution for secure and efficient interaction between end-users and organizations. Designing transparent, usable systems in support of personal privacy, security, and trust includes everything from understanding the intended use of a system to users' tasks and goals, as well as the contexts in which the users will use the system.

Obviously new security and privacy schemas are required to cover the requirements of Web 2.0 applications which are being raised due to the following reasons:

- Web services are the building block of Web 2.0 applications and freeing the web services from organization environments, make it necessary to have appropriate information disclosure and information usage policies.
- Web 2.0 has made the content creation much easier and as a result a huge amount of data is created by people every day. The volume of the data on the web is doubled since the emergence of Web 2.0 technologies. The data mining in the user generated entities and extracting the derived knowledge and information patterns is the new threat to privacy of individuals. We would need more elaborated methods for analyzing the web contents in order to facilitate data sharing and data reuse in a trustworthy and efficient way. Moreover targeted data mining in the web data might be helpful in some use cases such as inter-organization trust.
- As mentioned before, the Web 2.0 architecture is tending to utilize the client-side processing power. This attribute of Web 2.0 can be used intelligently for

better integration of user data with global business processes. In other words user desktop can interact with the real world processes and provide the requested data without human interaction. Before such dreams can come true, we would need an efficient mechanism to define user's security and privacy policies.

2.5 Information Integration Use cases

In this section, some categories of information integration scenarios will be presented that are either not fully supported by existing technologies or needs a complicated stack of technologies that in practice make it impossible for novice users to realize the scenarios by their own efforts. Some of these use cases will be revisited once more in the next chapters of this thesis, to show how the presented solution of this thesis can facilitate information integration for end users.

2.5.1 Personalized Services

Now a day, the end users can create a personal information repository for themselves in different ways. The personal information store can range from a simple excel sheets that keeps the detail of user's contact people to more sophisticated methods such as Semantic Desktops. Disregarding the repository's technical aspects and size of solution, they are generally used to answer the queries that user is facing in his/her daily activities. Nevertheless to extract and deliver the required information, either end user should be familiar with the structure of underlying data and corresponding query schema, or queries will be limited to the system predefined queries. In order to answer more complex queries, end users should be able to create situational solutions based on the confronted circumstances. Such situational solutions can be seen as personal-services that can be shared and used by trusted contacts or business processes. For instance a user may create a personal service that provides a list of his/her publications that contain particular keywords. This specific personal service can be then called for each member of an online community and be accumulated to create a ranked list of people who are interested in a specific research topic.

A more complex example could be an online shopping use case (Figure 2.4), where the price limits should be first tested against user's bank account and credit

information. A personal service is able to call the external bank web services in a trusty way and calculate the user's shopping limit based on user's cache amount in the banks and the planned monthly loans. It is also important to note that the security and privacy perspective are vital parts of such use cases and none of the user information should be disclosed unnecessarily during the interaction with business processes.

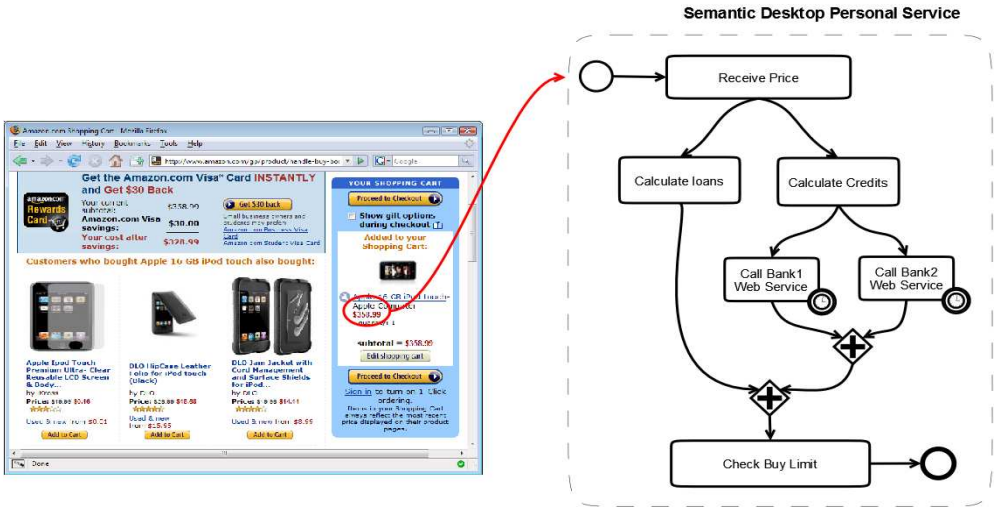

Figure 2.4: Personalized Services

Personal Services that are some customized web services, available on end-user's device, are closely related to the entities that are typically used in everyday life such as emails, appointments, web pages, etc. These services are customized for their owner and not only vary from user to user; they also vary depending on the context of use.

2.5.2 Personal Services for People with special needs

Despite the advances in design and implementation of web applications, the human interaction with the applications is still a frustrating experience for most of the users. The user with "normal" cognitive abilities is able to combine the application's logic and goals with his / her experiences and knowledge to accomplish the use of Internet applications. On the other hand, the users with severe cognitive limitations either face difficulties in identifying the data model behind the web form, or cannot map the model to their knowledge models. Moreover the recent Web 2.0 applications are

not supported by assistive technologies such as screen readers and this has made the web interaction of people with disabilities more difficult.

To address these requirements, some international initiatives have proposed standards and guidelines to make the information accessible to people with special needs. One of such recent initiatives is the Web Accessibility Initiative for Accessible Rich Internet Applications (WAI-ARIA, 2009). In spite of such standard and guidelines a high percentage of web content is not yet accessible and the content authors are moving very slowly toward WAI-ARIA goals.

As a result it is necessary to have a temporary and personal solution for people with special needs for the transition phase from traditional web to modern web pages that support accessibility features.

2.5.3 Global Business Processes

We are daily dealing with some processes that need parts of our personal information to complete. The fact is that most of this personal information are scattered among files, emails, web pages, etc and we manage to feed them into the right process at appropriate time using our memory and reasoning power.

As an example consider the scenario of planning for a vacation. The typical flow of this process is depicted in Figure 2.5.

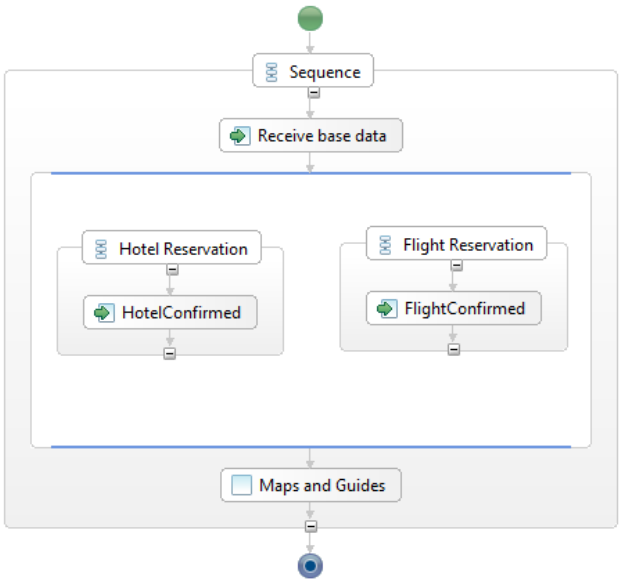

Figure 2.5: A typical trip planning process

Each step in this process can be equipped and supported by some personal life items like emails, photos, web pages, etc. Additionally in such processes a data exchange between the global processes and our personal information repository is necessary to facilitate data contribution to business processes.

Chapter 3

Semantic Desktop

Today there is a handful of semantic applications such as semantic navigation & search tools, annotating and authoring tools, semantic calendars, annotated photo albums, and semantic wikis. In this context, Semantic Desktop is a trend of software programs that are aiming at realizing the goals of "Memories for Life" grand challenge. The grand challenge has expanded the scope of Memories for Life and outlined it as a multidisciplinary problem that should address issues such as knowledge and databases, information retrieval, security/privacy, and human-computer interaction. A more concrete definition of Semantic Desktop has been formulated as follows (Sauermann, 2005):

> *"A Semantic Desktop is a device in which an individual stores all her digital information like documents, multimedia and messages. These are interpreted as Semantic Web resources, each is identified by a Uniform Resource Identifier (URI) and all data is accessible and query-able as RDF graph. Resources from the web can be stored and authored content can be shared with others. Ontologies allow the user to express personal mental models and form the semantic glue interconnecting information and systems. Applications respect this and store, read and communicate via ontologies and Semantic Web protocols. The Semantic Desktop is an enlarged supplement to the user's memory."*

As stated in this definition, the Semantic Web and ontologies play a significant role in Semantic Desktops and has accelerated the research toward realizing the dream of "Memories for Life" by modeling a diverse range of memory items and their associations. Today, there is a number of Semantic Desktop solutions, such as SemanticLIFE (developed at the Vienna University of Technology), Blackman (Blackman, 2007), Gnowsis (Gnowsis, 2009), and IRIS (IRIS, 2009) that are to some extent capable of organizing and managing the personal life items. However, their application is limited to the storage and retrieval of various personal items. The next step towards the goals of the Memories for Life is extending the footprint of such memories for interactions with global and social network services.

One of the earliest Semantic Desktop implementation is the Gnowsis system (Gnowsis, 2009) which consists of two parts: the Gnowsis server that performs the data processing, storage and interaction with native applications; and the graphical user interface (GUI) part, implemented as Swing GUI and Web-based interfaces. External applications such as Microsoft Outlook or Web browsers are integrated using standardized interfaces. The upper ontology of the personal information management (PIM) ontology, used by Gnowsis, is assumed to reconcile different models for information to make that information available over a common conceptual model. However, The Gnowsis system lacks a formalism to represent context and the conceptual view determined by the upper ontology and cannot be changed to represent the reconciled information in different contexts as required by different tasks.

A similar Semantic Desktop system is the Haystack project (Haystack, 2009) which aims at connecting application data and let people manage their information using personalization; however, the Haystack client is a rather complicated and extensive application. Haystack supports several PIM tasks and is extensible by a plug-in mechanism that allows to implement new functionality and to integrate new information. Similar to the Gnowsis prototype, Haystack allows to associate different pieces of information to each other, organize them, e.g. in collections and reuse them in particular Haystack modules. Plug-ins, however, implement their own, application specific models that do not necessarily conform to the conceptualization of existing and future Haystack plug-ins.

Both projects, Gnowsis and Haystack, enable the linking of information and the classification of that information by user defined ontologies. Nevertheless, they miss means for context representation and the broad conceptual scope that allows to integrate and to inter-relate information managed with arbitrary PIM tools. They lack a profound conceptual model that enables the reuse of information across contexts and that provides the extensibility to integrate new applications that require new contextual views onto existing information. Support for information reuse and a reconciliation of PIM applications is limited to applications that do not require different conceptual views as those would conflict with domain models of these applications.

A slightly different approach has been followed by the Nepomuk project (Gorza, 2007) which is aiming to enhance data sharing and exchange across social and organizational relations and creating social Semantic Desktops. The social Semantic Desktop will support the personal aspects of knowledge work by developing tools for knowledge articulation and visualization, the interfaces and data structures of the personal Semantic Web, and integration of support for personal work processes. It supports the social aspects of communication, distributed collaboration and social exchange by providing solutions for distributed search and storage and of semantic social networks and knowledge exchange (Nepomuk, 2009).

Our SemanticLIFE project (SemanticLIFE, 2004) is another effort to implement a PIM system over a Human Lifetime using ontologies as a basis for the representation of its content. In the framework of the SemanticLIFE project, we have built a semantic repository of lifetime personal data from a variety of sources such as emails, contacts, running processes, Web browsing history, calendar appointments, chat sessions, and other documents. This PIM system acts as a digital memory and provides an ontology-based profile for users.

In addition to the acquisition, annotation, and storage of data, SemanticLIFE also provides an intuitive and effective search mechanism based on the stored semantics.

The whole SemanticLIFE system has been designed as a set of interactive plug-ins that fit into main application and this guarantees the flexibility and extensibility of the SemanticLIFE platform. Communication within the system is based on a service-

oriented design with the advantage of its loosely coupled characteristics. The Service Oriented Pipeline Architecture (SOPA) [1] has been introduced in order to compose complex solutions and scenarios from atomic services of SemanticLIFE plug-ins. The SOPA solution is one of our basic approaches for implementing the information integration of various information resources. Due to the significant role that SOPA plays in SemanticLIFE framework, it will be explored with more details in the succeeding section.

SemanticLIFE has been developed using the Eclipse Rich Client Platform (Eclipse RCP) technology following the industry standard Eclipse IDE. Eclipse RCP offers several advantages over traditional Java rich client applications. From its plug-in architecture, our project has been benefiting the most. The whole SemanticLIFE system has been designed as a set of interactive plug-ins that fit into the main RCP application. To clarify the system architecture and information integration approach, more details on some plug-ins of SemanticLIFE application will be provided in the next section.

3.1 SemanticLIFE Architecture

In this section the overall architecture of SemanticLIFE framework will be presented and the functionality of its basic plug-ins will be explored in more details. Figure 3.1 depicts the architecture of SemanticLIFE framework that has been used to realize PIM use cases.

[1] SOPA Framework has been selected among top 10 finalists of Jax Innovation Award in 2006 together with other notable nominations such as Spring Framework and Rich Ajax Platform (http://jax-award.de/jax_award06/nominierungen_en.php)

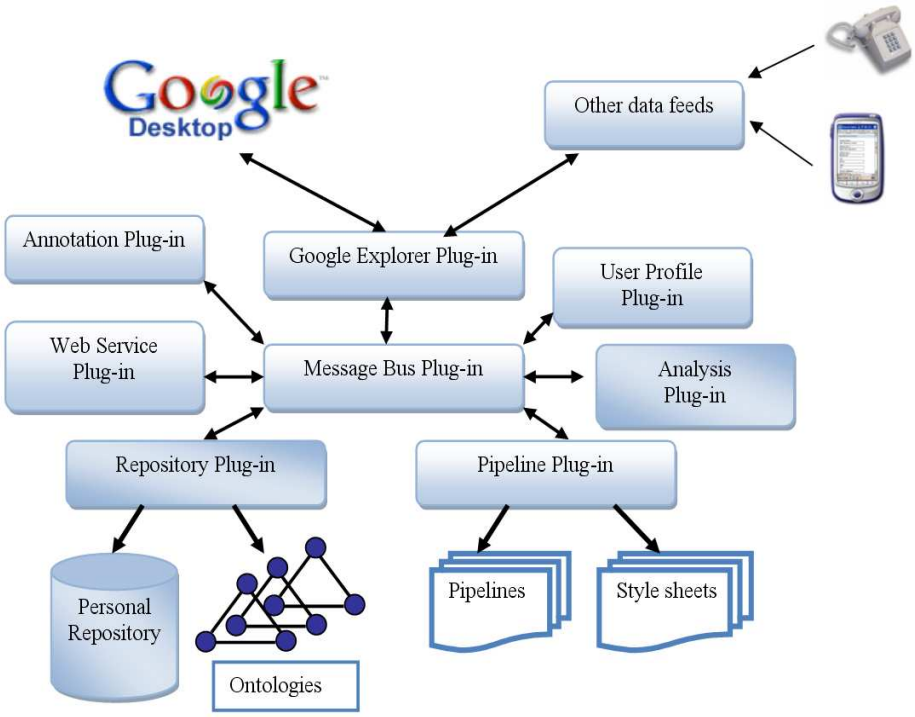

Figure 3.1: SemanticLIFE's system architecture

3.1.1 Data Feeds and Semantic Store

An important component of SemanticLIFE framework is data acquisition which is responsible for acquiring the life items such as emails, browsed web pages, files, etc. At the early stages of project development, the project team started to build multiple connectors and data processors for various data feeds. The result of these attempts was realized as several data connectors such as Firefox plug-in for reading browsed web pages, a Microsoft Outlook connector for accessing the emails and calendars, and a file system watchdog to monitor the altered files. After a while we found out that this approach will be a never-ending cycle of programming which is necessary to support the new data feeds and new version of applications. To get rid of such issues, we made a strategic change and decided to use one of existing desktop search program. In this way, the search engine will be responsible to deal with the complexity of data feeds and the Semantic Desktop solution will merely focus on semantifying this information.

For this purpose we have selected Google Desktop as desktop search solution and added a specific plug-in for data import from Google Desktop's repository. In this way the Google Desktop plug-in captures the user information items that can be of

various types such as browsed web pages, emails, chat, images, audio or video streams, etc and hand it to the SemanticLIFE system. In SemanticLIFE system these information entities are associated with one's lifespan entities and are then stored in an ontological way according to the already established RDF metadata. This will then facilitate semantic queries, life trails and processing of life events.

Furthermore, this established set of information items enables the SemanticLIFE system to rank and filter the queries and events based on user interest and preferences. Unlike the desktop search programs which are mostly based on full-text content indexing, the SemanticLIFE framework is aiming at taking into consideration the dynamic, long term activities of user to find out the user interests which may be even change from time to time. The information captured by this plug-in can later on further be refined and annotated by the user in order to make a more precise user behavioral model.

3.1.2 Data Services and Messaging

In the SemanticLIFE architecture, the Web Service and Message Bus plug-ins are the two core components that are responsible for messaging and communication with internal and external services. In this section these plug-ins will be explored.

The Message Bus plug-in provides an infrastructure for plug-in interactions. All SemanticLIFE processes and their corresponding information exchanges are designed via a Message Bus plug-in. SemanticLIFE plug-ins that need to negotiate with other system components should register themselves to this backbone by extending the Message Bus extension point. Then at runtime all messages and calls will be routed to relevant system services that are backed by the implemented plug-in collection. Another important functionality of Message Bus is to supply a level of abstraction between system services which greatly improves the flexibility and transparency of system use cases; i.e. all system-wide offered services including external web services, business functions and data analysis services will be served uniformly by this plug-in.

The second plug-in, Web Service plug-in, is on the other hand responsible for providing a uniform access layer to internal and external services and their semantics. Internal system services include some SemanticLIFE-specific composite

services that are called pipelines and other services provided by system plug-ins. External services can be plugged at anytime to the SemanticLIFE system by locating the corresponding web service configurations (URL of the WSDL files).

Ideally the services offered by the Web Service plug-in should have a machine interpretable description to automate the service discovery and service composition scenarios in an unambiguous way. For achieving certainty and machine-interpretability, semantics are added to Web Services to explicitly present requirements and capabilities. The Web Service plug-in is aimed to manage the web service annotation and defining the meaning of terms, such as the meaning of parameters or business objects and the meaning or intent of an operation. This information will be extremely helpful for running semantic search phase in which the services' semantics and the corresponding input/output parameters are evaluated to match a specific query. For example we may search the repository of all services that accept a country name and provide the list of touristy cities of that country. Finding such services will be a more or less straightforward task, provided that the country and city concept are matured in the system's ontology and also service parameters are annotated with the domain ontology. The Web service plug-in offers the following three categories of services:

- Service-finder services: these services are responsible for finding the appropriate services for a specific request. The return-value of such a service is a list of services ranked by the user preference. An example of a service-finder service is a service that provides a list of web services for hotel reservations in a specific city.
- Service-invocation service: This service invokes the requested service using the SemanticLIFE platform service calling mechanism that is mainly implemented in Message Bus plug-in. Based on the requested service type, the invocation mechanism may call an internal pipeline, an internal service or an external web service.
- Semantic-recommender service: This service will invoke the recommendation pipeline of SemanticLIFE for a given service. The pipeline in turn will invoke that service, semantically rank and enrich the service-result, provide a ranked list of options for user choices and finally return the selected item. For

example, when calling the semantic-recommender service for a hotel-finder service, the corresponding SemanticLIFE pipeline firstly runs the web service (hotel-finder), then ranks the results and finally display a selection list to the user. A user will select his/her choice from the list and this value will be returned to the system to further continue with the execution of business processes.

In addition to services that are available on Internet as web services, the user's desktop can also provide some web services that are based on personal information. As explained in the previous chapter, Semantic Desktop systems such as SemanticLIFE contribute some personal services that can be used in service composition scenarios. These services are enriched with semantic information and can be queried based on the service parameters and/or service intent.

To clarify the issue, consider a web page rendering service that is provided by the user's Semantic Desktop. A non-semantic service will simply render the web pages according to some predefined formatting or logical rules. The drawback of a “non-semantic” service is that the service is not able to conceive the web page content and instantly visualize it according to user preferences and restrictions. On the other hand the semantic personal rendering service may perform more sophisticated tasks such as highlighting the items and connect them to user's history items. Such personal rendering services are especially helpful for memory impaired users that cannot remember their previous interactions about specific subjects. Another use case of personal services is applying the appropriate style to the web pages according to the end-user's visual restrictions for sight impaired users.

Personal services can be also categorized according to their internal complexity. Some personal services perform a simple one step action whereas more complex services may be composed of multiple actions and additional conditions and service calls. BPEL processes fall under the category of more complex processes. A Semantic Desktop that supports BPEL processes might be seen as an information resource that extract and process the data about from personal items or appropriate services and hand in the useful information to external world.

3.1.3 User Profile

The SemanticLIFE framework is equipped with a dedicated plug-in to manage the user profile consisting users' static and dynamic profile. Part of the user profile such as the user demographics, user interests, contacts etc. can be considered as the static part of the profile. However in the long term these data will be elaborated and enriched either by automatic or manual annotation and additions. For example the user interests may be ranked by the number of relevant web pages and other items that are tagged for each interest item. So after a while of user monitoring, the system can infer if the user is more interested in football rather than skiing due to the number of related items that are attached to each interest item. Figure 3.2 shows a fragment of the user model used in the SemanticLIFE system.

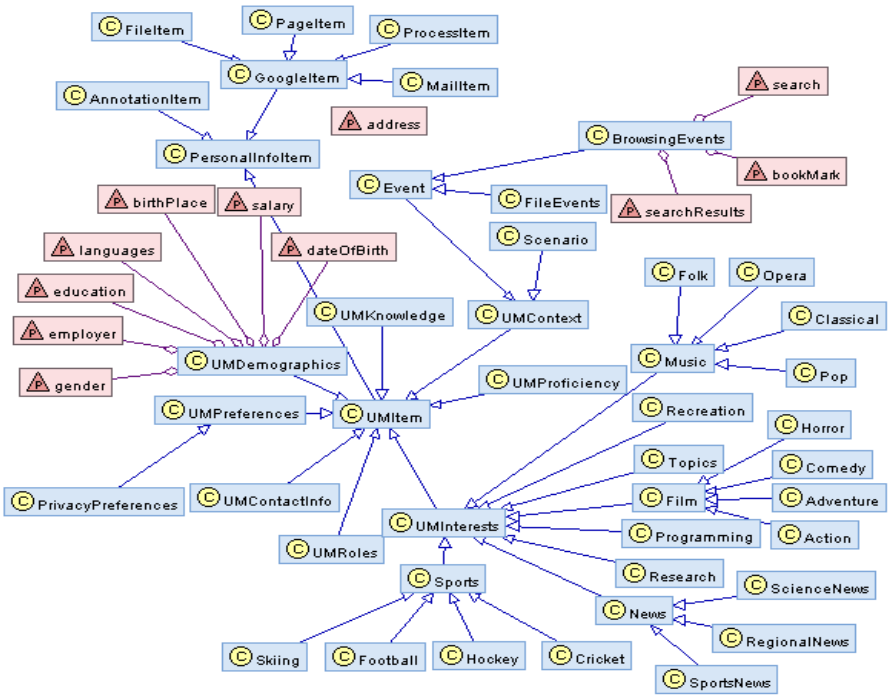

Figure 3.2: User model in SemanticLIFE framework

This schema will help the system to create a matching behaviour model for user and enhance the user modelling in the following ways:

- User will have a unique profile that can be reused for many business processes.

- User model is dynamic and will be adjusted based on the long term user interactions.

3.1.4 Collaboration and Information Sharing

The common limitation and a major shortcoming of Semantic Desktops approaches is the fact that their scope is limited to users' desktops and the precious semantic information are not yet effectively used in business processes and tasks that people are dealing with in their daily life. One of the basic goals of the SemanticLIFE project was to leverage the application of Semantic Desktops, beyond the rather complicated semantic queries and putting the information to work for their owners in a trustworthy way. It is important to note that trust plays an important role in the integration of information spaces; however applying trust in information integration scenarios is not always straightforward and usually there are many intangible and project-specific aspects that should be considered for a successful trust implementation. Basically trust can be gained from a person/agent's own experiences with an entity and has the following characteristics:

- Context specific: Trust is context specific and depends on the (different) roles of the entity.
- Multi-faceted: Even in the same context, there is a need to develop differentiated trust in different aspects.
- Dynamic: Trust can increase or decrease with further experience.

There are multiple approaches for implementing and applying trust to IT systems:

- Centralized & decentralized systems: Whether a system is centralized or decentralized determines the feasibility and complexity of a trust and mechanism. In a centralized system, a central node will take all the responsibilities of managing reputations for all the members. In a decentralized system, e.g. a peer-to-peer system, there is no central node. The members in the system have to cooperate and share the responsibilities.
- Person & Resource based systems: Systems can be also classified as person systems or resource systems. In person systems, the focus is on modeling the

trust of people or agents, acting on behalf of people. In resource systems, the focus is on modeling trust of resources, which could be products or services.

- Global and personalized systems: In global systems, the reputation of a person, agent, product or service is based on the opinions from the general population, which is public and visible to all the system members, while in personalized systems, the reputation of a person, agent, product or service is built on the opinions from a group of particular people.

Formulating the business policies in formal ways, is one of the important approaches for applying trust in IT systems. There are number of candidate policy implementation languages like SWRL (SWRL, 2004), KAoS (KAoS, 2003) and REI (REI, 2003) and also there are number of policies that are needed to facilitate the information interchange in a collaborative environment. Users define these policies for some specific operation, e.g. project resource sharing policies, project member access policies, stakeholder access policies etc. Take an example where the Person-P1 asks for documents related to a project. Person-P1 can be granted access if he is member of that project and his status matches to the confidentiality of that document.

Figure 3.3 below shows different components of SemanticLIFE and how policies and filters are used to control the information flow between SemanticLIFE-enabled workstations. Basically the information flow on the user desktops will be affected by domain ontologies that are defined for each collaboration environment separately and include the abstraction of the surrounding business entities. Based on these business entities, users and organization will setup their information sharing and information filtering policies. Finally the service invocation mechanisms that were explained before will take care of information exchange at lower levels.

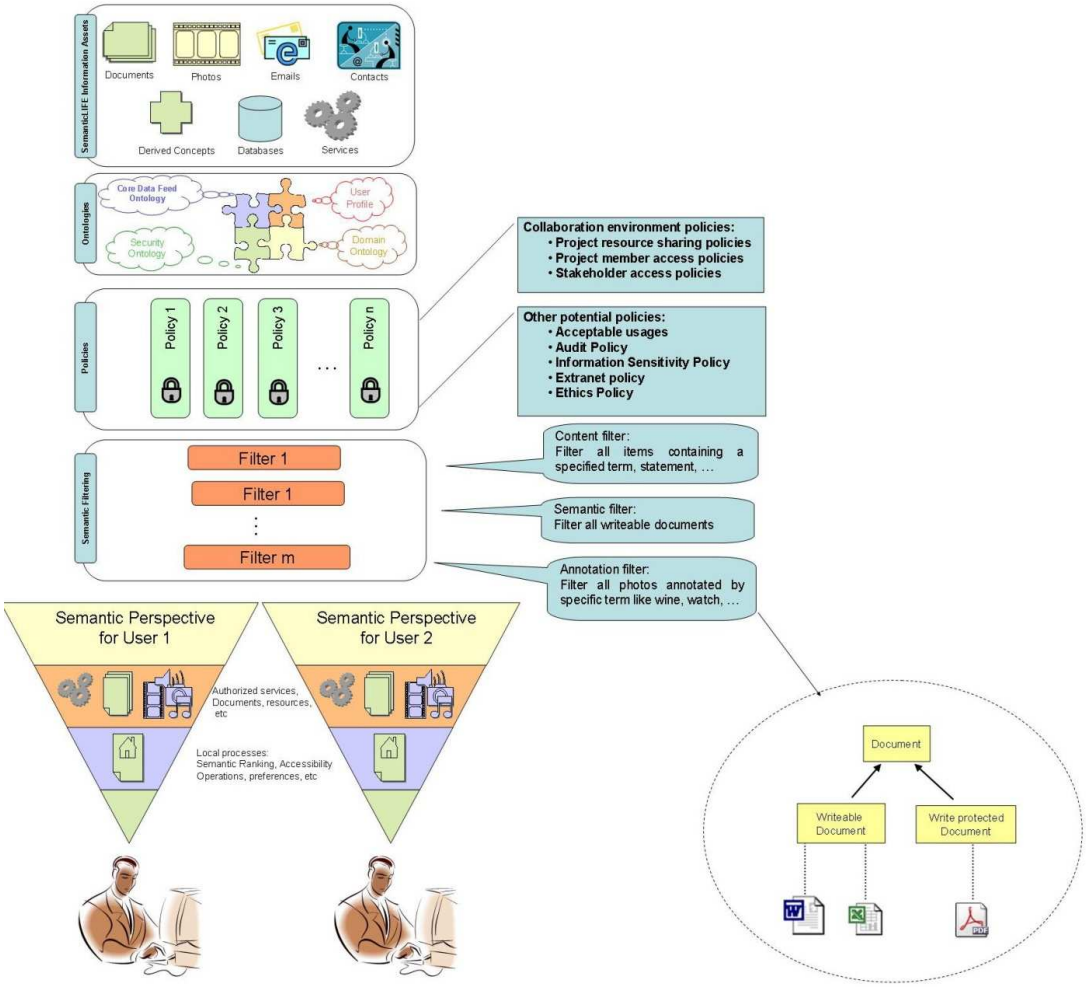

Figure 3.3: SemanticLIFE collaboration model

3.2 Information Integration in SemanticLIFE

Designing complex and large business processes requires a language that supports component integration and process automation. Service Oriented Architecture aims to address these requirements which are proved to be two of the most important issues facing the organizations today. In the context of SemanticLIFE framework, we developed our own service composition and information integration framework which is called Service Oriented Pipeline Architecture (SOPA) and provides a paradigm to describe the web service compositions as pipelines. The proposed SOPA framework is mainly discussing the enabling components of the SOPA systems for Java and particularly Eclipse developers community. The SOPA framework has two

basic components (plug-ins) namely “Services Bus” and “Pipeline”. In the subsequent sections these components will be explored in more details.

3.2.1 Pipelines

A “Pipeline” in SOPA terminology can be defined as a uniquely named set of service-calls and intermediate transformations. The pipelines are defined using an XML structure that specifies pipeline steps and relevant transformations. In SOPA paradigm a pipeline may be composed of other pipeline or services and as a result the pipelines are highly reusable. The pipeline concept provides a higher level of abstraction between services and applications that are benefiting SOA. SOPA provides some mechanisms for orchestration of services and transformation of results. It also supports many data processing features and flow management. In the SemanticLIFE architecture pipeline plug-in provides the SOPA features and plays a central role in the orchestration of basic system services and also in the creation of new business services.

The pipeline idea has been inspired from Apache Cocoon (Cocoon, 2009) which is a web development framework built around the concepts of separation of concerns and component-based web development. Cocoon implements these concepts around the notion of “component pipelines” and each component on the pipeline specializing on a particular operation. This makes it possible to use a Lego-like approach in building web solutions and hooking together components into pipelines without any programming required. Listing 3.1 shows a simple pipeline which is identified by its name at the first line.

```
1.    <pipeline name="square">
2.    <parameters>
3.            <parameter name="num" type="xsd:double"/>
4.    </parameters>
5.    <call service="org.example.arithmatics" operation="multiply">
6.            <parameter>{num}</parameter>
7.            <parameter>{num}</parameter>
8.    </call>
9.    <transform method="xml" stylesheet="result.xsl"/>
10.   </pipeline>
```

Listing 3.1: A simple pipeline

Each pipeline may receive some input parameters that might be used anywhere inside the pipeline's scope. Lines two to four (see Listing 3.1) show the parameter section and definition of a parameter which is called `num`. The most interesting part of a pipeline which distinguishes our approach from other such solutions is the service-call part. At line 5 (see Listing 3.1) the "multiply" operation of the service `org.example.arithmatics` is requested. The operation call can consume parameters of pipeline.

The results returned by the services may be transformed during the execution of a pipeline. This feature let the results be transformed and converted to required format. The transformation is performed by applying an XSLT transformation to the current pipeline results. The pipeline plug-in keeps the results internally and finally at serialization phase the results are rendered in specified format. The supported serialization formats are TEXT, XML, HTML, and XSWT (XSWT, 2009).

The created pipelines will be used by other system components and may provide a range of services covering the business logic, visualization features, or a combination of these two. A pipeline can make multiple calls to other services or pipelines and finally returns the final result to the calling application. The pipelines on the higher levels can be documented and reused as new business services. Visual rendering and styling of the results is also an edge of pipelines that combines the results of business processes with different visualization options. As a result a specific set of results can be rendered differently based on the context and user requirements.

The pipeline concept also supports a higher level of services that may be compared with BPEL for Web Services that provides some mechanisms for orchestration of services and transformation of results. The advantage of SOPA in the proposed framework is that is provide a much simpler information integration approach by encapsulating small pieces of functionalities in pipelines. In contrast BPEL language currently does not support the explicit definition of business process "fragments" that can be invoked from another (or the same) business process. The only way to approximate similar behavior today is by defining a complete business process as an independent service and invoking it as a separate activity. The fact that the invoked activity is really implemented as another process is completely hidden from the

parent process, in other words, there is no chance to establish any coupling of process instance lifecycles (BPEL Sub-processes, 2009).

A pipeline may contain multiple calls to SOPA services. The call results can be then coupled together using the internal data management and data manipulation of SOPA. For example the example presented in Listing 3.2 shows how the `add` and `multiply` operations of `org.example.arithmatics` service are called one after another.

```
1. <pipeline name="combined" serialization="xml">
2.  <parameters>
3.      <parameter name="first" type="xsd:double"/>
4.      <parameter name="second" type="xsd:double"/>
5.  </parameters>
6.  <call id="firstCall" service="org.example.arithmatics" operation="add">
7.       <parameter>{first}</parameter>
8.       <parameter>{second}</parameter>
9.  </call>
10. <call id="secondCall" service="org.example.arithmatics" operation="multiply">
11.             <parameter>{first}</parameter>
12.             <parameter>{second}</parameter>
13.  </call>
14. </pipeline>
```

Listing 3.2: Multiple calls to SOPA services

After processing the pipeline with input parameters 5 and 6 the result would be returned as shown in Listing 3.3.

```
1.    <result>
2.       <firstCall>
3.          11
4.       </firstCall>
5.       <secondCall>
6.          30
7.       </secondCall>
8.    </result>
```

Listing 3.3: Pipeline's multiple call execution results

Please note that the name of result nodes are taken from ID attribute of call in the corresponding pipeline.

It is also possible to make nested calls to SOPA services; i.e. the services may be chained together to exchange the parameters and results. The example presented in Listing 3.4 depicts such situation.

```
1.      <pipeline name="nested" serialization="xml">
2.      <parameters>
3.                <parameter name="input" type="xsd:double"/>
4.      </parameters>
5.      <call id="parent" service="org.example.arithmatics" operation="multiply">
6.           <parameter type="xsd:double">2</parameter>
7.           <parameter>
8.               <call service="org.example.arithmatics" operation="add">
9.                    <parameter type="xsd:double">15</parameter>
10.                   <parameter>{input}</parameter>
11.              </call>
12.          </parameter>
13.     </call>
14.     </pipeline>
```

Listing 3.4: Nested call to SOPA services

As shown in Listing 3.4 the first service call (see line 5) takes two parameters, the first parameter is 2 and the second one is the output of a call to another service. The result of calling this pipeline with an input parameter value of 5 is shown in Listing 3.5.

```
1.      <result>
2.         <parent>40</parent>
3.      </result>
```

Listing 3.5: Pipeline's nested call execution results

The pipeline plug-in provides a set of tags that allow us to call services based on some XPATH-like conditions. For this purpose two conditional structures are provided which are "if" and "choose". In both conditional structure a test condition which can be an XPath token (this may also include variable replacement operator) will be checked and based on check results the appropriate service will be called. The pipeline given in Listing 3.6 shows an "if" structure (see line 5) with test condition `{input} > 0`.

```
1.      <pipeline name="nested" serialization="xml">
2.      <parameters>
3.                <parameter name="input" type="xsd:double"/>
4.      </parameters>
5.      <xsl:if test="{input} > 0">
6.          <call id="parent" service="org.example.arithmatics" operation="multiply">
7.                <parameter>2</parameter>
8.                <parameter>{input}</parameter>
9.          </call>
10.     </xsl:if>
11.     </pipeline>
```

Listing 3.6: Pipeline's conditional structure for "if"

An example of a pipeline using “choose” structure is given in Listing 3.7.

```
1.  <pipeline name="nested" serialization="xml">
2.     <parameters>
3.          <parameter name="input" type="xsd:double"/>
4.     </parameters>
5.     <xsl:choose>
6.       <xsl:when test="{input} > 0">
7.         <call id="parent" service="org.example.arithmatics" operation="multiply">
8.            <parameter>2</parameter>
9.            <parameter>{input}</parameter>
10.        </call>
11.      </xsl:when>
12.      <xsl:otherwise>
13.        <call id="parent" service="org.example.arithmatics" operation="multiply">
14.           <parameter>2</parameter>
15.           <parameter>10</parameter>
16.        </call>
17.      </xsl:otherwise>
18.    </xsl:choose>
19.  </pipeline>
```

Listing 3.7: Pipeline’s conditional structure for “choose”

Up to now three kinds of parameters for service calls have been introduced:

- Literals: in this case the relevant parameter value is passed to services directly
- Pipeline parameter: in this case the parameter value will be replaced with the value of pipeline parameter at run-time.
- Service call parameter: in this case the result of a service call is fed as input parameter of another service call

Now a new type of parameter will be introduced that will have interesting applications. This parameter type will extract its value by applying an XPath statement to current XML results. The following example shows such a situation that the parameter value is extracted by XPath statement:

```
1.<call id="parent" service="org.example.arithmatics" operation="multiply">
2.    <parameter>{xpath:/result/previous}</parameter>
3.    <parameter>{input}</parameter>
4.</call>
```

Listing 3.8: Pipeline call with XPath-based parameters

An important benefit of service composition as pipelines is to create new complex services based on the existing services. The created pipelines can be called again and reused by other pipelines or SOPA components via their names. The provided

example of Listing 3.9 demonstrated the reuse of a pipeline in another pipeline. At line 4, a call has been made to `square` pipeline and then the results are forwarded to second parameter of `multiply` service.

```
1.<call id="parent" service="org.example.arithmatics" operation="multiply">
2.   <parameter>5</parameter>
3.    <parameter>
4.        <call id="parent" service="org.sopa.pipeline" operation="square">
5.           <parameter>6</parameter>
6.        </call>
7.     </parameter>
8.</call>
```

Listing 3.9: Pipelines calling another pipeline

The SOPA framework is equipped with a Web Service component that can capture a WSDL file and provide its operations to SOPA environment as pipelines. As a result the end-user just needs to drag and drop the WSDL file into the system. SOPA Web Service component will then generate the corresponding pipelines for Web Service operations and finally the can be called like any other SOPA service. The pipeline shown in Listing 3.10 demonstrates a sample of automatic generated pipelines for a given WSDL file.

```
1. <pipeline name="add" serialization="xml">
2.   <parameters>
3.     <parameter name="a" type="xsd:string"/>
4.     <parameter name="b" type="xsd:string"/>
5.   </parameters>
6.   <call service="org.sopa.webservice" operation="wsCall" return="xsd:string">
7.          <parameter name="wsdlLOC" type="xsd:string">
8.               http://127.0.0.1:8080/axis/HelloWorld.jws?wsdl
9.          </parameter>
10.    <parameter name="serviceName" type="xsd:string">HelloWorldService</parameter>
11.    <parameter name="portName" type="xsd:string">HelloWorld</parameter>
12.    <parameter name="operName" type="xsd:string">add</parameter>
13.    <parameter>{a}</parameter>
14.    <parameter>{b}</parameter>
15.  </call>
16. </pipeline>
```

Listing 3.10: automatic generated pipeline from WSDL file

For many collaborative use cases of SemanticLIFE we need to make calls to other desktop systems or expose parts of user information via a service call. For that reason, pipelines allow to submit calls to distributed services which are located on peer SemanticLIFE systems. For this purpose it is enough to include the IP address of the destination system in the service call as shown in Listing 3.11.

```
1.   <call service="at.slife.search@192.168.1.100" operation="mailSearch">
2.       <parameter>anjomshoaa@ifs.tuwien.ac.at</parameter>
3.   </call>
```

Listing 3.11: Calling the remote pipeline

To illustrate and evaluate the proposed approach a business process which has been realized using SOPA framework will be presented. In this scenario, the weather condition of some cities should be extracted using multiple calls to corresponding services. For this purpose, first the `listCities` service will be called which is an internal service, then the weather-conditions for each city will be queried from an external web service and finally the results will be ranked and rendered using an XSL style sheet.

```
1.  <pipeline name="checkWeather" serialization="xswt">
2.    <parameters>
3.      <parameter name="startDate" rdf:datatype="xsd:date"/>
4.      <parameter name="endDate" rdf:datatype="xsd:date"/>
5.    </parameters>

6.    <call id="cities" service="org.sopa.webservice" operation="listCities"/>
7.
8.    <xsl:for-each select="/result/cities/city">
9.     <call id="city-weather" service="org.sopa.webservice" operation="getWeather">
10.      <pipe:attribute name="city">{xpath:cityName}</pipe:attribute >
11.      <parameter>{xpath:cityNmae}</parameter>
12.      <parameter>{startDate}</parameter>
13.      <parameter>{endDate}</parameter>
14.    </call>
15.   </xsl:for-each>

16.    <call id="my-destinations" service="at.slife.profile" operation="rankData">
17.      <parameter>{xpath:/result/city-weather}</parameter>
18.    </call>
19.    <transform stylesheet ="weather.xsl"/>
20.  </pipeline>
```

Listing 3.12: A complete pipeline based use case

3.2.2 Service Bus

The heart of Eclipse framework is its plug-in and extension point mechanism. A particular plug-in can expose extension points where other plug-ins can be connected (Eclipse Plug-ins, 2009). A set of basic extension points and corresponding registries are provided along with the Eclipse platform for managing, mostly GUI intensive, extensions. A common example is `ViewsRegistry` for Views. Although different third party plug-ins exists for developing web services and deploying to already configured servers, but unfortunately Eclipse platform doesn't support plug-n-play

mechanism for web services as it does for Views. Such a mechanism can ease the web services development process by shifting the burden of service deployment from developers to the services management component. Consequently, service developers can focus on the functionality of the service instead of taking care of deployment details.

One part of the SOPA solution is Services Bus offering the extension point for service developers to publish their standard Java classes as web services. The standard extension point mechanism of Eclipse facilitate visual configuration of extensions with the extension provider. During the application start-up, the Service Bus loads all the connected services and automatically deploys them using embedded instance of Jetty servlet container and Apache Axis for creating and deployment of web services. The deployment scripts are created on the fly from the service description. Thus developers can, at the same time, benefit from Rich Client environment of Eclipse and Java web services using the this uniform and coherent mechanism.

The rationale behind the development of Services Bus is to achieve the vision of plug-n-play web services using plug-in and extension mechanism of Eclipse platform. First of all an extension-point was configured by following the service specification and deployment standards such as WSDL and WSDD. An abridged version of the extension point schema is depicted in Listing 3.13. The non-abridged version of the schema also includes its alignment with the above mention standards.

```
<schema targetNamespace="org.sopa.sbus">
   <element name="service">
      <complexType>
         <sequence>
            <element ref="operation" minOccurs="1" maxOccurs="unbounded"/>
         </sequence>
         <attribute name="name" type="string" use="required">
         </attribute>
         <attribute name="class" type="string" use="required">
            <annotation>
               <appInfo>
                  <meta.attribute kind="java"/>
               </appInfo>
            </annotation>
         </attribute>
      </complexType>
   </element>

   <element name="operation">
      <complexType>
         <sequence>
```

```
            <element ref="parameter" minOccurs="1" maxOccurs="unbounded"/>
         </sequence>
         <attribute name="name" type="string" use="required">
         </attribute>
         <attribute name="returnType" use="required">
         </attribute>
      </complexType>
   </element>

   <element name="parameter">
      <complexType>
         <attribute name="type" use="required"/>
         <attribute name="name" type="string"/>
      </complexType>
   </element>
</schema>
```

Listing 3.13: Abridged version of the business services extension-point schema.

Importantly the Services Bus exposes this extension point whereas the web services developers consume it to publish standard Java classes as services (see Figure 3.4 and code listing Listing 3.14). Thus the web services could be developed and maintained analogous to other Eclipse plug-ins. The Services Bus on the other hand reads configuration details of all the connected services during application start-up. It then automatically creates the WSDD based deployment script and uses embedded Jetty and Apache Axis to complete the task.

```
<extension point="org.sopa.sbus.services">

<service name="org.example.arithmatics" class="org.example.Arithmatics">
  <operation name="multiply" returnType="xsd:double">
    <parameter name="first"  type="xsd:double"/>
    <parameter name="second" type="xsd:double"/>
  </operation>
</service>

</extension>
```

Listing 3.14: Abridged version of a service description as an extension

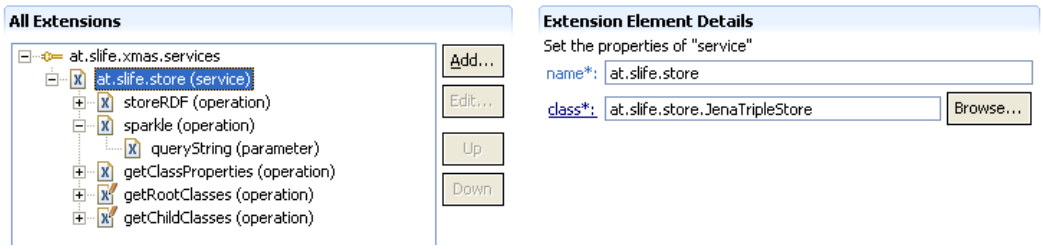

Figure 3.4: An example of service extension from the SemanticLIFE project

The Services Bus can be seen as the door to the Eclipse based SOPA systems. It is responsible for routing the service call requests to the actual connected service. An important functionality of Services Bus is to deliver a level of abstraction between system services which greatly improves the flexibility of SOPA systems. Thus it provides a uniform access layer and transparency to internal and external services.

Service Call: The services plugged into the Services Bus could be called using either the utility classes provided by the Services Bus (see the code listing x) or by using Apache AXIS. The former shares the same naming conventions for class and method names of the later, and both type of calls return exactly the same results. Additionally for local services specifying only their name is sufficient but calling external web services requires providing complete end-point URI.

```
Object[] params = ...
Call client = new Call("at.slife.store");
Object result = client.invoke("sparkle", params);
```

Listing 3.15: Calling a service plugged into the Services Bus.

As explained in the previous section, the available services in the SOPA environment are routed via the Service Bus plug-in; i.e. all services will be requested from Services Bus which is responsible for finding and then invoking the corresponding service to do the task. This feature provides a service transparency in the whole SOPA environment. As stated earlier the services in SOPA are not limited to plug-in-exposed services but optionally may include pipelines and external Web Services too. As a result the SOPA system brings the service orchestration scenarios to a new horizon. The business scenarios developed under Eclipse programming framework can combine resources coming from internal or external components via a single service routing plug-in (Services Bus plug-in). Figure 3.5 depicts the service transparency and the fact that SOPA framework presents a holistic view of all available services including pipelines, plug-in services, and also external web services.

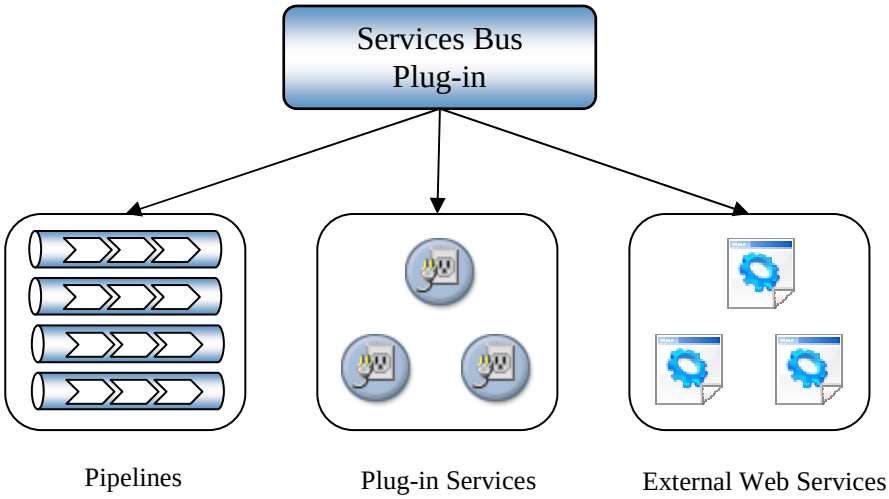

Figure 3.5: Service transparency in SOPA

Chapter 4

Web Form Integration

Nearly all human-computer interactions are happening through Input forms which are responsible for receiving the user input and sending it to appropriate component for further processes. Particularly, a large portion of Internet advances owes the human-computer interaction and data exchange via web forms and we are using them extensively in our daily activities. Current complex Internet applications demand a significant amount of time for development and maintenance of web forms which are solely designed for human users.

Over two decades history of World Wide Web, the Internet has evolved from a unidirectional information stream to an enterprise application framework; however, the web forms and their structure have remained unchanged. The traditional web forms are simple interfaces that aim to transfer data between server side components and browser application. An interesting attribute of web forms is that they contain atomic elements that can be more efficiently interpreted and processed, compared to the text content of web pages.

Despite the advances in design and implementation of web applications, the human interaction with the applications is still a frustrating experience for most of users. Every day, we are dealing with some processes that are scattered among different web based systems and the only way to run such processes successfully is to manage the data flow in the various web forms manually. In most of the cases we need to use

our previous experiences and knowledge to glue up the different steps of a process in our mind. In other words, human should play the role of a smart middleware who understands the logic behind the web forms, keeps track of the steps of main process, and interchanges the data between diverse sub-processes. In this context, the web forms are nothing more than an instant data exchange method with end user. As an example consider the typical use case of trip planning that requires interaction with several reservation systems, calendar items, and payment systems. In order to complete this process, the complex flow of events should be followed in a self-administered procedure.

Another issue that makes the web form interaction even more difficult is the usability issues. The fact is that most of web forms are being designed by programmers whose center of attention is undertaking the technical aspects of web applications and the usability and accessibility issues are not their main concern.

In this chapter, two novel approaches for overcoming the web form integration namely Semantic XForms and Web Form Services are presented and some of the corresponding use cases are discussed with more details.

4.1 Semantic XForms

The W3C solution for modernizing the web forms is called XForms with the mission to address the patterns of intricacy, dynamism, multi-modality, and device independence that have become prevalent in Web Forms Applications around the world (Forms, 2009). XForms offers separation of the form's purpose from its presentation and allows processing of data to occur using a declarative model composed of form elements for data calculations and constraints. It is also equipped with a view layer composed of intent-based user interface controls that are bound to the model. Finally XForms provides an imperative controller for orchestrating data manipulations, interactions between the model and view layers, and data submissions. Thus, XForms accommodates the reuse of form component, fosters strong data type validation, eliminates unnecessary round-trips to the server, offers device independence and reduces the need for scripting (XForms, 2009). Additionally, XForms offers a browser-neutral approach that concentrates on the data model first and then as a second step renders the data model for different

applications. The primary application of this concept is for interacting with end-users who are using various tools to access the web forms. In the XForms approach, forms are comprised of a section that describes what the form does, called the XForms Model, and another section that describes how the form is to be presented (XForms, 2009).

An XForms model is an XML structure that is included in the header part of the XHTML document and its elements are conforming to the documents' name spaces. Listing 4.1 shows an XForms example that includes a model for typical registration information. This model is then bounded to presentation components such as select and input fields. Furthermore some restriction can be added to each form elements and finally the submission element specifies target of the submitted data.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ev="http://www.w3.org/2001/xml-events"
      xmlns:conf=http://sample-conference.com/registration
       xml:lang="en">
<head>
 <title>XForms Sample</title>

 <model xmlns="http://www.w3.org/2002/xforms">
    <instance>
      <conf:registration as="register">
        <conf:reg-type />
        <conf:first-name />
        <conf:last-name />
      </conf:registration>
    </instance>
    <submission action=http://sample-conference.com/register
                method="post" id="submit" />
    <bind nodeset="/conf:registration/conf:reg-type" required="true()"/>
    <bind nodeset="/conf:registration/conf:first-name" required="true()" />
    <bind nodeset="/conf:registration/conf:last-name" required="true()" />
 </model>
</head>
<body>
    ...
    <group xmlns="http://www.w3.org/2002/xforms">
        <select1 ref="/conf:reg-type">
          <label>Select Registration Type</label>
          <item>
              <label>Member</label>
              <value>member</value>
          </item>
          <item>
              <label>Student</label>
              <value>student</value>
              <message level="modeless" ev:event="xforms-select">
              Please send a copy of your Student Card per fax.</message>
          </item>
        </select1>

        <input ref="/conf:first-name">
          <label>First name</label>
        </input>
```

```
        <input ref="/conf:last-name">
          <label>Last name</label>
        </input>

        <submit submission="submit">
          <label>Register</label>
        </submit>
    </group>
    ...
  </body>
</html>
```

Listing 4.1: Anatomy of XForms

The `instance` element of XForms model essentially holds the skeleton of XML document that gets updated as the user fills out the form. It gives the author full control on the structure of the submitted XML data, including namespace information. When the form is submitted, the instance data is serialized as an XML document and sent to specified target in the `submission` element. In the given example an instance of submitted data will look like the Listing 4.2.

```
<conf:registration>
      <conf:reg-type>student</conf:reg-type>
      <conf:first-name>Amin</conf:first-name>
      <conf:last-name>Amin</conf:last-name>
</conf:registration>
```

Listing 4.2: XForms's submitted data

With the emergence of complex business processes, there is a growing need to embed the web forms into user's information context. In other words the generic web forms should be personalized according to user history and context information. The W3C' XForms standard is a candidate to realize this goal. In this section a novel approach will be presented that focuses on exploring the role that the XForms model can play to connect an XHTML form to other available services and models by using Semantic Web technologies.

For this purpose the Semantic XForms concept is introduced that uses the official definition of data model's namespace in order to attach the model elements to a known ontology. So every element is specified with its fully qualified URI and this URI can be selected to be identical with the name space of any given domain

ontology. For instance the first name item of the data model in Listing 4.1, document has the URI of `http://www.sample-conference.com/first-name` which in context of an ontology with the same name space, will be completed with all other required information and rules about first name concept. In other words the form elements are mapped to an ontology via the XForms embedded model. As soon as the form elements are bounded to the ontology space, the application would benefit from all advantages of Semantic Web technologies. For example the XForms data model can be coupled to semantic description of some business services to facilitate the information integration of business services. Alternatively the XForms data model can benefit the user ontology and feed the web forms with useful information from personal services which are supported by a Semantic Desktop solution such as SemanticLIFE. The semantic coupling with personal services would be especially important for people with memory and learning impairments where the "lifetime memories" of Semantic Desktops can help them to recall and track the interconnection of events and information items.

Figure 4.1, depicts how XForms data model is used to connect the form elements to business or personal services.

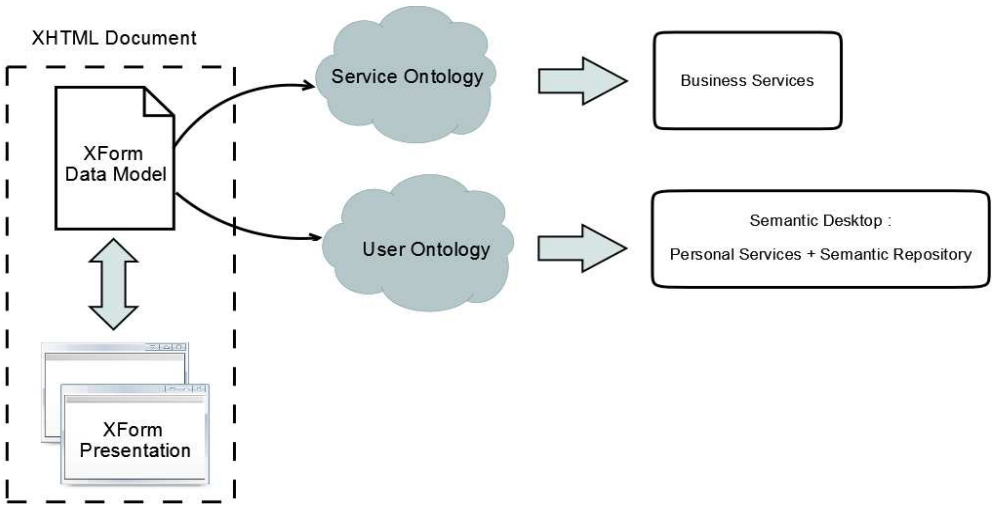

Figure 4.1: XForms's Service integration methodology

In the rest of this section the possible scenarios that have the potential to be addressed by Semantic XForms concept will be explored.

3.1.1 Web Application Design

Web forms contain atomic elements that can be better processed in comparison to web pages and natural text. Elements that appear on a typical form usually convey a logical relationship between elements. For example if a web form contains a city name and country name, there is a high likelihood that the city should be located in the specified country. Traditionally, such relation can be identified only by human users and the semantic of element relationships cannot be interpreted by computers. The logic behind a web forms is a determining factor for programmers to check the data consistency before passing the data to backend business processes. With the current advances of Semantic Web technologies, the form elements and their relationships can be described in a machine processable way which turns them to usable resources in business processes. In other words the form elements are mapped to ontologies via the XForms embedded model and this will enable the system to utilize semantic processes on form elements.

An interesting use case of this style is to apply the validation services to form elements via appropriate web services that are also annotated and described by the same domain ontology. The domain ontology which plays an important role in the presented solution, should include the description of all form elements and more importunately the relationships between them. In a typical web application, the basic ontology can be extracted from the database schema and then completed by system designer.

After mapping the XForms's data model to domain ontology by means of Semantic XForms, the domain ontology may support the programmer in the following ways:

- First of all the domain ontology can specify the checks that can be done based on those elements that appear on the web page and their relationships. For instance if the page contains a country name and a city name as parts of user's address, then the city should be located in the specified country. This result is inferred from the ontology predicate in the domain ontology that connects a city to country with a `locatedIn` relationship.
- The domain ontology can also be used to find the appropriate validation services from semantically-enabled service repository. In this approach the

required logical tests that should be applied to form elements will be deduced from the domain ontology. As the next steps the service repository will be searched for the services that can fulfill the logical tests and the chain of such services will outline the form validation process that might force the user to correct the data before submission.

To clarify the proposed method, consider a typical conference registration form that is shown in Figure 4.2.

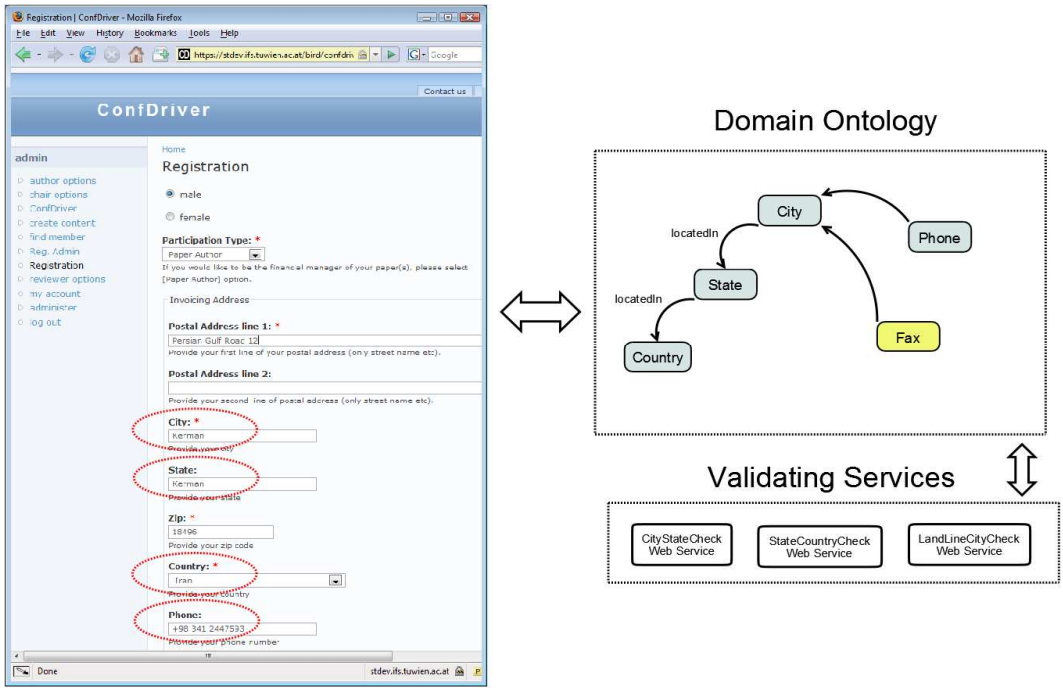

Figure 4.2: Domain ontology connecting the form elements to validating services

The relationship and logic behind the elements of registration form can be deduced from domain ontology.

Figure 4.3 shows the domain ontology that has been used for the registration use case. According to domain ontology city name should be located in the specified state and country or the telephone and fax number should match the country and city codes.

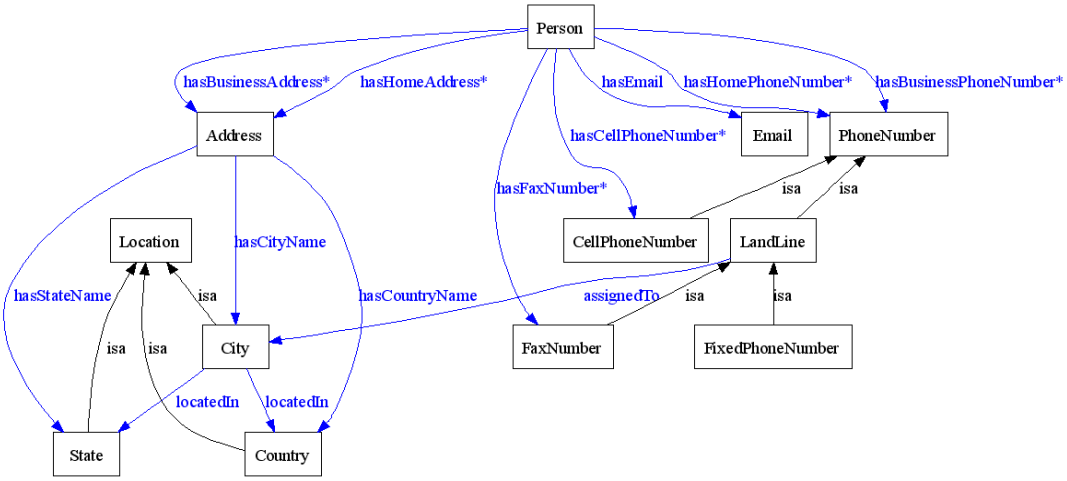

Figure 4.3: Domain ontology and element relationships

These logical relationships can be detected from the given domain ontology and consequently following set of logical tests is produced:

- city should be located in specified state
- state should be located in specified country
- the syntax of telephone number should match the city and country codes

It is supposed that the corresponding services to perform these checks are either available as web service or as a well defined API. Moreover the validation services should be also described using the domain ontology. So to generate the validation component, system will consult the repository of validating services and find the relevant functions. Then these functions will be composed into one validating procedure that calls the validation web services one after another to validate the submitted web form data. The resulted validation procedure can be then either:

- added to web form as an Ajax web service call that makes the checks during the user-interaction with the web form. Or
- implemented as a server-side validation function that is invoked as soon as the submitted data arrives.

It is important to note that the resulted validation procedure can be automatically regenerated according to the latest configuration of form elements. In the context of the conference registration use case, adding a new form element for Fax number, should add the corresponding check for fax number to the validation procedure.

According to domain ontology the fax number is assumed to be a Land Line (see Figure 4.3) and as a result the telephone checking web service can be also used for checking the validity of fax numbers.

This scenario is also true when a field is removed from the web form. So for example by removing the city name from registration form the validation function should be changed and the telephone and fax numbers will be only checked to be valid in combination with selected country. Additionally the city-state checking will not be necessary anymore and corresponding check will be removed from validation function.

3.1.2 Personal Service Integration

Another interesting aspect of Semantic XForms that will be explored in this section is the role that it can play in the integration of desktop information (user's world) with other business processes. More precisely, the Semantic Web should bridge the gap between user information and external processes by mapping the user resources to those needed by a specific web form. As an example consider an online shopping system that requires the payment information from the user. Such data should be provided by each shopping (or once per shopping system); however, this could be avoided by integrating the user information which reside on the user's Semantic Desktop.

Semantic XForms may facilitate the integration of user profile information into web forms via the shared common understanding of form elements and user profile items. In this context the form elements that are mapped to domain ontology via XForms model, will be queried and their corresponding value will be extracted from user profile. For instance a required `first-name` element in a web form is automatically connected to the user’s first name in his/her profile and user does not need to fill it out many times in different forms.

The required form elements are not always a primitive value as indicated in previous example, but in some cases the extraction of element values involves using appropriate personal services or applying the semantic inferences. An example of such use cases is feeding the hotel reservation form with the preferred location and correct start and end date that match a specific conference event. In this case the

user's calendar and its associated personal services will be contacted for finding the suitable hotel and carrying out the reservation process.

3.1.3 Web Form Accessibility

The current complex Internet applications are not easy to use for people with cognitive impairments and they usually use the Internet as a publication platform. The human interaction issues gets even more complex when the web forms should be used by people with special needs who have their specific limitation and requirements. The Internet as the widest medium for business and communication can be effectively adapted to the requirements of differently-abled people.

To change this state, the web forms, as the basic block of Internet applications, should be made accessible for people with varying levels of cognitive abilities, especially for severely challenged users in this category. New technologies such as the Semantic Web and XForms can be combined to make the Web applications more accessible for these people. In this section, the possible application of XForms in combination with the Semantic Web technology is explored and a solution model for providing accessibility for people with cognitive and visual impairments is presented.

Elements that appear on a typical web form usually convey a kind of logical relationship between the elements. These relationships can be easily understood and modeled by a normal user during the interaction with a specific web form; however, the users with different type of impairments (e.g. Visual, memory and cognitive impairments) may not easily conceive those relations. There might be many causes that disable such user to capture this model. For example, some users with vision impairments may use software tools that magnify a small area of the screen and show it on the whole screen. As a result, the screen area visible to the user will be limited and the logical connections of a specific field under focus with surrounding fields cannot be identified by the user. As another example, consider the case of a user with cognitive impairments that cannot clearly connect the items together or forgets the previous items when focusing on part of a Web form. The capture of form logic is especially important for people with cognitive impairments who are not able to detect the logical relationship between elements or keep in mind longer chain of relationships, and consequently cannot use web forms efficiently.

The user with normal cognitive abilities is able to combine the application's logic and goals with his / her experiences and knowledge to accomplish the use of Internet application; however, the users with severe cognitive limitations either face difficulties in identifying the data model behind the web form, or cannot map the model to their knowledge models. Also relating the page contents with lifelong memory of end users which might be of the great advantage for memory impaired people, is not possible in many cases.

Semantic XForms is an attempt to bridge the gap between user's world (Semantic Desktop and Personal Services) and global world (global web services, business processes, etc.). In this section some possible solutions based on XForms and Semantic Web are presented and the feasibility of implementation of corresponding use cases is discussed in details.

Similar to the use cases of the previous sections, the Semantic XForms concepts can be again helpful to address the accessibility use cases by applying the appropriate semantic mappings between form elements and the domain ontology. So for example in a typical web form that requires both city name and country name, the given city name should match the selected country. This relationship which has already been captured in data model ontology can be combined with the user ontology to find out the appropriate rendering option according to user's cognitive abilities. A possible rendering solution can be providing a color guide map and coloring the relevant fields with same color. Another solution is to provide additional help text to the relevant fields that implies the relation of a field with other fields on the web form. Figure 4.4 shows how XForms data model can be combined with data model and user ontology that will empower a customized rendering of form elements.

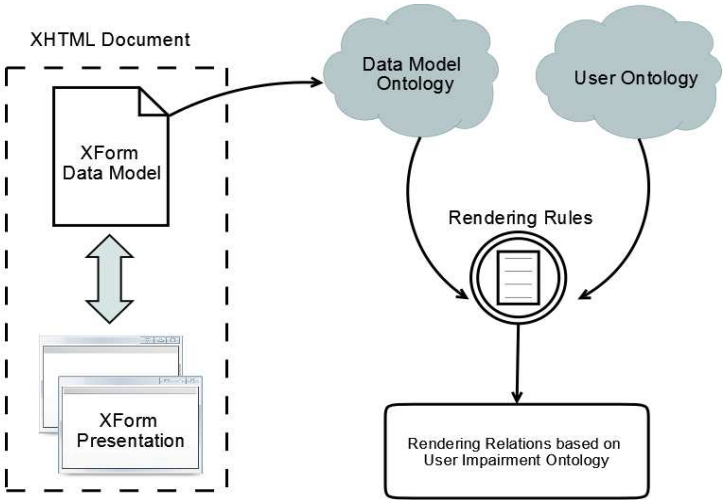

Figure 4.4: Xform's Model Rendering methodology

Beside the element dependency visualization that is discussed above, there is another type of visualization that is dealing with visual impairment issues. The latter type of visualization, adopt the web pages according to end user's capability and restrictions. For example to overcome the colorblindness of the end user, it is required that some specific color get rendered differently. The rendering process can also be automated using the user's profile and applying the inferred rules to web pages. An interesting use case of SemanticLIFE system was dedicated to realization of this scenario where the ontologies formally describe the mapping information about user's impairments, and the available interface characteristics (Shuaib, 2006)(Shuaib, 2007). Especially important for such use cases, is the semantic profile of the end-user (people with cognitive or visual impairments) that includes the user's physical and perceptual capabilities as well as the user preferences.

Dealing with cognitive problems, the proposed solution has among others to deal with the linguistic vocabulary of the user which may not be very broad. That might be because of learning deficit or memory problems. Also, it is challenging for such types of users to associate a concept with another, for example, natural association of cities with countries, and gender with title on the input forms. The XForms can adopt themselves with the software applications that use them and present the forms according to user's requirements. This well-known aspect of the XForms provides a great possibility to make web forms more accessible for users with special needs. The presented Semantic XForms solution can also provide customized help texts based on the user's cognitive level and user's history.

Another issue that should be considered for people with different levels of memory impairment is lifetime data consistency. In the long periods of time a memory impaired persons might forget the detail of their interactions with a service provider and submit some data that is in conflict with previously provided data. For example, user's name or address might be spelled differently (e.g. Special characters in European languages) and it can cause ambiguities and inconsistencies at the backend systems. An XForms data model that is mapped to a personal ontology can help to add a consistency check before submitting the data. So the user's history is consulted to avoid such problems.

Moreover, similar Internet applications usually require a common set of data that should be provided by the end-user. For example to register in an online shop, a user needs to provide the payment information, the delivery address, etc. The Semantic Desktop can help the user by auto completion of common fields. Again the XForms data model that is mapped to user ontology can be used to query the Semantic Store and extract the required information. The auto completion web forms using the user's profile or personal services will significantly ease the web form completion for people with memory, cognitive or motor impairments.

XForms as the next generation of web forms has attracted the attentions and many scenarios can be realized more efficiently using that. In this section, the feasibility of combining the XForms and ontologies was presented in order to leverage the web form semantic and realize advanced information integration scenarios. To our belief, the emergence of Semantic PIM systems like SemanticLIFE plus Semantic XForms mappings makes the integration of personal desktops into real world business processes, conceivable.

4.2 Web Form Services

As explained in the previous section the web forms play an important role in human-computer interactions; however, there are two basic obstacles that should be defeated first in order to utilize the full power of web forms:

- The semantic behind web forms is yet understood and used by human users based on their previous knowledge and experiences

- The business processes that use scattered web forms to exchange the required data are running manually and depend on human actors.

The Semantic XForms solution that was presented in previous section tackles the first obstacle and tries to exploit the semantic of web forms and embed its information into user context. In order to solve the second problem, the web forms and the services behind them should be turned into well-defined pieces of functionality. In this way the web forms can be employed by more complex processes and get executed automatically without human intervention.

To clarify the Web Form Service concept consider the simple scenario of getting the latest stock quotes of Google in Euro currency. Although this service is one of the most primitive services of stock exchange programs and the result can be acquired gracefully from one of service providers in this field, we assume that calling the relevant web forms is the preferred approach for a specific end-user. It is important to note that there are numerous web forms whose services are not as simple as the proposed scenario and no straightforward way for calling their functionality is available.

Apparently the proposed problem can be simply answered in two steps. As step one the Google Finance page can be queried for Google Inc. stock quotes. The result of this query is the latest stock quote of Google in U.S. Dollar. Figure 4.5 shows the Google Finance page and the query results after submitting the query for term Google.

Figure 4.5: Google Finance service for latest stock quotes

As the next step the result of first step should b converted to Euro. This task can be also done using a currency conversion service such as Yahoo Finance. Figure 4.6 shows the web form of Yahoo Finance that accepts the input currency, output

currency, and the amount to be converted in order to calculate the amount in target currency.

Figure 4.6: Yahoo Finance service for currency conversion

In the process of using web forms there are some common flow of standard steps that will be interesting for the proposed solution:

- Step 1: browsing to the URL that contains the required web form
- Step 2: entering the input values into the web form. In the proposed scenario, to fulfill this step for the Yahoo Finance page, the input currency should be set to U.S. Dollar, output currency to Euro, and input amount to the given amount. Finally the corresponding form should be submitted.
- Step 3: extracting the results from resulting page. For the proposed scenario the result of Google Finance query will be extracted from the resulting HTML page.

A human user can follow these steps one by one and transfer the required values to the other web forms or service; however, a more generic view of the proposed process shows that this process might also get automated. In this regard and to extend the footprint of web forms, we have implemented a novel component to translate the web forms to a plain formal Web Service that can be described semantically and

integrated in more complex solutions. This web services are referred to as Web Form Services in the rest of this section. Similar to normal web services, the Web Form Services also present their definitions to the end users via WSDL convention. Furthermore the end-users will not be aware of the form-based nature of these services and will merely see them as formal web services.

It is interesting to note that the Web Form Services do not necessarily needs an input parameter and may simply return some values when they are called. For example a simple weather web service will return the weather of a default location for each received call and requires no input parameter. The equivalent of this situation for Web Form Services happens when a web page is simply browsed and its content contains some useful information that should be extracted. An example of such web pages is the extraction of latest news from a university website. This kind of data extraction methods which is also known as web scraping or web harvesting is a specific case of Web Form Services; however the Web Form Services in addition provide a web service interface for extracted data which makes the reuse and integration of the resulting data more convenient.

This idea is especially helpful in automating the tasks between multiple web applications in a formal way. It is now very common in the big enterprises to use web applications for different systems and usually there is no unified system that can take over all the tasks. So the enterprises are now encountering a bunch of different web applications such as accounting, personnel management, etc, and each application is responsible for part of the business activities. In use cases requiring data records that are managed by different applications, end users should either wait for some complex backend integration from software architects, or manually transfer the data between applications to get the required results. In this context Web Form Services provide a unified way of describing the services and reusing them for different business activities.

Another interesting aspect of Web Form Services is that, they can be mixed with real web services in context of business processes. For instance the proposed scenario of this section is solved using two Web Form Services for Google Finance and Yahoo Finance. These services can be combined in a business process that calls them one after another to fulfill its goals. Figure 4.7 shows such a business process that

invokes these two Web Form Services and uses the results to calculate the volume of investments in Euro currency. The Investment calculator service in this business process, simply multiplies the stock quote of the given company by number of shares that is taken from another service that can be an instance of a personal service.

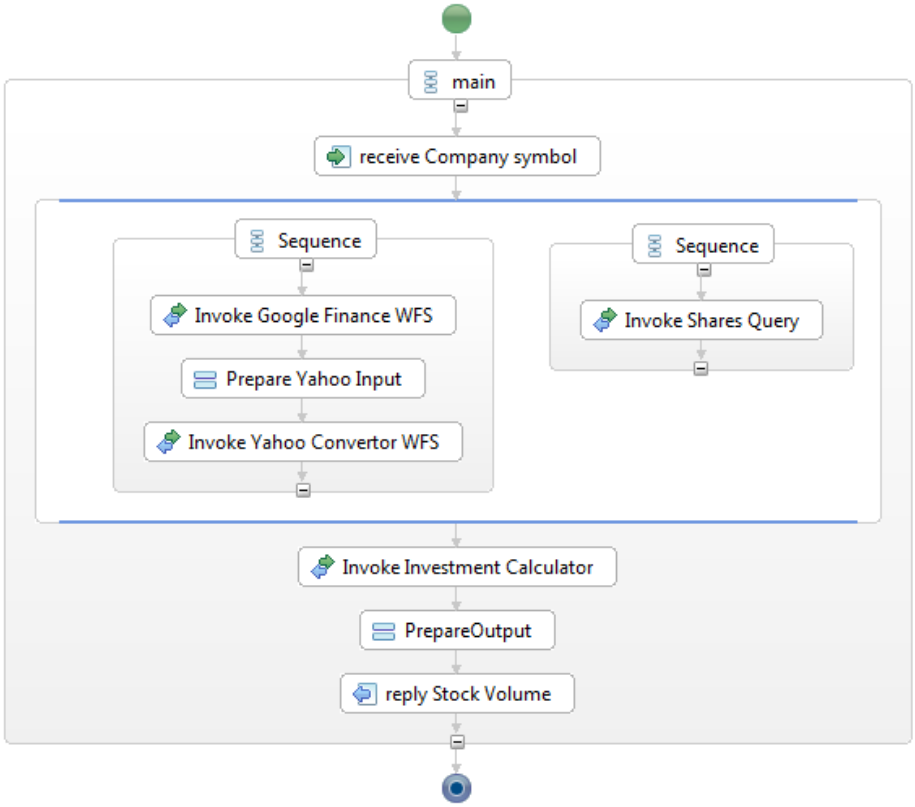

Figure 4.7: A business process that uses Web Form Services

4.2.1 Web Form Service Architecture

In this section an architectural overview of required components for realizing Web Form Services (WFS) is presented. These components support the process of translating web forms to formal web services and are implemented as server-side components. Figure 4.8 depicts different components of WFS server.

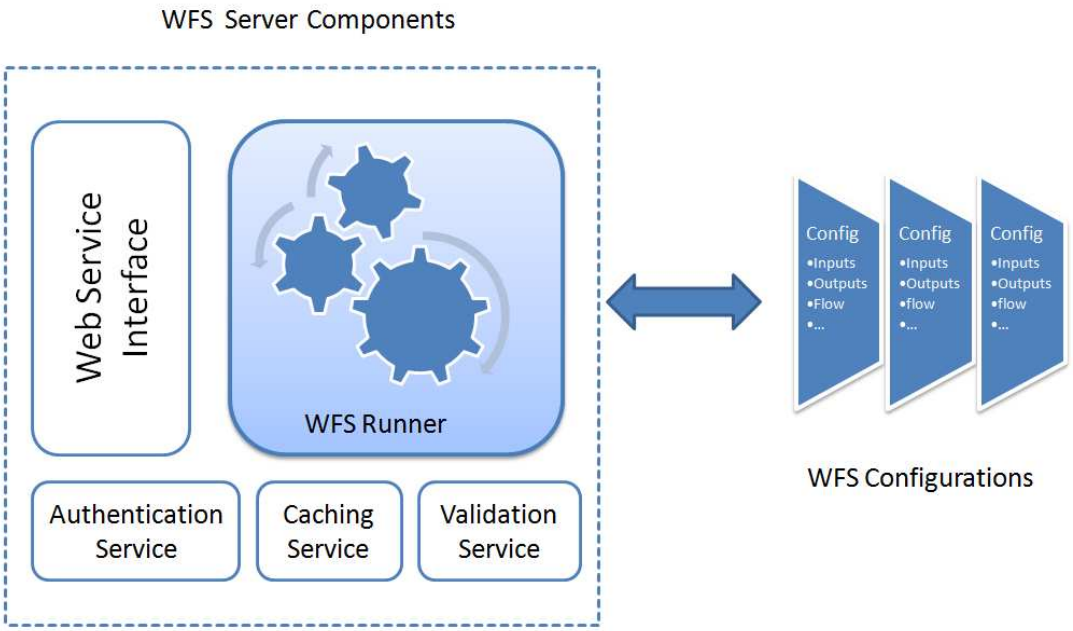

Figure 4.8: Web Form Service architecture overview

The core element of WFS architecture is its configuration files that define the necessary interactions between system and a specific web form to achieve the required results. The configuration file captures such interactions in a formal way and at runtime the system will use these documented interactions to simulate the actions of a human user.

The configuration files are also used to generate the definition of corresponding web services and presenting them as WSDL files that include input and output parameters. Listing 4.3 shows a typical Web Form Service configuration which defines the currency converter web service using Yahoo finance services. The configuration file defines one method for each data exchange process with the target web form. The methods are then exposed as formal web service operations in the corresponding WSDL file and can be called by any web service client program.

Each method is composed of a sequence of actions which describe the precise user interactions with target web form (see point 3 in Listing 4.3). For instance in Listing 4.3, after browsing to the start URL (defined in point 1), the system should perform some actions such as:

- `puttext` : for filling out the text box of amount

- `select` : for selecting the appropriate currency and setting “from currency” and “to currency” fields.
- `click` : for simulating the click on the “calculate” button.
- `getvalue` : to extract the selected value from resulting page.

For selection of the appropriate HTML element in order to perform required actions, the XPath query language has been used.

```
<?xml version="1.0" encoding="UTF-8"?>
<service name="CurrencyConverter"
         xmlns="http://www.tuwien.ac.at/serviceenabler/service">

<login startUrl=http://de.finance.yahoo.com/waehrungsrechner ❶
       validTime="604800">
  <actions />
</login>

<method name="convert" validTime="28800"> ❷
  <actions> ❸
    <puttext>
      <xpath>//input[@name="amt"]</xpath>
      <argname>amount</argname>
    </puttext>

    <select>
      <xpath>//select[@name="from"]</xpath>
      <argname>from</argname>
    </select>

    <select>
      <xpath>//select[@name="to"]</xpath>
      <argname>to</argname>
    </select>

    <click>
      <xpath>//input[@value="Umrechnen" and @type="submit"]</xpath>
    </click>

    <getvalue name="value"> ❹
      <xpath>//div[@id='converter']/table/tbody/tr/td[1]/table[3]/
             tbody/tr/td/table/tbody/tr[2]/td[5]
      </xpath>
    </getvalue>

    <getvalue name="date">
      <xpath>//div[@id='converter']/table/tbody/tr/td[1]/table[3]/
             tbody/tr/td/table/tbody/tr[2]/td[3]
      </xpath>
    </getvalue>
  </actions>
</method>
</service>
```

Listing 4.3: Web Form Service configuration for currency convertor

Another component of WFS server is the caching component that is responsible for keeping track of results and reusing them if possible. Although the WFS are behaving link formal web services from end-user's point of view, but they are a little bit slower. This is because of the fact that Web Form Services needs to negotiate with the corresponding web form and follow the steps which are specified in configuration file one by one. To improve the competence of WFS, the caching mechanism will try to reuse the previous results for answering new service calls. This feature is realized by a timeout attribute which is added to each defined method in configuration file. For instance in the currency convertor configuration file results of convert method can be reused within the specified timeout period which is set to eight hours and will get outdated after that (see Listing 4.3, point 2).

The web service view of WFS is supported by the "Web Service Interface" component that reads the configuration files as input and offers a web service operation for each defined method of web form. After receiving each call, the web service interface will load the corresponding configuration file and delegate the task to WFS runner component. The runner component uses the HtmlUnit (HtmlUnit, 2009) library which is a GUI-Less browser for Java programs. It models HTML documents and provides an API that allows programs to invoke pages, fill out forms, click links, etc.

A major drawback of approaches such as WFS that use web harvesting techniques to achieve their goals is that the web pages that host the required data might change their content or presentation style and this will invalidate the web harvesting procedure. A workaround that has been applied in WFS architecture is a verification mechanism that checks the document structure before web harvesting action. For this purpose, the configuration file is equipped with some defined tests that must be verified on the target page first to ensure that the web page structure has not been changed since the definition of method configurations. Listing 4.4 shows such verification test (see point 2) that extracts a node value from web page and checks it against a static value. In case the verification fails, the corresponding WFS will get invalidated and service provider is notified to fix the configuration file by considering the changes of target web page.

Another challenge of WFS approach is that many web pages with useful information and important web forms are protected by appropriate authorization and authentication mechanism. Thus to enable the WFS to access protected pages and extract required data, the WFS server should also manage the authentication issues. For this purpose the WFS configuration file includes a login section that submits the username and password to the target platform and receives a valid session for authenticated user. Using this session, WFS server will be able to access internal pages of web application and follow the commands of configuration file to feed the web service interface.

Similar to WFS caching mechanism for recent calls, the authenticated sessions are also cached to ease multiple calls to the internal pages of the same web application. In this regard, each authenticated session will have a time-to-live that is specified by `validTime` attribute of login section in configuration files (see point 1 in Listing 4.4). Since the authentication tokens are directly stored in configuration file, the proposed method is not that secure and more elaborated mechanisms should be used for this purpose. As the main concern of the WFS approach is the data integration, we have used this simple authentication method to demonstrate the feasibility of such solution. There are many best practice solutions that can replaced the proposed approach and provide a better security solution.

The WFS configuration file also defines the output value that should be returned by corresponding web service invocation. The return value of a web service can be an atomic value or a collection of data records. Listing 4.4 shows a real world example for the latter case, where the return value is the list of courses of a specific student (with defined authentication information) that is extracted from the TUWIS++ (the student information system of Vienna University of technology). The extracted information is a collection of courses that includes course title, course hours, and course link. The web service interface component of WFS server will translate the specified collection to equivalent XML data and will include it in the web service responses.

```
<?xml version="1.0" encoding="UTF-8"?>
<service name="TUWIS"
xmlns="http://www.tuwien.ac.at/serviceenabler/service">

<login startUrl="https://tuwis.tuwien.ac.at/" validTime="600"> ❶
```

```
  <actions>

    <puttext>
      <xpath>//input[@name="mn"]</xpath>
      <argname>matrnr</argname>
      <constantvalue>0123456</constantvalue>
    </puttext>

    <puttext>
      <xpath>//input[@name="pw"]</xpath>
      <argname>password</argname>
      <constantvalue>**********</constantvalue>
    </puttext>

    <click>
      <xpath>//form[@name="mnrform"]//input[@type="submit"]</xpath>
    </click>
  </actions>
</login>

<method name="tuwisCourseList">
  <actions>
    <click>
      <xpath>//a[. = "Abonnierte LVAs "]</xpath>
    </click>

    <verify> ❷
      <xpath>//table//th[5]/a</xpath>
      <expected>Titel</expected>
    </verify>

    <gettable name="lvas">
      <column>
        <name>Titel</name>
        <xpath>//table//td[5]</xpath>
      </column>
      <column>
        <name>Stunden</name>
        <xpath>//table//td[6]</xpath>
      </column>
      <column>
        <name>Link</name>
        <xpath>//table//td[5]/a/@href</xpath>
      </column>
    </gettable>
  </actions>
</method>
</service>
```

Listing 4.4: Web Form Service authentication

Chapter 5

SEMANTIC MASHUPS FOR ENTERPRISE

As explained in the previous chapters the Semantic Web and Mashups can provide a solid basis for many interesting applications and boost each other. The Mashup support for Semantic Web has come into view via the ad-hoc mashups that on one hand connects to the preconfigured information resources and processes the data of these resources and on the other hand maps its context data to the relevant domain ontology. In other words, instead of embedding the semantic meaning to the web content, the semantic is attached to relevant content via mashups in a dynamic and loosely coupled manner. Today, there are a handful of projects such as Google Sidewiki (Google Sidewiki, 2009), Dapper (Dapper, 2009), and Lixto (Lixto, 2009) that follow this approach to extract and enrich web page information; however in most of the cases the process remains at simple data extraction and text annotation level and semantic aspects are not covered completely. For example the Google Sidewiki which is a browser sidebar and has been released recently, lets users to contribute and read information alongside any web page. Google Sidewiki does not provide any semantic support at the moment and just aims to use the power of crowd to enrich metadata of web pages. The interesting part of this approach is the free annotation of page content which may overlap each other. In other words part of the page content can be annotated differently by different end-users based on their target applications and use cases. Also the quality of user annotations will be evaluated by crowd and better annotations appear on top of the list. Figure 5.1 shows an example

of Google Sidewiki that adds some description to parts of my home page that are related to the dissertation topics.

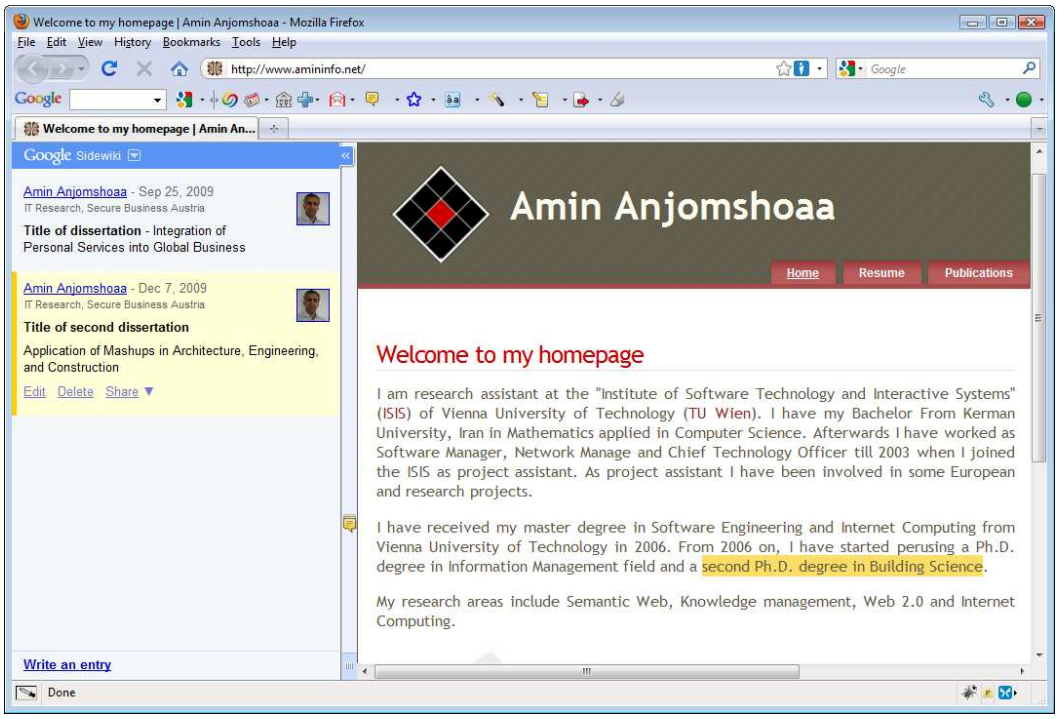

Figure 5.1: Google Sidewiki example

In near future, the Mashup support for Semantic Web will play an important role in realizing the Semantic Web goals and transforming the current web to an information space that can be interpreted and used by machines for advanced use cases.

The other facet of relation between Mashups and Semantic Web is the support that Semantic Web and ontologies can provide for Mashups to facilitate the creation of Mashups for novice users. This application of Semantic Web has its roots in Semantic Web Services that are aiming to automate service discovery and composition without human intervention. The basic difference between Semantic Web Services and Semantic Mashups approaches is derived from their different target users. The Semantic Web Services are mainly managed and used by IT experts who are aware of underlying data structures and corresponding services; however, the Semantic Mashups target group is novice users who need to combine the Mashup Widgets for their specific purposes (Hoyer, 2009). This issue is especially important for creation of Enterprise Mashups that always involve programming and include awareness of workflow. Enterprise mashups usually depend on some server-side

components and compete with data integration and service orchestration technologies such as BPEL and Enterprise Service Buses (ESBs). In Semantic Mashup context, these server-side components are described by appropriate domain ontology that eases the composition of mashup widgets and creating new mashups.

One of the major drawbacks of Mashup based solutions is the fact that such solutions are fragile and not as stable as formal business process. To provide a solid basis for more serious business application, the gap between mashups and well established BPEL solutions should be covered. In this chapter a novel approach for bridging this gap will be provided which enables the end user to benefit from simple service composition of Mashups and at the same time to manage and control the process by stable business process engines.

5.1 Semantic Mashup Use Cases

Mashups provide a paradigm to describe the user-defined service compositions and data resources as pipelines. It enables the end user to use existing data resources and make customized data integrations for user-specific scenarios. The result will be a new data resource that can be again reused by other Mashups as a data resource. By applying the Semantic Web concepts the mashups will have a machine-understandable description and this will enable the end-users to use the services and also the composed mashups easily and efficiently.

In this section the two interesting applications of Semantic Mashups and its added-value to different uses cased will be presented. These selected use cases that have a big potential for exploiting Semantic Mashup concept are:

- Semantic Mashups support for Architecture, Engineering, and Construction (AEC) fields
- Semantic Mashup Support for Personal Service integration

5.1.1 AEC Use Case

The Architecture, Engineering and Construction (AEC) industry is composed of multiple knowledge domains that are formed corresponding to the needed skills and professions. Sharing and exchanging knowledge is the key factor to success in such a

collaborative environment; however the distributed nature of AEC knowledge has lead to knowledge gaps between AEC related domains. Each domain has its own tools and applications and the data exchange between domain applications is not straightforward. In the other words the inter-domain communication is done only by the knowledge of expert building constructors (Shayeganfar, 2009). The vital need for more efficient data integration in Architecture, Engineering and Construction domains has forced the emergence of new data integration methods that can smartly share the building information among the stakeholders.

In this context Mashups can be considered as a business enabler in AEC domain that has the capability to cross the borders of different AEC knowledge domains and ease the creation of new situational solutions for various scenarios.

As an example consider the use case of energy simulation for the specific area of a building. For this purpose the end-user may create a situational solution based on some existing elementary AEC web services and widgets such as energy simulation web service, weather web service, and zone selector.

Figure 5.2 depicts this situational solution where zones and building location are extracted from building model widget and passed to appropriate web services. Without the Mashup solution, users should deal with complexity of parsing the building model, extracting zones and building location, and finally implement the corresponding web service client to communicate with required web services. All the above mentioned tasks can be transparent to user by creating a rich set of elementary services. Accordingly the user community may create and share new mashup services that can be reused in other mashups.

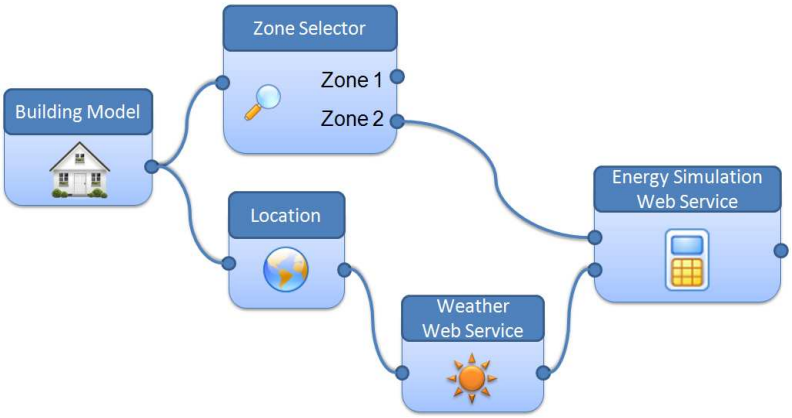

Figure 5.2: Mashup solution for energy smulation

Another issue that can assist the end-users to create mashups more efficiently and easily is the application of Semantic Web to mashup widgets. Similar to Semantic Web services, mashups can also benefit the Semantic Web and ontologies to describe the service profile, process model, and service grounding (OWL-S, 2009). This will enable the end-users to query and discover the appropriate services and properly compose them in their situational solution based on the service's inputs, outputs, and preconditions. In the context of AEC domain, the process model of a widget can be created based on Semantic Web service ontologies such as OWL-S and the proper domain ontology which in our case would be the Industry Foundation Classes (IFC) ontology. Figure 5.3 shows how the mashup widget input and output ports are mapped to IFC ontology which includes the proper description for all AEC related concepts such as temperature, humidity, etc.

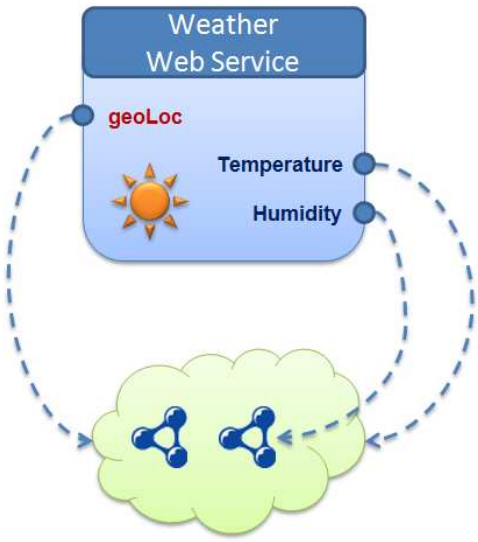

Figure 5.3: Mashup widget description using IFC ontology

Using the semantic definition of a widget that includes proper ontology mappings for input and output parameters, end-user would be notified about candidate input ports that can accept the results of selected output port. In the proposed scenario, the user notification is done by highlighting the candidate target ports. Figure 5.4 demonstrates the case when Temperature field of weather web service widget has

been selected. The mashup editor will then highlight the input ports of other widget that have a matching data type.

It is also important to note that, in some cases the comparison of data types would not be enough and a more sophisticated widget annotation is needed. As an example consider the SetPoint port of the cooling system in Figure 5.4 which defines the ideal temperature in the space when cooling is required. The meaning of setpoint is semantically different from the meaning of current temperature; however both are measured by a temperature unit. To avoid such situations, either the ports should be described more precisely or the end-user will select the correct port according to his/her knowledge.

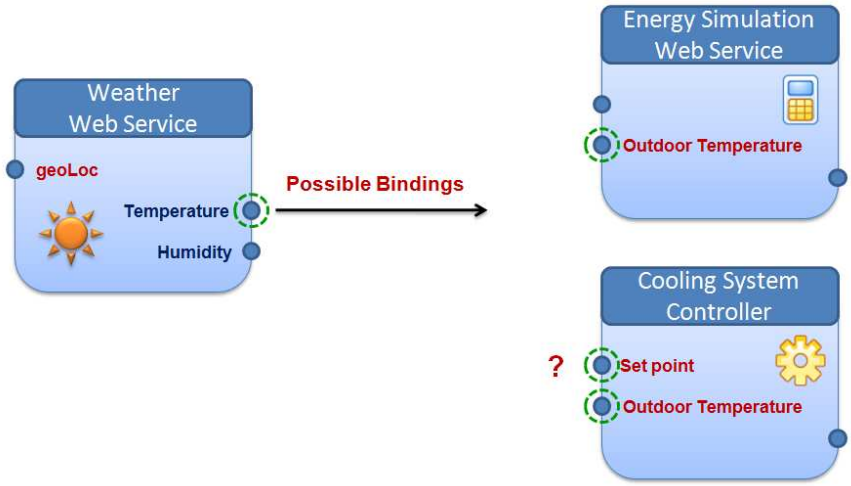

Figure 5.4: Highlighting of target ports based on semantic description of widgets

In order to find candidate bindings, a semantic query will be constructed based on the semantic description of selected port and will run against the semantic description of widgets in the current mashup. Result of this query is a list of ports that will be highlighted to assist the end-user in composition of required widgets. Listing 5.1 shows a SPARQL query that is used to select the proper target ports for Temperature port of weather web service.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ifc: <http://www.buildingsmart.at/IFC2X3.owl#>
PREFIX mash: <http://www.bit-future.com/Mashup.owl#>

SELECT ?widget ?input WHERE {

   ?widget mash:hasInput mash:input .
   ?input  rdf:type ifc:IfcThermodynamicTemperatureMeasure
```

```
}
```

Listing 5.1: SPARQL query for finding appropriate widget ports

5.1.2 Personal Service Use Case

To clarify the usefulness of the Semantic Mashup concept and the role it can play in empowering the Semantic Desktop services and personal data integration, another semantic mashup use case will be presented as follows:

> *A user wants to make a GeoRSS feed of the interesting locations nearby the conference he/she is attending and show them on a map.*

To make this happen, one may use his/her "Calendar Event" (Personal Service of Semantic Desktop) and feed the location to a "City Explorer" (external Web Service) service to get the customized GeoRSS feed that can be directly displayed on Google Maps (Google Maps, 2009). This mashup solution is shown in the upper part of Figure 5.5.

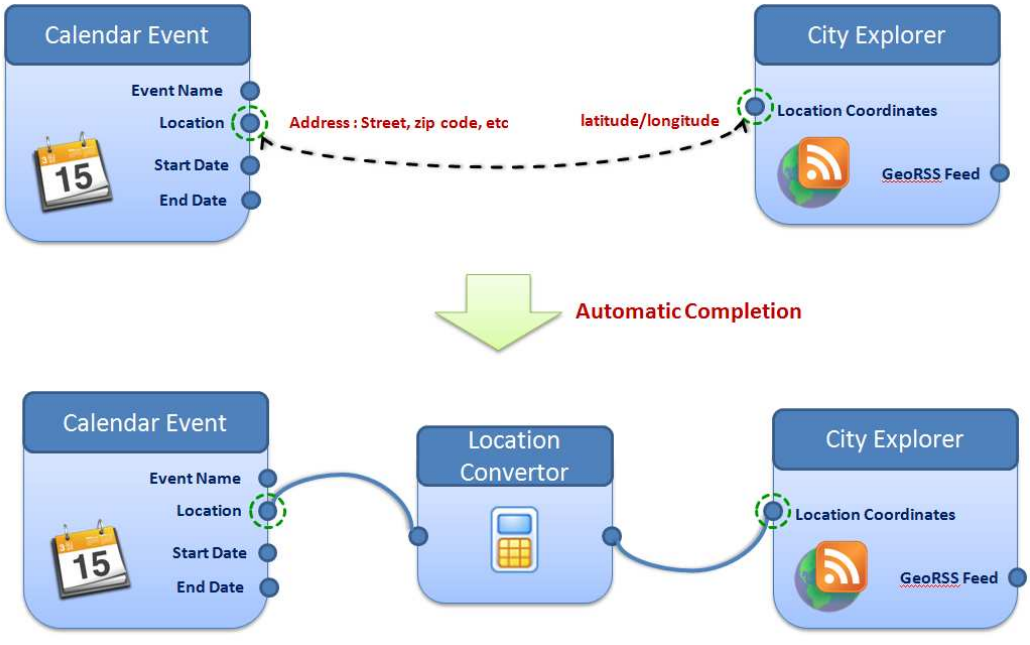

Figure 5.5: Automatic completion of mashups using semantic of available service

Although this process looks to be straightforward, the Location attributes of "Calendar Event" and "City Explorer" might be of different types. For instance suppose that the location of calendar item is formed as typical postal addresses and includes street name, postal code, city name, etc and the required location field of

city explorer service web service is a geographic coordinate. In a semantically enriched mashup environment such inconsistency of concepts can be easily inferred from widget semantics. Similar to AEC use case, a semantic query is constructed to select the appropriate target port. Assuming that the widgets are well-annotated, the result of this query on proposed mashup will be an empty set. As a result end-user will not be able to fulfil the task; however, an expert user can complete the mashups by employing a web service such as “Google Maps API Geocoding REST Service” (Geocoding, 2009) to convert the location address of event item into required geographic coordinates of city explorer web service.

The same job is doable by a novice user if the mashup editor could be aware of such geo-coding service and automatically inserts the geo-coding service between these two inconsistent location ports. To achieve this goal again the semantic of available widgets may assist the user to know that these location attributes are subtypes of the “Location” concept in the Semantic Desktop ontology and one of them (Address) can be converted to the other one via a publicly available service (geo-coding web service). In this context the Semantic Mashup will automatically add the convertor service as soon as these two location ports are connected (see lower part of Figure 5.5).

5.2 Mashup to BPEL Conversion

As explained before Mashups are very helpful in creating fast solutions for data integration, however a major drawback of Mashup based solutions is the fact that such solutions are fragile and not as stable as formal business process. To provide a solid basis for more serious business application, the gap between mashups and well established BPEL solutions should be covered.

A closer look into main mashup categories (presentation, data, and enterprise mashups) reveals that the functionality of mashups is to some extent similar to SOA and its service composition; however the overlap with SOA is much greyer for logic (enterprise) mashups than the other two categories. Client-side mashups compete with server-side orchestration technologies such as BPEL (Business Process Execution Language), but that hasn't stopped large enterprise SOA vendors IBM, Microsoft, BEA, Sun and Oracle from embracing enterprise mashups. The reason is

that, like BPEL, mashups depend on standardized Web services and APIs, all of which SOA is aimed at making available. This doesn't mean that mashups depend on SOA. In a field survey, nearly two thirds of enterprises that use browser-based mashups have done so without SOA, and one of the biggest advantages of mashup technology is the low barrier to entry, with many applications using free services on the Internet (Dornan, 2007).

Figure 5.6 (Watt, 2007) lays out the mashup framework and its relation with SOA to create situational application. In this context mashup provides a face or visual representation for services which consequently facilitates access to the properties and features of the service as well as the ability to reflect relationships (wiring) between services (Watt, 2007).

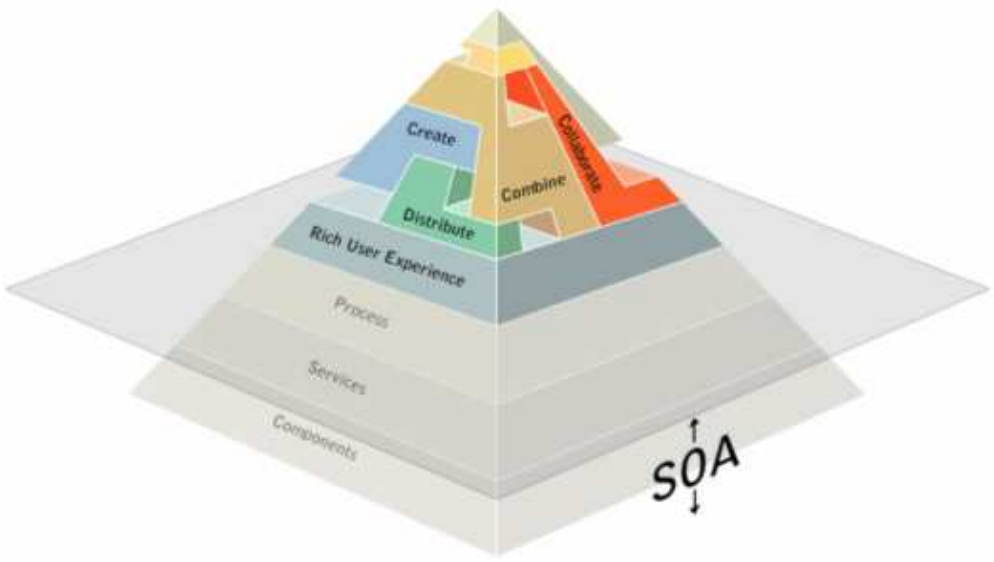

Figure 5.6: Mashup and SOA relationship

As the number of serious enterprise mashups grows and they get used for more serious tasks, there is a growing need to make the mashups more stable and robust for complex business scenarios. As a matter of fact, in context of SOA and Mashup solutions, stability and ease are at odd with each other (see Figure 5.7). In a mashup-based approach the services can be easily composed to create a situational solution; however the result is not that stable. On the other hand a SOA-based approach provides a stable basis for creating and running the business processes for expert users who are familiar with complex stack of SOA, but a novice user will not be able to create solutions based on available services.

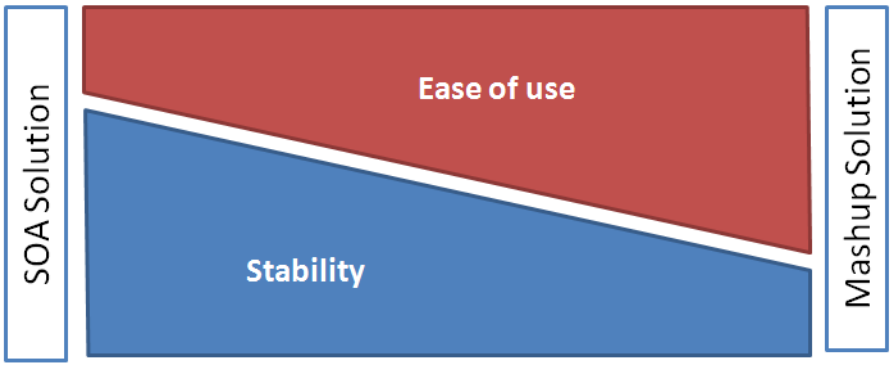

Figure 5.7: SOA Solution vs. Mashup Solution

To change this situation, a novel approach will be proposed that benefits from advantages of both SOA and Mashup solutions to facilitate creation of situational solutions that are also stable and business process can rely on them.

The core idea of the proposed approach is to break the design-time and runtime processes between Mashup and SOA environments respectively. In other words the solutions will be first created in a mashup environment and benefits from all advantages of mashup solutions. As the next step the Mashup will be translated to a formal business process using a Mashup-BPEL convertor. Finally any stable BPEL engine can run the process and take care of process management issues. As a result the end user will benefit from simple service composition of mashups and at the same time the process will be managed and controlled by well established business process engines. In this way, the process of running a mashup is safely transferred from the browser on the client computer to a BPEL engine on server-side. This approach might not be that helpful for presentation and data mashups, but will be of great advantage for enterprise mashups.

It is also important to note that, the proposed approach is not trying to reinvent the wheel and duplicate a BPEL editor. Unlike a typical BPEL editor, the proposed approach hides the complexities of a business process such as defining the partner links, services, etc and saves the end-users from technical details and they can solely concentrate on the their use case.

The challenging part of this approach is of course the Mashup-BPEL convertor that should translate the mashup steps to formal steps of a business process. The basic tasks of an Enterprise Mashup can be roughly categorized as follows:

- Data manipulation
- Formatting and presentation
- Support of conditional cases
- Data acquisition

Interestingly many of the known tasks of a mashup can be also described and implemented by BPEL methods. As a proof of concept, a list of mashup tasks and their equivalent BPEL methods are presented in Table 5.1.

Task Category	Examples	BPEL equivalent methods
Data manipulation	• adding result sets • subtracting result sets • merging datasets • applying data filters	• assign and copy elements • XPath • Calling external services
Formatting and presentation	• Styling the data • Format conversions	• XSL transformation (bpel:doXslTransform)
Support of conditional cases	• If-then-else • while	• structured activities (if, while, sequence, etc)
Data acquisition	• Calling web services / REST services • RSS feeds • User inputs	• Invoking web services • Invoking the wrapper web services

Table 5.1: Mashup tasks and their equivalent BPEL methods

To demonstrate the feasibility of this approach a small use case will be presented that shows how a simple BPEL process can be generated from its corresponding mashup. This use case is based on the basic math services that can be combined to do more complex calculations. Figure 5.8 depicts a simple Mashups that uses math services and delivers the result to display.

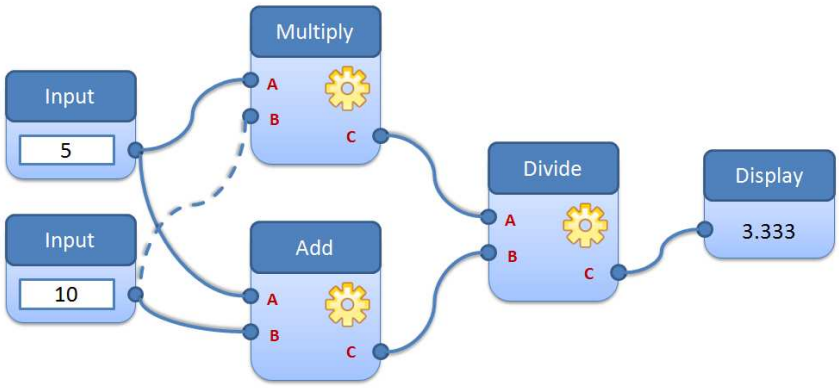

Figure 5.8: Simple calculation Mashup

The corresponding BPEL process for this mashup is shown in Figure 5.9. The Mashup-BPEL convertor uses the available widgets and connections of mashups and formulates the widget services as BPEL invoke actions. The resulted business process can be then deployed on any BPEL engine for execution. At runtime, user will submit the required input values to this process and the rest of steps will be handled by BPEL engine.

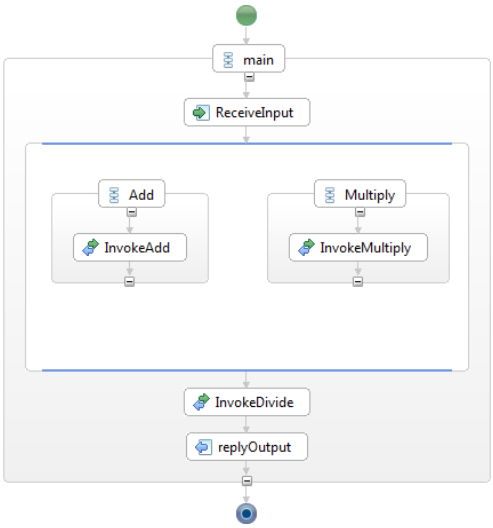

Figure 5.9: The generated BPEL process for Simple calculation Mashup

5.3 Human Interaction using XForms

The proposed approach of previous section for converting the mashups to BPEL processes can get more complicated when end-user's contribution is required. Unlike mashups that the user inputs are simply added on screen, the BPEL does not include the human interactions in its domain. Despite wide acceptance of Web services in distributed business applications, the absence of human interactions is a significant gap for many real-world business processes. To fill this gap, BPEL4People extends BPEL from orchestration of Web services alone to orchestration of role-based human activities as well (BPEL4People, 2009).

In context of the proposed approach, the required user interactions are implemented using XForms to send the missing data to the business processes and provide a simplified BPEL4People integration. To clarify this, consider the previous example that was presented in Figure 5.8 and suppose that the dashed connection between

input and multiply widget is removed. As a result the corresponding BPEL can be run in one service call and will break on "multiply web service" invocation. At this point the end-user needs to provide the missing parameter for continuation of business process.

In order to include the user contribution in the target business process, the Mashup-BPEL convertor will detect the location of missing parameter and inserts a people notification web service call before this location. In the proposed example an invoke activity (`InvokePeopleNotification`) and its corresponding receive activity are added before invocation of multiply web service as depicted in Figure 5.10. The functions of these two extra activities are as follows:

- `InvokePeopleNotification` will call a web service that is responsible to notify the end-user about required interaction via appropriate communication method such as email, instant message, SMS, etc. This web service creates a link and sends it to user via the selected communication method. By browsing the created link, an XForm will be displayed that will submit the required data to the business process.
- `ReceiveMissingPart` will wait for the user input that will be contributed via the XForm and then complete the missing parts of messages with the received data. In the proposed use case the submitted values will be used to prepare the input message of multiply web service.

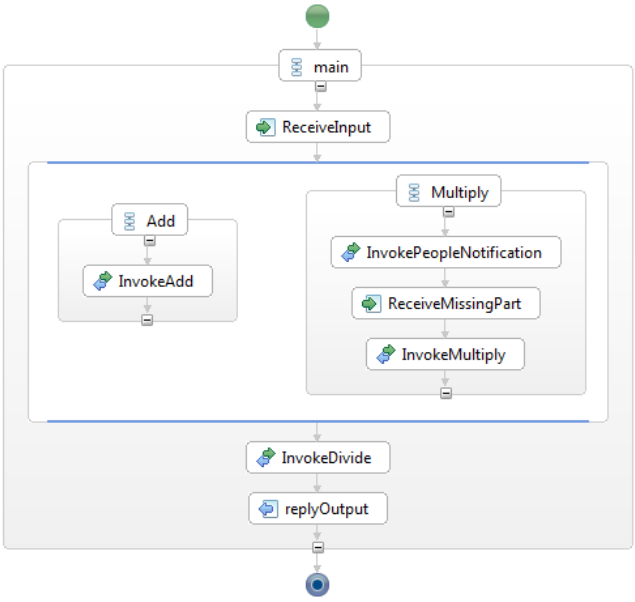

Figure 5.10: The generated BPEL process for People Interaction

One of the great advantages of XForms is the fact that an XForms client can send its data as XML. This capability can be elegantly used to send a SOAP message which is also encoded in XML to end point of web service. In the context of the proposed solution, this feature of XForms has been used to receive the missed parameters from user. Listing 5.2 shows an XForms example that displays the missing parameter as an input field and asks the user to complete and submit it.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ev="http://www.w3.org/2001/xml-events"
      xmlns:conf=http://sample-conference.com/registration
       xml:lang="en">

  <head>
    <title>XForms Web Service Call</title>

   <xforms:model id="MultiplyService">

     <xforms:instance id="messages">

       <SOAP-ENV:Envelope
            xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/1999/XMLSchema">
          <SOAP-ENV:Body>
             <ns1:getTemp xmlns:ns1="urn:xmethods-Temperature"
                SOAP-ENV:encodingStyle=
                       "http://schemas.xmlsoap.org/soap/encoding/">
                <b xsi:type="xsd:double">0</b>
             </ns1:getTemp>
          </SOAP-ENV:Body>
       </SOAP-ENV:Envelope>

     </xforms:instance>
```

```
     <xforms:submission id="calculation"
           method="text-xml-post"
           replace="instance"
           action="http://localhost:9080/Mashup/BPEL_CalculateMashup"/>

   </xforms:model>

   <xforms:input ref="instance('messages').//b">
     <xforms:label>b value: </xforms:label>
     <xforms:hint>Provide the missing value for Multiply service.
     </xforms:hint>
  </xforms:input>

</body>
</html>
```

Listing 5.2: XForms example for calling a web service

The given XForms of Listing 5.2, is generated by the servlet that reads the metadata of missing parameter (this has been previously stored by `PeopleNotification` web service) and return the XForms to end-user. By submission of this form the data will be directly sent to business process and the `ReceiveMissingPart` activity of this business process will receive the data and will follow with the rest of business process steps.

5.4 System Architecture

To address the requirement of these use-cases, a three-layer architecture is proposed. Figure 5.11 demonstrates the main idea of the proposed architecture graphically.

- **Service layer:** This layer provides the basic data acquisition and analysis methods for handling the web data. It includes the Web Form Services for dealing with web forms and also web annotation services to facilitate extraction of useful information from website.
- **Mashup layer:** The Mashup layer facilitates the integration of global web services which also includes the Web Form and annotation services in a uniform way. This layer also includes the required ontologies for describing the services and resulting Mashups.
- **Business Process layer:** The business layer includes the required services for translating mashups to formal business processes with added value of human interactions.

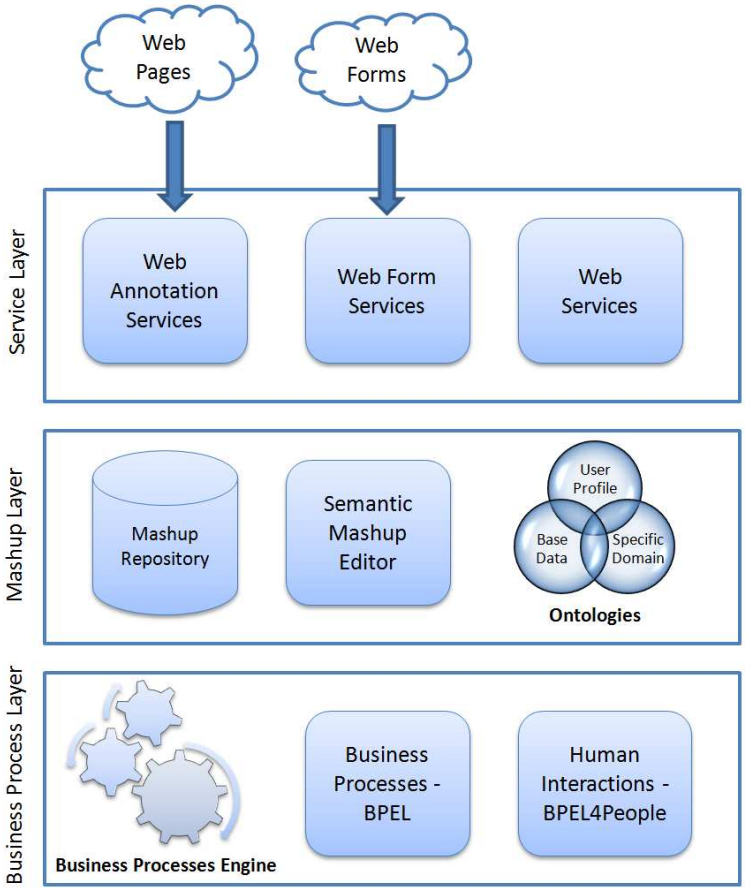

Figure 5.11: Semantic Mashup architecture overview

The details of system architecture and its components have been already explored in the previous sections. It is also interesting to note that the Semantic Desktop services which are supported by personal items (Personal Services) can also be used and integrated in Mashups via the web service component in the service layer.

In the context of the proposed architecture, the Semantic Mashups for enterprise applications are generally translated to a formal business process with human interaction support and get executed by business process engine. Alternatively any single widget of an Enterprise Mashups can be directly executed in the end-user's browser. This method is especially helpful to debugging the enterprise mashups and preparing them for getting translated to business processes. For this purpose user may call the `runWidget` (see Listing 5.3) remote method via the mashup editor. This method receives the widget ID and its input parameters and then tries to run the widget's embedded service by loading its corresponding WSDL file and executing the appropriate operation. The presented method in Listing 5.3 is a generic execution

method that dynamically calls the web service and returns the result as a `LinkedHashMap`.

```
public LinkedHashMap<String, String> runWidget
       (long widgetId, HashMap<String, String> inputs) {
       Session session = null;
       LinkedHashMap<String, String> results;
       Transaction transaction = null;
       try    {
              session = HibernateContext.getSessionFactory().openSession();

              WidgetType wt = (WidgetType)session.load(
                                          WidgetType.class, widgetId);
              switch (wt.getMethodId()) {
              case WidgetType.WEB_SERVICE:
                     try {
                            org.json.JSONObject details = new
                               org.json.JSONObject(wt.getDetails());
                            String wsdl = (String) details.get("wsdl");
                            String operation = (String)
                                                 details.get("operation");
                            return WebServiceInvoker.run(wsdl,
                                                 operation, inputs);
                     } catch (Exception e) {
                            e.printStackTrace();
                            break;
                     }

              case WidgetType.REST_SERVICE:

                     . . .

                     break;
              default:

                     . . .

                     break;
              }

              return null;
       } catch (RuntimeException e) {
              e.printStackTrace();
              throw e;
       } finally {
              if (session != null) {
                     session.close();
              }
       }
}
```

Listing 5.3: Running standalone mashup widgets

5.5 Implementation

The proposed approach of this thesis has been implemented as a prototype. A major part of the implementation activities are invested on the web based user interface that is of great importance for mashup frameworks. The main task of web based user interface is to support users in creating mashups.

In order to simplify the user interaction, the front end has been implemented using the Google Web Toolkit framework that has set a new trend in Rich Internet Application (RIA) world. The Google Web Toolkit (GWT, 2009) is an open source Java software development framework that allows web developers to develop Ajax applications in Java and benefit the Java best practices at implementation phase. However at runtime, the GWT compiler will translate the Java code to browser-compliant JavaScript and HTML that only needs a web server for launching the application pages and as a result no java-enabled server components will be needed at runtime.

The implemented system includes a mashup editor component that facilitates interaction of the end-user with registered services such as web services and Web Form Services. In order to import available services and make them visible to end-user, the WSDL file of required services should be provided. The server-side components of system will the take care of parsing the WSDL file and adding the web service operations to the system. The widgets and their parameters can be further improved by adding the semantic annotation using the existing domain ontologies. As an example consider the WSDL file of the simple math web service (see Figure 5.12) that can be added to system as widgets with all input and output parameters.

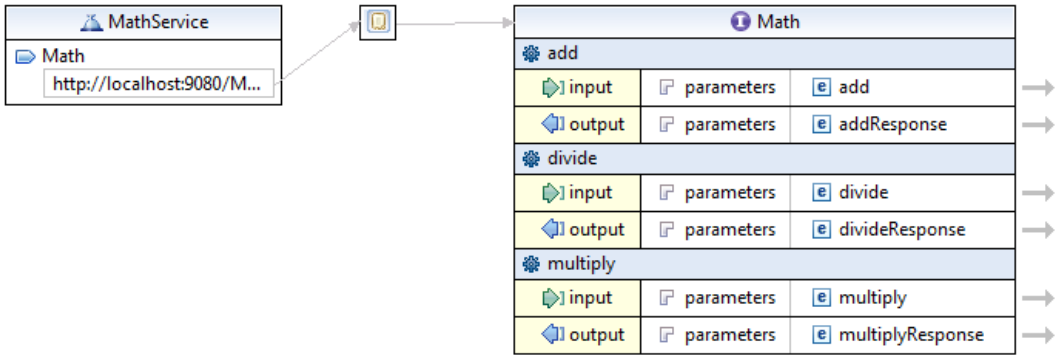

Figure 5.12: WSDL description of simple math web service

The imported operations will be then appear on the mashup editor interface as services (widgets) that can be combined with other services. Figure 5.13 shows the three basic services of simple math web service that are listed among mashup editor services. It also shows that the input and output parameters of these services are automatically imported from WSDL file and correctly displayed in mashups that use these services.

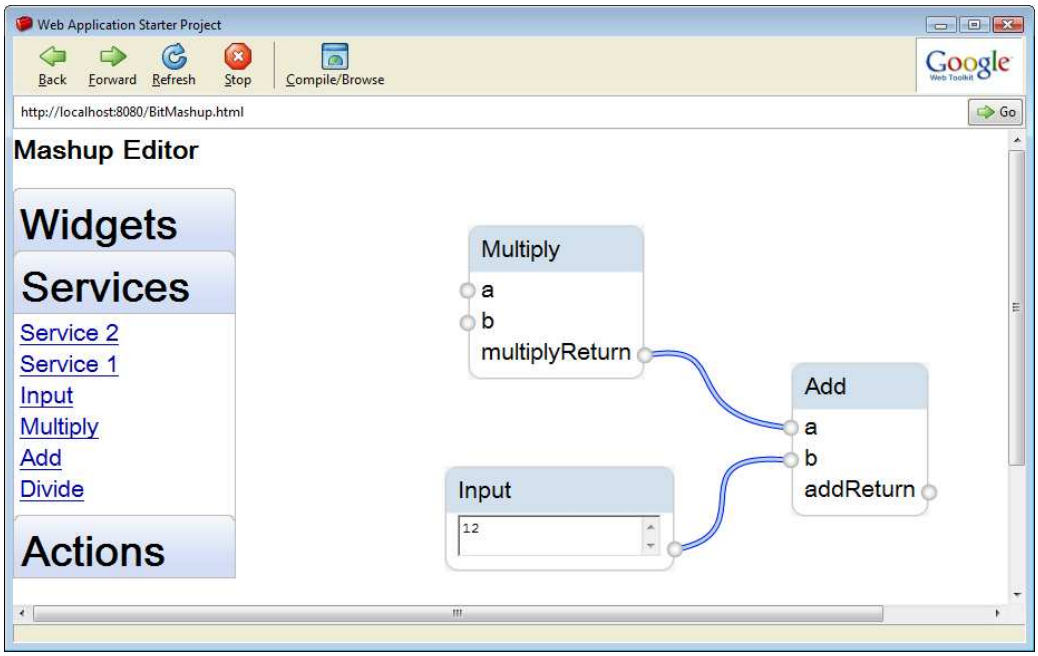

Figure 5.13: Mashup editor prototype

For implementation of mashup editor, canvas element has been used that is part of HTML 5 and allows for dynamic scriptable rendering of bitmap images. The canvas element realizes the connecting lines (wires) between mashup widgets that should also get updated whenever the widget elements are moved.

The Mashup Editor is also supported by a mashup repository that allows loading and storing of mashups. Mashup repository is composed of the following parts:

- A relational part that stores the widget types, widgets, and mashups:
    - Widget types are abstract services that can be used in composition of new mashups and are defined by their attributes such execution details, UI renderer class, ports, etc.
    - Mashup widgets are instantiated from widget types and store the visualization information of widget in the mashup context such as position.
    - Mashups that are composed of widgets and the connections (wiring) configuration between them.
- A semantic part that stores the semantic description of different item. The URI of these items are generated according to the ID of corresponding item in relational part and as a result the items can be traced bidirectional between relational and semantic parts. The semantic repository of mashups will be used for querying the appropriate services and supporting the users when creating mashups in mashup editor.

Figure 5.14 demonstrates the relational part of mashup repository and the relationship between different items.

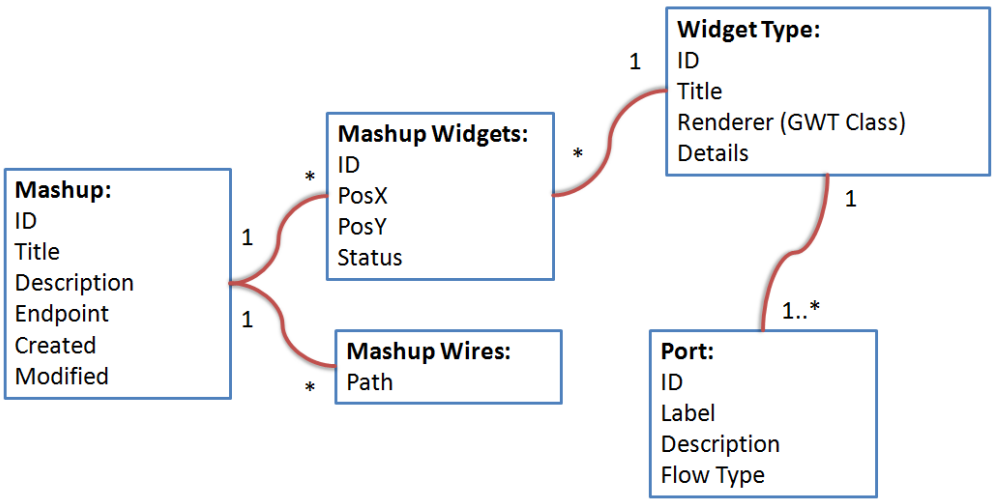

Figure 5.14: Relation part of mashup repository

The mashup configurations are transferred in JSON (JSON, 2009) convention which can be simply parsed and rendered in browsers. Listing 5.4 shows the mashup loading function that is used to load a mashup and render its widgets and connections on the page.

```
public void loadMashup() {
  mashupService.get("mashupId",
       new AsyncCallback<String>() {
         public void onFailure(Throwable caught) {
              GWT.log("Remote Procedure Call - Failure", null);
         }

         public void onSuccess(String result) {

              JSONValue json = JSONParser.parse(result);
              JSONArray widgets = json.isObject().get("widgets").isArray();

              for (int i = 0; i < widgets.size(); ++i) {

                JSONObject widgetObject = widgets.get(i).isObject();

                MashupWidget box = MashupFactory.makeWidget(
                     widgetObject.get("type").isString().stringValue(),
                     widgetObject.get("id").isString().stringValue(),
                     widgetObject.get("title").isString().stringValue());

              JSONValue portsNode = widgetObject.get("ports");
              if (portsNode != null) {
                JSONArray ports = widgetObject.get("ports").isArray();

                for (int j = 0; j < ports.size(); ++j) {

                     JSONObject port = ports.get(j).isObject();
                     box.addTerminal(
                            port.get("label").isString().stringValue(),
                            (int)port.get("type").isNumber().doubleValue());
                }
              }

              addMashupWidget(box,
                     (int)widgetObject.get("x").isNumber().doubleValue(),
                     (int)widgetObject.get("y").isNumber().doubleValue());

                                  }

              display();

              JSONArray wireNames = json.isObject().get("wires").isArray();

              for (int i = 0; i < wireNames.size(); ++i) {
                String wireName = wireNames.get(i).isString().stringValue();
                drawWire(wireName);

              }


            }
       });
}
```

Listing 5.4: Loading a mashup

In addition to mashup editor, the system is also supported by a collection of server side components such as WFS server, BPEL engine, and XForms generator for human interactions which are already explored in previous sections.

Chapter 6

RESULTS AND OUTLOOK

The shift away from traditional Web 1.0, is forced by the growing need for more efficient information sharing, collaboration and business processes. Mashup Architecture is one of the outcomes of Web 2.0 paradigm that has been widely accepted and used for user-centric information processing. At the moment mashups are mainly used for less fundamental tasks such as customized queries and map-based visualizations; however it has the potential to be used for more fundamental and sophisticated tasks too.

As more serious applications make use of mashup architecture, there is a growing need to study the business chances and feasible scenarios of mashup architecture and foster its applications for organizational use cases.

In my belief, mashups have the potential to facilitate the transition from traditional web to Semantic Web era and support this paradigm shift with “zero footprints” on the web pages. Furthermore the mashups can leverage the application of Semantic Desktops and facilitate the integration of their data into business processes.

The research presented in this thesis is aiming at demonstrating the uniform integration of different services such as web services, personal services, and Web Form Services in an easy way. More specifically, it shows how elaborate different information integration solutions in combination with mashups can be used to facilitate the information integration in business processes and support creation of situational solutions that can be shared and reused by other users. Furthermore the translation of enterprise mashups to stable business processes can accelerate the

application of mashups for more serious use cases such as organizational data sharing and data integration scenarios. Finally the propose solution of this thesis can be summarized in three pillars:

- Resources and basic services including
  - Personal services that are served by Semantic Desktops. In the presented solution, SemanticLIFE framework has been used and personal services are created using innovative concept of SOPA pipelines
  - Web Form Services that are served by WFS Server and turn the traditional web forms to formal web services
  - Other web resources such as web services, REST services, mashups, RSS feeds, GeoRSS feeds, etc.
- Mashup core components
  - Semantic Mashup Editor that supports query and composition of services based on their semantic description and as a result facilitates the creation of mashups for novice users.
  - BPEL convertor that allows translating the mashups to BPEL processes. As a result end-user will benefit from simple service composition of Mashups and at the same time the process will be managed and controlled by well established business process engines.
  - Human Interaction component that bridges the gap between human users and business process in the proposed framework using a sophisticated application of XForms.
- Mashup repository that stores the situational solution created for solving different information integration use cases. These mashups can be also shared and reused by other users for creating new solutions.

Figure 6.1 depicts an overview of proposed solution and its components which were described above.

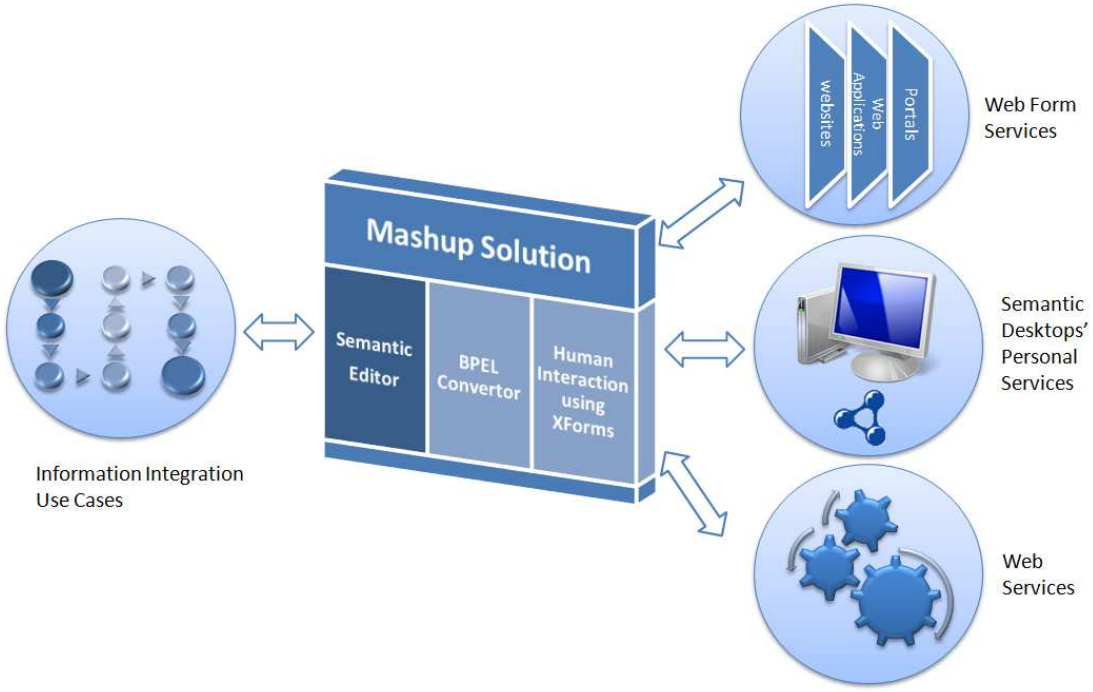

Figure 6.1: Overview of proposed Solution

In this rest of this section the research questions that were raised at the beginning of this thesis will be revisited to show how the proposed solution will address the challenging issues.

How can the semantics of personal information and their associations be modeled accurately for open world interactions?

Personal Services that are supported by Semantic Desktop solutions have the potential to share the user information with business processes and facilitate the user interaction with open world. These services are enriched with semantic information and can be queried based on the service parameters and/or service intent.

How to define personal information sharing services that can interact with global services and share useful information about a specific person in a secure and trustworthy way?

Creating the services that are able to query and share the appropriate data with global services is a challenging task. In the proposed approach of this thesis, the mashups

have been introduced as a user-driven micro-integration of web-accessible data that can be easily created and managed by novice users. Mashups and the security and privacy services of Semantic Desktops, create a solid basis for trustworthy information sharing with external services.

How can user requirements and preferences be represented and how should they be taken into account in tailoring global services to a particular user?

Each user has a unique set of requirements and preferences. Supporting the diverse use cases of all users in a single system is not realistic; however the mashups may bridge this gap by providing an easy method for service composition. In other words the end-users themselves can create their required services based on some elementary services and apply their requirements and preferences in the solutions they create.

How security and privacy policies can be applied to information flow between personal information and global web?

The security and privacy requirements can be supported by Semantic Desktops that on one hand have access to semantic repository of personal items and on the other hand can apply semantic rules and sharing policies to data interactions with the outside world.

How information integration solutions can be created and managed intuitively by end users who are not IT experts?

The mashups can facilitate the creation of situational solutions for the novice users. In the presented use cases of this thesis, some practical solutions were provided that shows how Semantic Web and domain ontologies can leverage the creation of new services using mashups. Especially the BPEL translation of mashups provides a solid basis for extending the footprint of mashups into more complex use cases and let the business processes to rely on them for implementing efficient data integration scenarios.

Appendixes

Appendix 1: SemanticLIFE's Pipeline component

```
package at.slife.pipeline;

import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Vector;

import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamSource;

import org.dom4j.*;
import org.dom4j.io.DocumentResult;
import org.dom4j.io.DocumentSource;
import org.dom4j.io.SAXReader;

import java.io.File;
import java.io.IOException;
import java.io.StringReader;

import at.slife.xmas.Call;


/**
 * This is the manin class that manages a pipeline, the main responsibility
of this class is
 * to parse the pipeline and create sub-elements like calls that do the
service calling
 * @author Amin Anjomshoaa
 */
public class Pipeline {

    /**
     * hash map of pipeline parmeters
     */
    private HashMap<String, Parameter> parameters;

    /**
     * body of pipeline as XML
     */
    private Element body = null;

    /**
     * the pipeline space name
     */
    private String pipeSpace = null;

    /**
     * callMap as XML document taht stores all pipelines
     */
    private static Document callMap = null ;
```

```
/**
 * pipeline results
 */
public String result = "";


  /**
   * pipeline constructor, to initialize the callMap at first run
   */
  public Pipeline() {
        if (null == callMap){
              loadPipelines();
        }

  }

  /**
   * reads the specific pipeline and extract the releated parameters
   * @param pipeName
   */
  public void loadPipeline(String pipeName){
        setPipeSpace(pipeName.substring(0,pipeName.indexOf(':')));

        Node node = callMap.selectSingleNode(
              "/pipelines/pipeline[@name='"+pipeName+"']" );

        body = (Element)node;

        parameters = new HashMap<String, Parameter>();
        Element params = (Element)body.selectSingleNode("parameters");
        System.out.println(params.asXML());
        for (Iterator i = params.elementIterator( "parameter" );
              i.hasNext(); ) {
       Parameter param = new Parameter((Element) i.next());
       parameters.put(param.getName(),param);
   }

  }


  /**
   * constructor to load a named pipeline
   *
   * @param theName pipeline name
   */
  public Pipeline(String theName) {
        if (null == callMap){
              loadPipelines();
        }

        setPipeSpace(theName.substring(0,theName.indexOf(':')));

        Node node = callMap.selectSingleNode(
           "/pipelines/pipeline[@name='"+theName+"']" );

        body = (Element)node;

        parameters = new HashMap<String, Parameter>();
        Element params = (Element)body.selectSingleNode("parameters");
        System.out.println(params.asXML());
        for ( Iterator i = params.elementIterator( "parameter" );
              i.hasNext(); ) {
       Parameter param = new Parameter((Element) i.next());
       parameters.put(param.getName(),param);
   }
```

```
	}


	/**
	 * returns the paramneter value of the given parameter name
	 * @param paramName the parameter name
	 * @return the value of given parameter
	 */
	public Parameter getParameter(String paramName){
		return parameters.get(paramName);
	}

	/**
	 * setter method for pipe space
	 * @param thePipeSpace
	 */
	private void setPipeSpace(String thePipeSpace) {
		pipeSpace = thePipeSpace;
	}

	/**
	 * getter method for pipe space
	 * @return the pipe space
	 */
	public String getPipeSpace(){
		return pipeSpace;
	}

	/**
	 * invokes the named pipeline with given parameters
	 * @param pipeName name of th epipeline to be invoked
	 * @param xmlParams pipeline parameters
	 * @return pipeline process results as string
	 */
	public String invokePipe(String pipeName,String xmlParams){
		result = "";
		loadPipeline(pipeName);
		setParameterValues(Util.parseParams(xmlParams));
			return processPipeline();
	}

	/**
	 * this is the main process that runs a pipeline at request.
	 * @return the pipeline call results
	 */
	public String processPipeline(){
	    for ( int i = 0, size = body.nodeCount(); i < size; i++ ) {
	      Node node = body.node(i);
	      String name = node.getName();
	      if (null != name){
	       // the sub-node is aso a call
	       if (name.startsWith("call")){
	         CallElement call = new CallElement((Element) node,this);
	             // the previous results will be cleaned
	                         if (name.equals("call-clean"))
	                                 result = call.invoke();
	                         else
	                                 result += call.invoke();
	       }
	       // the next node is an XSLT transformation
	       else if (name.equals("transform")){
	                         Element elm = (Element) node;
	                         transform(elm.attributeValue("xsl"));
	       }
	       // pipeline process is finished, result should be serializes
	       else if (name.equals("serialize")){
	                         Element elm = (Element) node;
```

```
              / call the relevant serializer (browser, form, xml, etc)
                      Call client = new Call(elm.attributeValue("service"));
                           try {
                                 Object[] content =
                                   {getResultDocument().asXML()};

      client.invoke(elm.attributeValue("operation"),content);
                           } catch (Exception e) {
                                 e.printStackTrace();
                           }

                 }
               }
         }
                 return result;
      }

      /**
       * setter method for parameter
       * @param name name of the parameter
       * @param val value of the parameter
       */
      public void setParameterValue(String name, String val) {
            Parameter param = parameters.get(name);
            param.setValue(val);
            parameters.get(name).setValue(val);
      }

      /**
       * setting all pipeline parameters from given string array
       * @param vals parameter values
       */
      public void setParameterValues(String[] vals) {
            int i = 0;
            for (String paramName : parameters.keySet()){
                  parameters.get(paramName).setValue(vals[i++]);
            }
      }

      /**
       * setting all pipeline parameters from given object array
       * @param vals parameter values
       */
      public void setParameterValues(Object[] vals) {
            int i = 0;
            for (String paramName : parameters.keySet()){
                  parameters.get(paramName).setValue(""+vals[i++]);
            }
      }

      /**
       * returns the value of given parameter name
       * @param name name of parameter
       * @return value of parameter
       */
      public String getParameterValue(String name) {
            return parameters.get(name).getValue();
      }

      /**
       * loads all pipelines from pipeline folder and stores them in
callMap for
       * further processes
       */
      private void loadPipelines(){

            File dir = null;
```

```
            try {
                  dir = PipelinePlugin.getResource("pipelines");
            } catch (IOException e1) {
                  e1.printStackTrace();
            }

            SAXReader reader = new SAXReader();

            callMap = DocumentHelper.createDocument();
       Element root = callMap.addElement( "pipelines" );;

            File[] files = dir.listFiles();

            for(int i=0,count=files.length;i<count;i++) {
                  String fileName = files[i].getName();
                  if (fileName.toUpperCase().endsWith(".XML")) {
                        String prefix =
                          fileName.substring(0,fileName.indexOf('.'));

                    try {
                        Document doc = reader.read(files[i]);
                        Element pipes = doc.getRootElement();

                        for ( int j = 0, size = pipes.nodeCount();
                             j < size; j++ ) {
                             Node node = pipes.node(j);
                             if (null != node && node.hasContent()){
                              Element elm = (Element)node;
                              elm.addAttribute("name",
                              prefix+":"+elm.attributeValue("name"));
                              root.add(elm.detach());
                             }
                              }

                        } catch (DocumentException e) {
                              e.printStackTrace();
                        }
                  }

            }
            System.out.println(callMap.asXML());
      }


      /**
       * applys an XSLT transformation to the pipeline results
       * @param xslFile name of the XSL file
       */
      public void transform(String xslFile){
            File xsl = null;
            try {
                  xsl = PipelinePlugin.getResource("styles/"+xslFile);

              // load the transformer using JAXP
              TransformerFactory factory =
                        TransformerFactory.newInstance();
              Transformer transformer;
                  transformer = factory.newTransformer(
                      new StreamSource(xsl)
                  );

              // now lets style the given document
              DocumentSource source = new
                              DocumentSource(getResultDocument());
              DocumentResult target = new DocumentResult();
              transformer.transform( source, target );
```

```
            // return the transformed document
            result = target.getDocument().getRootElement().asXML();
        } catch (IOException e1) {
            e1.printStackTrace();
        } catch (TransformerConfigurationException e) {
            e.printStackTrace();
        } catch (TransformerException e) {
            e.printStackTrace();
        }
    }

    /**
     * running an XPATH query on the results to extract parameter values
     * for next calls
     *
     * @param xpathQuery the XPATH query string
     * @return the selected part of pipeline result as string
     */
    public String runXPathQuery(String xpathQuery) {
        String items="";
        List list = getResultDocument().
                    selectNodes( "/result"+xpathQuery );

        boolean isArray = false;
        if (list.size() > 1)
            isArray = true;

        for (Object obj : list){
            String name = obj.getClass().getName();
            String temp = "";
            if (name.equals("org.dom4j.tree.DefaultElement"))
                temp = ((Element)obj).asXML();
            else if(name.equals("org.dom4j.tree.DefaultAttribute"))
                temp = ((Node) obj).getText();
            else if (name.equals("org.dom4j.tree.DefaultText"))
                temp = ((Text) obj).getText();

            if (isArray)
                items += "<item>"+temp+"</item>";
            else
                items += temp;
        }
        return items;
    }


    /**
     * returns the pipeline results, the results are stored locally in
     * and will be returned to the calling program in XML format
     *
     * @return the pipeline results as string
     */
    public Document getResultDocument(){
      SAXReader reader = new SAXReader();
        Document doc = null;
        try {
            if (result.startsWith("<?xml"))
              result = result.substring(result.indexOf("?>")+2);
              doc = reader.read(
                 new StringReader("<result>"+result+"</result>"));
        } catch (DocumentException e) {
            e.printStackTrace();
        }
        return doc;
    }
```

```
	/**
	 * returns the paramneter value of the given parameter name
	 * @param paramName the parameter name
	 * @return the value of given parameter
	 */
	public String[] getParamArray(){
		String[] params = new String[parameters.size()];
		int i = 0;
		for (String paramName : parameters.keySet())
			params[i++] = paramName;
		return params;
	}


}
```

Appendix 2: Pipeline examples from SOPA framework

```
<?xml version="1.0" encoding="UTF-8"?>
<pipelines>

  <!--  a simple pipeline for math calculations -->
  <pipeline name="square">

	<parameters>
	  <parameter name="input" type="int"/>
	</parameters>

	<call service="at.slife.test" operation="multiply" returns="int">
		<parameter type="int">{input}</parameter>
		<parameter type="int">{input}</parameter>
	</call>

	<serialize type="xml"/>
  </pipeline>

  <!--  a simple pipeline for math calculations -->
  <pipeline name="addPipe">
	<parameters>
		<parameter name="a" type="int"/>
		<parameter name="b" type="int"/>
	</parameters>
	<call service="at.slife.test" operation="add" returns="int">
	<parameter type="int">{a}</parameter>
	  <parameter type="int">{b}</parameter>
	</call>
	<transform xsl="test.xsl"/>
	<serialize type="xml"/>
  </pipeline>

  <!--  a simple pipeline for math calculations with fix inputs -->
  <pipeline name="multiplyPipe">
	<parameters/>
	<call id="first" service="at.slife.test" operation="multiply"
	     returns="int">
		<parameter type="int">5</parameter>
		<parameter type="int">10</parameter>
		</call>
		<transform xsl="test.xsl"/>
		<serialize type="xml"/>
	</pipeline>
```

```
<!--  a complex pipeline with multiple and nested calls -->
<pipeline name="sample">
     <parameters>
           <parameter name="a" type="int"/>
           <parameter name="b" type="int"/>
     </parameters>

     <call id="node" service="at.slife.test" operation="multiply"
                returns="int">
           <parameter type="int">{a}</parameter>
           <parameter type="int">{b}</parameter>
     </call>

     <call id="node" service="at.slife.test" operation="add"
           returns="int">
        <parameter type="int">{b}</parameter>
        <parameter type="int">
           <!-- calling a pipeline with given namespace and name -->
          <call service="at.slife.pipeline" operation="complex:square">
           <parameter type="int">{b}</parameter>
          </call>
        </parameter>
     </call>
     <transform xsl="transformer.xsl"/>

    <!-- calling remote services:localhost is replaced with IP Address-->
     <call id="node" service="at.slife.test@localhost"
           operation="add" returns="int">
       <parameter
           type="int">{xpath:/html/body/number[1]/text()}</parameter>
        <parameter
           type="int">{xpath:/html/body/number[2]/text()}</parameter>
     </call>
     <serialize type="xml"/>
</pipeline>

<!-- embedding a SPARQL query in a pipeline with broser serializer -->
<pipeline name="query">

     <parameters>
       <parameter name="predicate" type="String"/>
       <parameter name="format" type="String"/>
     </parameters>

    <call service="at.slife.query" operation="runQuery" returns="String">
       <parameter type="string">
           <![CDATA[
           PREFIX  rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
           PREFIX  rdfs:   <http://www.w3.org/2000/01/rdf-schema#>
           PREFIX  owl:    <http://www.w3.org/2002/07/owl#>
           PREFIX  daml:   <http://www.daml.org/2001/03/daml+oil#>
           PREFIX  google: <http://www.ifs.tuwien.ac.at/slife-google#>

           SELECT  ?subject ?object
           WHERE   { ?subject ]]>
           {predicate}
           <![CDATA[ ?object } ]]>
       </parameter>
       <parameter type="string">{format}</parameter>
     </call>
     <transform xsl="triple.xsl"/>
     <serialize service="at.slife.browser" operation="render"/>
</pipeline>

<!-- embedding a SPARQL query in a pipeline with specific serializer -->
<pipeline name="itemTimeline">
```

```
    <parameters/>
    <call service="at.slife.query" operation="runQuery" returns="String">
    <parameter type="string">
      <![CDATA[
      PREFIX  rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
      PREFIX  rdfs:   <http://www.w3.org/2000/01/rdf-schema#>
      PREFIX  owl:    <http://www.w3.org/2002/07/owl#>
      PREFIX  daml:   <http://www.daml.org/2001/03/daml+oil#>
      PREFIX  google: <http://www.ifs.tuwien.ac.at/slife-google#>

      SELECT  ?subject ?object
      WHERE   { ?subject ]]>
      {predicate}
      <![CDATA[ ?object } ]]>
    </parameter>
    <parameter type="string">xml</parameter>
    </call>
    <transform xsl="triple.xsl"/>
    <serialize service="at.slife.timeseries" operation="renderTime"/>
  </pipeline>


  <!--  a pipeline to call an external web service -->
  <pipeline name="check">
      <parameters>
            <parameter name="acccessCode" type="string"/>
            <parameter name="blz" type="string"/>
            <parameter name="account" type="string"/>
            <parameter name="country" type="string"/>
      </parameters>
      <call service="at.slife.wsrep" operation="callWS" returns="string">
         <!--  returns a soap response -->
        <parameter name="wsdl" type="string">test.wsdl</parameter>
        <parameter name="service" type="string">accountService</parameter>
        <parameter name="port" type="string">Account</parameter>
        <parameter name="operation" type="string">check</parameter>
        <parameter type="string">{acccessCode}</parameter>
        <parameter type="string">{blz}</parameter>
        <parameter type="string">{account}</parameter>
        <parameter type="string">{country}</parameter>
      </call>
      <serialize type="xml"/>
  </pipeline>
</pipelines>
```

Appendix 3: Part of Mashup Widget code

```
package com.bit.mashup.client;

import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.NoSuchElementException;

import com.google.gwt.user.client.ui.AbsolutePanel;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratorPanel;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.DOM;

public class MashupWidget extends Composite  {

    /**
     * The widget that goes into the header.
     */
    private Widget header;
    /**
     * The widget contains content to display in the center of the panel.
     */
    private AbsolutePanel content = null;
    public LinkedHashMap<String, Terminal> terminals = new
                          LinkedHashMap<String, Terminal>();
    private String widgetId = null;

    // absolute position of containing panel
    private int panelX;
    private int panelY;
    private static int last = 1;


    public void makeUnique() {
       widgetId += "_"+last;
       last++;
    }

    public int getRelativeLeft() {
            return this.getAbsoluteLeft() - panelX;
       }

    public int getRelativeTop() {
            return this.getAbsoluteTop() -  panelY;
       }

       public void setPanelPosition(int x, int y) {
              panelX = x;
              panelY = y;
       }

       // controls the visibility of terminal labels
    boolean visibleLabels = true;



    public MashupWidget(String theWidgetId, String widgetTitle) {

       widgetId = theWidgetId;
         this.header = new Label(widgetTitle);
         content = new AbsolutePanel();
```

```
        DecoratorPanelWithHeader panel = new DecoratorPanelWithHeader();
        initWidget(panel);

        setStyleName("roundedContainerWithHeader");
    }

    public void setVisibleLabels(boolean isVisible){
       visibleLabels = isVisible;
    }

    public void appendBody(Widget body){
        content.add(body);
    }

    public Widget getHandle() {
       return header;
    }

    public String getWidgetId(){
       return widgetId;
    }

    public void addTerminal(String terminalName, int theType) {
       Terminal t = new Terminal(terminalName,widgetId, theType);
       terminals.put(t.getName(), t);
       if (visibleLabels) {
              Label terminalLabel = new Label(terminalName);
              terminalLabel.setStyleName("terminal-Type-"+theType);
              appendBody(terminalLabel);
       }
    }


    public Terminal getTerminal(String name) {
       return terminals.get(name);
    }

    /**
     * Extend the DecoratorPanel.
     */
    private class DecoratorPanelWithHeader extends DecoratorPanel {

        private DecoratorPanelWithHeader() {
            setHeader(header);
            add(content);
        }

        /**
         * Set the widget that goes in the center of the top row.
         *
         * @param header The widget.
         */
        private void setHeader(Widget header) {
            DOM.appendChild(getCellElement(0, 1), header.getElement());
            adopt(header);
        }

        public Iterator<Widget> iterator() {
            final Iterator<Widget> superIterator = super.iterator();
            return new Iterator<Widget>() {
                boolean hasHeader = header != null;

                public boolean hasNext() {
                    return superIterator.hasNext() || hasHeader;
                }

                public Widget next() {
```

```
                    if (superIterator.hasNext()) {
                        return superIterator.next();
                    } else {
                        if (hasHeader && (header != null)) {
                            hasHeader = false;
                            return header;
                        } else {
                            throw new NoSuchElementException();
                        }
                    }
                }

                public void remove() {
                    throw new UnsupportedOperationException();
                }
            };
        }
    }

    /*
     * checks the mandatory input fields to make sure all
     * required fields are provided
     */
    public boolean isReady() {
      for (String terminalId: terminals.keySet()) {
            Terminal t = terminals.get(terminalId);
            if (!t.isOptional() && !t.isAttached()){
                  return false;
            }
            }
      return true;
    }

       public Terminal getWireEndTerminal(String wireId) {
            int pos = wireId.indexOf('/', wireId.indexOf('/')+1)+1;
            return terminals.get(wireId.substring(pos));
       }
}
```

Appendix 4: Wire class for connecting Mashup Widgets

```
package com.bit.mashup.client;

import gwt.canvas.client.Canvas;

/**
 * Wire Canvas for drawing connectors between Mashup widgets
 *
 * @author Amin Anjomshoaa
 */
public class WireCanvas extends Canvas {

	// relative position of start
	private Point start = null;

	// absolute position of canvas
	private Point position = null;

	// coordinate of end point
	private int endPosX = -1;
	private int endPosY = -1;

	public WireCanvas(int startX, int startY) {
		start = new Point(0, 0);
		position = new Point(startX,startY);
		setBackgroundColor(TRANSPARENT);
		setStyleName("connector-wire");
	}

	public int getEndPosX(){
		return endPosX;
	}

	public int getEndPosY(){
		return endPosY;
	}

	/*
	 * stores end position locally
	 */
	public void saveEndPosition(int x , int y){
		endPosX = x;
		endPosY = y;
	}

	public void setStart(int startX , int startY){
		start = new Point(startX-position.x, startY-position.y);
	}

	/*
	 * returns the appropriate canvas position
	 * the input coordinates values are absolute
	 */
	public Point drawTo(int endX, int endY) {
		// make target position non-absolute
		Point end = new Point(endX - position.x,endY - position.y);

		int coeffMulDirection = 100;

	   int distance = start.distanceTo(end);

	   if(distance < coeffMulDirection){
	      coeffMulDirection = distance / 2;
	   }
```

```
        Point d1 = new Point(1 * coeffMulDirection, 0);
        Point d2 = new Point(-1 * coeffMulDirection, 0);

        Point[] points = new Point[4];
        points[0] = start;
        points[1] = new Point(start.x + d1.x, start.y +d1.y);
        points[2] = new Point(end.x + d2.x, end.y + d2.y);
        points[3] = end;

            Point min = start.clone();
            Point max = start.clone();

            for (int i=1; i < 4; i++) {
                  if (points[i].x < min.x) {
                        min.x = points[i].x;
                  }
                  if (points[i].y < min.y) {
                        min.y = points[i].y;
                  }

                  if (points[i].x > max.x) {
                        max.x = points[i].x;
                  }
                  if (points[i].y > max.y) {
                        max.y = points[i].y;
                  }
            }


               int lw = max.x - min.x;
               int lh = max.y - min.y;

                   setPixelSize(lw+20, lh+20);
                   clear();

               for (int i= 0; i < 4;i++) {
                     points[i].moveDelta( -1 * min.x , -1 * min.y );
               }

               // Draw the border
               setLineCap(Canvas.ROUND);
               setStrokeStyle("#0000ff");
               setLineWidth(5);
               beginPath();
               moveTo(points[0].x+15,points[0].y+15);
               cubicCurveTo(points[1].x,points[1].y,points[2].x,
                            points[2].y,points[3].x+10,points[3].y+10);
               stroke();

               // Draw the inner bezier curve
               setLineCap(Canvas.ROUND);
               setStrokeStyle("#ADD8E6");
               setLineWidth(3);
               beginPath();
               moveTo(points[0].x+15,points[0].y+15);
               cubicCurveTo(points[1].x,points[1].y,points[2].x,
                            points[2].y,points[3].x+10,points[3].y+10);
               stroke();

               position.moveDelta(min.x, min.y);
               return position;
      }

}
```

Appendix 5: Part of Mashup Editor code

```
package com.bit.mashup.client;

import java.util.Collection;
import java.util.HashMap;
import java.util.LinkedHashMap;

import com.bit.mashup.client.remote.MashupRemote;
import com.bit.mashup.client.ui.WidgetLink;
import com.bit.mashup.domain.Mashup;
import com.bit.mashup.domain.WidgetType;
import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedStackPanel;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.VerticalPanel;

public class MashupEditor extends Composite {

	private static MashupPanel mashupPanel;
	// Mashup Data Object
	private Mashup mdo = null;
	private VerticalPanel services = null;

	public MashupEditor(){
		HorizontalPanel panel = new HorizontalPanel();
		mashupPanel = new MashupPanel(800, 500);

	    DecoratedStackPanel stackPanel = new DecoratedStackPanel();
	    stackPanel.setWidth("200px");

	    . . .

	    services = new VerticalPanel();
		listServices();
		stackPanel.add(services,"Services",false);
		VerticalPanel actions = new VerticalPanel();
		stackPanel.add(actions,"Actions",false);
		Button  loadButton = new Button("Load Mashup");
		actions.add(loadButton);
		loadButton.addClickHandler(new ClickHandler() {
			public void onClick(ClickEvent event) {
				loadMashup();
			}
		});
		Button  saveButton = new Button("Save Mashup");
		actions.add(saveButton);

		saveButton.addClickHandler(new ClickHandler() {
			public void onClick(ClickEvent event) {

				saveMashup();

			}
		});

		Button  runButton = new Button("Run Mashup");
		actions.add(runButton);
```

```
		runButton.addClickHandler(new ClickHandler() {
			public void onClick(ClickEvent event) {

				runMashup();

			}
		});

	    panel.add(stackPanel);
	    panel.add(mashupPanel);

	    initWidget(panel);

	}

	public static void addBox(MashupWidget aWidget){
		mashupPanel.addMashupWidget(aWidget, 100, 100);
	}

	public void loadMashup(){
		mashupPanel.clear();
		mashupPanel.loadMashup();
	}

	public void listServices(){
		MashupRemote.Util.getInstance().listWidgetTypes(
			 new AsyncCallback<Collection<WidgetType>>(){

			@Override
			public void onFailure(Throwable caught)
			{
				Window.alert(caught.getMessage());
			}
			@Override
			public void onSuccess(Collection<WidgetType> widgets)
			{
			    for (WidgetType widgetType : widgets) {
					services.add(new WidgetLink(widgetType));
				}
			}
		});
	}

	public void saveMashup(Mashup mdo){
			mdo.setBoxList(mashupPanel.getBoxes());
			mdo.setPathList(mashupPanel.getPaths());

			MashupRemote.Util.getInstance().saveMashup(
				 mdo, new AsyncCallback<Long>(){
				@Override
				public void onFailure(Throwable caught)
				{
					Window.alert(caught.getMessage());
				}

				@Override
				public void onSuccess(Long result)
				{
					Window.alert("Mashup has been saved");
				}

			});
		}
	}
}
```

Appendix 6: Part of Mashup Panel code

```
package com.bit.mashup.client;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Set;

import com.allen_sauer.gwt.dnd.client.DragContext;
import com.allen_sauer.gwt.dnd.client.drop.AbsolutePositionDropController;
import com.bit.mashup.domain.Box;
import com.bit.mashup.domain.Path;
import com.google.gwt.core.client.GWT;
import com.google.gwt.json.client.JSONArray;
import com.google.gwt.json.client.JSONObject;
import com.google.gwt.json.client.JSONParser;
import com.google.gwt.json.client.JSONValue;
import com.google.gwt.user.client.DOM;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.AbsolutePanel;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Widget;

public class MashupPanel extends AbsolutePanel {

	private HashMap<String, WireCanvas> wires =
		  new HashMap<String, WireCanvas>();
	private WireCanvas currentWire = null;
	private String currentWireId = null;
	private HashMap<String, MashupWidget> widgets =
		  new HashMap<String, MashupWidget>();

	private static MashupWidgetDragController mashupDragger = null;
	private static AbsolutePositionDropController mashupDropper = null;

	private static ConnectorDragController dragController = null;

	/**
	 * Create a remote service proxy.
	 */
	private final MashupServiceAsync mashupService = GWT
			.create(MashupService.class);

	private boolean stored = false;

	// absolute position of mashup panel
	private int top = 0;
	private int left = 0;

	public int getPanelTop() {
		return top;
	}

	public int getPanelLeft() {
		return left;
	}

	public MashupPanel(int width, int height) {
		setPixelSize(width, height);

		if (dragController == null) {

		  dragController = new ConnectorDragController(this, false){
```

```
                protected Widget newDragProxy(DragContext context) {
                  AbsolutePanel container = new AbsolutePanel();
                  DOM.setStyleAttribute(container.getElement(),
                                        "overflow", "visible");
              return container;
            }
                  };
                  dragController.setBehaviorDragProxy(true);
            }

            // drag & drop controllers of MashupWidget
            mashupDragger = new MashupWidgetDragController(this,false);
            mashupDragger.setBehaviorConstrainedToBoundaryPanel(true);
            mashupDragger.setBehaviorMultipleSelection(false);

          // instantiate our drop controller
           mashupDropper = new AbsolutePositionDropController(this);
          mashupDragger.registerDropController(mashupDropper);

        }

        public void drawWire(String wireName) {
              String[] parts = wireName.split("/");

              MashupWidget startWidget = widgets.get(parts[0]);
              Terminal startTerminal =
                    startWidget.terminals.get(parts[0]+"/"+parts[1]);

              MashupWidget endWidget = widgets.get(parts[3]);
              Terminal endTerminal =
                    endWidget.terminals.get(parts[2]+"/"+parts[3]);

              Point start = startTerminal.getAbsolutePosition();
              //start.moveDelta(-1*left, -1*top);
              WireCanvas wire = new WireCanvas(start.x-left, start.y-top);
              add(wire,start.x, start.y);
              wires.put(wireName, wire);

              Point end = endTerminal.getAbsolutePosition();
              end.log("drawWire");
              wire.saveEndPosition(end.x-left, end.y-top);
              moveEndOfWireTo(wireName, end.x, end.y);
        }

        public void addMashupWidget(MashupWidget widget, int x, int y){
          mashupDragger.makeDraggable(widget, widget.getHandle());
          widgets.put(widget.getWidgetId(), widget);
              add(widget,x,y);
              displayWidget(widget);
        }

        public void addWire(String startTerminal, int startX, int startY) {
              currentWire = new WireCanvas(startX-left, startY-top);
              currentWireId = startTerminal;
              add(currentWire,startX-left,startY-top);
        }

        public void removeCurrentWire() {
              remove(currentWire);
              currentWire = null;
              currentWireId = null;
        }

        public void removeWire(String wireId) {
              WireCanvas wire = wires.get(wireId);
              remove(wire);
              wires.remove(wireId);
```

```
	}

	/*
	 * Displays mashup and draws relevant components
	 */
	public void display() {
		for (String wname : widgets.keySet()) {
			displayWidget(widgets.get(wname));
		}
	}

	public void displayWidget(MashupWidget widget){
		top = this.getAbsoluteTop();
		left = this.getAbsoluteLeft();
		widget.setPanelPosition(left, top);
		int width = widget.getOffsetWidth();
		int delta = (widget.getOffsetHeight() - 50) /
				widget.terminals.size();
		int termY = widget.getRelativeTop()+15;
		int i = 1;
		for (String tname : widget.terminals.keySet()) {
			Terminal t = widget.getTerminal(tname);
			if (t.getType() == Terminal.OUTPUT_PORT) {
				dragController.makeDraggable(t);
				t.setPosition(width - 15, 15 + (delta * i));
				add(t, widget.getRelativeLeft()+ width -15,
					termY + (delta * i));
			} else {
				TerminalDropController dropControlle r =
				   new TerminalDropController(t);

	dragController.registerDropController(dropController);
		t.setPosition(-15, 15 + (delta * i));
		add(t, widget.getRelativeLeft()-15, termY + (delta * i));
		}
		i++;
	   }
	}

	/*
	 * fixes the currentWire
	 */
	public void fixWire(String targetId, int endX, int endY){
		String wireId = currentWireId+"/"+targetId;
		currentWire.saveEndPosition(endX-left, endY-top);
		wires.put(wireId, currentWire);
		moveEndOfWireTo(wireId, endX, endY);
	}

	public void redrawCurrentWire(int theLeft, int theTop){
		Point pos = currentWire.drawTo(theLeft, theTop);
		DOM.setStyleAttribute(currentWire.getElement(), "left",
					pos.x+"px");
		DOM.setStyleAttribute(currentWire.getElement(), "top",
					pos.y+"px");
	}

	public void moveStartOfWireTo(String wireId, int newLeft,
							int newTop){
		WireCanvas wire = wires.get(wireId);
		wire.setStart(newLeft-this.left, newTop-this.top);

		Point pos = wire.drawTo(wire.getEndPosX()+5,
					wire.getEndPosY()+5);
		DOM.setStyleAttribute(wire.getElement(), "left", pos.x+"px");
		DOM.setStyleAttribute(wire.getElement(), "top", pos.y+"px");
	}
```

```
	public void moveEndOfWireTo(String wireId, int newLeft, int newTop){
		WireCanvas wire = wires.get(wireId);
		Point pos = wire.drawTo(newLeft-left+5, newTop-top+5);
		DOM.setStyleAttribute(wire.getElement(), "left", pos.x+"px");
		DOM.setStyleAttribute(wire.getElement(), "top", pos.y+"px");
	}


	public ArrayList<String> getOutWires(String widgetId){
		ArrayList<String> outWires = new ArrayList<String>();
	    for (String wireId : wires.keySet()) {
		   if (wireId.startsWith(widgetId)) {
			outWires.add(wireId);
		   }
	    }
	    return outWires;
	}

	/* extract all wires that are connected to input ports of
	 * the given widget
	 */
	public ArrayList<String> getInWires(String widgetId){
		ArrayList<String> inWires = new ArrayList<String>();
	    for (String wireId : wires.keySet()) {
		   if (wireId.endsWith(widgetId)) {
			inWires.add(wireId);
		   }
	    }
	    return inWires;
	}

	public void saveEndPosition(String inWire,int theLeft,int theTop) {
		WireCanvas wire = wires.get(inWire);
		wire.saveEndPosition(theLeft-left, theTop-top);
	}

	public void loadMashup() {
	   . . .
	}

	public Set<Box> getBoxes() {
		Set<Box> boxes = new HashSet<Box>() ;
		for (String wname : widgets.keySet()) {
			MashupWidget widget = widgets.get(wname);
			Box box = new Box();
			box.setPosx(widget.getAbsoluteLeft());
			box.setPosy(widget.getAbsoluteTop());
			//box.setWidgetType(new WidgetType());
			boxes.add(box);
		}
		return boxes;
	}

	public Set<Path> getPaths() {
		Set<Path> paths = new HashSet<Path>() ;
		for (String wname : wires.keySet()) {
			//WireCanvas wire = wires.get(wname);
			Path path = new Path();
			path.setAddress(wname);
			paths.add(path);
		}
		return paths;
	}
}
```

Appendix 7: A simple input widget that extends the MashupWidget class

```
package com.bit.mashup.client;

import com.google.gwt.user.client.ui.TextArea;

public class InputWidget extends MashupWidget {

	private TextArea field = new TextArea();

	public InputWidget(String theWidgetId,String theTitle) {
		super(theWidgetId, theTitle);
		appendBody(new TextArea());
		setVisibleLabels(false);
	}

	public String run(){
		return field.getValue();
	}

}
```

Bibliography

Anjomshoaa (2009), Amin Anjomshoaa, Gerald Bader, and Amin Tjoa, Exploiting Mashup Architecture in Business Use Cases, The 12th International Conference on Network-Based Information Systems (NBIS 2009), Indianapolis USA

Anjomshoaa (2006), Anjomshoaa, A. , Karim, S., Shayeganfar, F. , Tjoa A M. (2006), Exploitation of Semantic Web technology in ERP systems, Procs. Of Confenis 2006

Apache Cocoon (2009), The Apache Cocoon Project, http://cocoon.apache.org/, last visited November 2009

Bell, Michael (2008). "Introduction to Service-Oriented Modeling". Service-Oriented Modelling: Service Analysis, Design, and Architecture. Wiley & Sons. pp. 3. ISBN 978-0-470-14111-3

Berners-Lee, T. (2001), The Semantic Web, Scientific American, May 2001

Blackman (2007), Andreas Ekelhart, Stefan Fenz, Gernot Goluch, Markus D. Klemen, Edgar R. Weippl , Architectural approach for handling semi-structured data in a user-centered working environment, International Journal of Web Information Systems, 2007, Volume 3, pp. 198-211, ISSN 1744-0084, DOI 10.1108/17440080710834247

BPEL Sub-processes (2009), BPEL for sub-processes, http://www-28.ibm.com/developerworks/webservices/library/specification/ws-bpelsubproc/, last visited November 2009

BPEL4People (2009), BPEL4People, http://en.wikipedia.org/wiki/BPEL4People, last visited November 2009

Bush, V. (1945). As we may think. The Atlantic Monthly 176(1) (1945) p101–108

Cherbakov (2007a), Cherbakov, L., Bravery, A., Goodman, B., Pandya, A., Bagget, J.: Changing the corporate IT development model: Tapping the power of grassrots computing. IBM System Journals 46(4), 2007

Cherbakov (2007b), Cherbakov, L., Bravery, A., Pandya, A.: SOA meets Situational Applications: Changing Computing in the Enterprise , 2007, http://www.ibm.com/developerworks/

Cocoon (2009), Apache Cocoon Framework, http://cocoon.apache.org/, last visited November 2009

CRA (2003), CRA Conference on "Grand Research Challenges in Information Security & Assurance", Final Report: Four Grand Challenges in Trustworthy Computing, p. 23

Daniel (2007), Daniel, F., Matera, M., Yu, J., Benatallah, B., Saint-Paul, R., Casati, F.: Understanding UI Integration. A Survey of Problems, Technologies, and Opportunities. IEEE Internet Computing 11(3), 59–66, 2007

Dapper (2009), Dapper homepage, http://www.dapper.net/ , last visited November 2009

Data Integration (2009), http://en.wikipedia.org/wiki/Data_integration , last visited November 2009

Dornan (2007), Andy Dornan, Mashup Basics: Three for the Money, 2007 http://www.networkcomputing.com/data-networking-management/mashup-basics-three-for-the-money.php , last visited November 2009

Eclipse Plug-ins (2009), Azad Bolour, Notes on the Eclipse Plug-in Architecture, http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html , last visited November 2009

EIU (2007), The Economist Intelligence Unit: Serious Business - Web 2.0 goes Corporate.

FAST (2008), FAST: EU Project, INFSO-ICT-216048 (2008), http://fast.morfeo-project.eu/, last visited November 2009

Feldman, S. (2005), Susan Feldman, Joshua Duhl, The Hidden Costs of Information Work, IDC White paper, March 2005

Fitzgibbon (2004), Fitzgibbon, A. and Reiter, E.; Memories for life: managing information over a human lifetime, Grand Challenges in Computing Research, pages 13-16. The British Computer So-ciety., http://www.ukcrc.org.uk/gcresearch.pdf

Forms (2009), W3C Forms Working Group, http://www.w3.org/MarkUp/Forms/, last visited November 2009

Gartner (2008), Top 10 Strategic Technologies for 2009, Gartner Symposium/ITxpo, http://www.gartner.com/it/page.jsp?id=777212, last visited November 2009

Gartner (2009), Gartner Five Business Intelligence Predictions for 2009, Gartner Business Intelligence Summit 2009, http://www.gartner.com/it/page.jsp?id=856714 , last visited November 2009

Geocoding (2009), Google Maps API Geocoding Service, http://code.google.com/apis/maps/documentation/geocoding/index.html , last visited November 2009

Google Maps (2009), Google Map homepage, http://maps.google.com , last visited November 2009

Google Sidewiki (2009), http://www.google.com/sidewiki/intl/en/index.html , last visited November 2009

Gruber, T. R. (1995), Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal Human-Computer Studies, 43(5-6):907-928, 1995.

Gnowsis project (2009), http://www.gnowsis.org/, last visited November 2009

GWT (2009). Google Web Toolkit. http://code.google.com/webtoolkit/, last visited November 2009.

Haystack (2009), Haystack project, http://groups.csail.mit.edu/haystack/, last visited November 2009

Hoyer (2008a), Volker Hoyer and Marco Fischer, Market Overview of Enterprise Mashup Tools, ICSOC 2008, Springer-Verlag Berlin Heidelberg, LNCS 5364, pp. 708–721, 2008.

Hoyer (2008b), Hoyer, V., Stanoevska-Slabeva, K., Janner, T., Schroth, C.: Enterprise Mashups: Design Principles towards the Long Tail of User Needs. In: IEEE International Conference on Services Computing (SCC), vol. 2, pp. 601–602, 2008

Hoyer (2009), Volker Hoyer and Katarina Stanoevska-Slabeva , The Changing Role of IT Departments in Enterprise Mashup Environments, ICSOC 2008, Springer-Verlag Berlin Heidelberg 2009, LNCS 5472, pp. 148–154, 2009.

HtmlUnit (2009), HtmlUnit, http://htmlunit.sourceforge.net/ , last visited November 2009

iGoogle (2009), iGoogle Homepage, http://www.google.com/ig, last visited November 2009

Information Integration (2009), http://en.wikipedia.org/wiki/Information_integration, last visited November 2009

IRIS Semantic Desktop (2009), http://www.openiris.org/ , last visited November 2009

JackBe (200), Jackbe company website, http://www.jackbe.com, last visited November 2009

Janner (2007), Janner, T., Canas, V., Hierro, J., Licano, D., Reyers, M., Schroth, C., Soriano, J., Hoyer, V.: Enterprise Mashups: Putting a face on next generation global SOA. In: Tutorial at the 8th Int. Conf. on Web Information Systems Engineering, 2007

JSON (2009), JavaScript Object Notation, http://json.org, last visited November 2009

KAoS (2003), Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., et al. (2003). KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement. Policy 2003: Workshop on Policies for Distributed Systems and Networks. Springer-Verlag.

Kapow (2009), Kapow technologies company website, http://kapowtech.com, last visited November 2009

Legner (2007), Legner, C.: Is there a Market for Web Services? - An Analysis of Web Services Directories. In: Proceedings of the 1st International Highlight Workshop on Web APIs and Services Mashups, 2007

Lixto (2008), Lixto Solutions, http://www.lixto.com, last visited November 2009

Mash Maker (2009), Intel Mash Maker, http://mashmaker.intel.com, last visited November 2009

Mashup Basics (2009), Mashup Basics: Three for the Money, Andy Dornan, http://www.networkcomputing.com/showitem.jhtml?articleID=201804223, last visited November 2009

Mashup Center (2009), IBM Mashup Center, http://www-01.ibm.com/software/info/mashup-center, last visited November 2009

Mashup Guide (2008), A Business Guide to Enterprise Mashups, JackBe Corporation, April 2008

Medjahed (2003), Brahim Medjahed, Boualem Benatallah, Athman Bouguettaya, Anne H. H. Ngu, Ahmed K. Elmagarmid, “Business-to-business interactions: issues and enabling technologies”,The VLDB Journal (2003) 12: pp 59–85.

METEOR-S (2009), Semantic Web Services and Processes, http://lsdis.cs.uga.edu/projects/meteor-s/, last visited November 2009

Microformats (2009), http://www.microformats.org, last visited November 2009

Music Mashups (2009), http://en.wikipedia.org/wiki/Mashup_(music), last visited November 2009

Gorza (2007), T. Groza, S. Handschuh, K. Moeller, G. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjonsdottir: The NEPOMUK Project - On the way to the Social Semantic Desktop. Procs. of I-Semantics' 07, 2007

Nepomuk (2009), Nepomuk Project Synopsis: http://nepomuk.semanticdesktop.org/xwiki/bin/download/Main1/Project+Summary/NEPOMUK-Synopsis.pdf, last visited November 2009

O'Reilly, T. (2005), What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software, September 2005, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html , last visited November 2009

O'Reilly, T. (2006), Levels of the Game: The Hierarchy of Web 2.0 Applications, July 2006, http://radar.oreilly.com/2006/07/levels-of-the-game-the-hierarc.html , last visited November 2009

O'Reilly Media (2007), Programming Collective Intelligence: Searching and Ranking, white paper, August 2007, http://whitepapers.techrepublic.com.com/abstract.aspx?docid=323363 , last visited November 2009

OWL-S (2009), Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S/, last visited November 2009

Popfly (2009), Microsoft Popfly, http://www.popfly.com, last visited November 2009

Programmable Web (2009), Programmable Web Top APIs, http://www.programmableweb.com/apis, last visited May 2009

QEDWiki (2009), QEDWiki homepage, http://services.alphaworks.ibm.com/graduated/qedwiki.html, last visited November 2009

REI (2003), Kagal, L., Finin, T., and Johshi, A. (2003). A Policy Language for Pervasive Computing Environment. Policy 2003: Workshop on Policies for Distributed Systems and Networks. Springer-Verlag.

Reliability (2009), Reliability of Wikipedia, http://en.wikipedia.org/wiki/Reliability_of_Wikipedia, last visited November 2009

RDFa (2009), RDFa in XHTML Syntax and Processing, http://www.w3.org/TR/2008/CR-rdfa-syntax-20080620, last visited November 2009

REST (2000), Representational State Transfer, Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures, Ph.D. Thesis, University of California, Irvine, Irvine, California, 2000.

RSS (2009), Really Simple Syndication, http://en.wikipedia.org/wiki/RSS, last visited November

Sauermann (2005), Sauermann L., Bernardi A., and Dengel A. (2005), Overview and Outlook on the Semantic Desktop, Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference

Schmelzer (2006), Roland Schmelzer, Rich Internet Applications: Market Trends and Approaches, zapThink foundation report, July 2006

Schmid (1998), Schmid, B., Lindemann, M.: Elements of a Reference Model for Electronic Markets. In: Proceedings of the 31st Hawaii Int. Conf. on System Sciences (HICSS) (1998)

Shayeganfar (2009), Ferial Shayeganfar, Application of Semantic Web Material Libraries in AEC Context , Ph.D. Thesis, Vienna University of Technology, Austria, 2009

SemanticLIFE (2004), 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, M.N. Tho, A.M. Tjoa: SemanticLIFE - A Framework for Managing Information of a Human Lifetime, Procs. Of the 6th International Conference on Information Integration and Web-based Applications and Services, 2004

Shuaib (2006), Shuaib Karim and A Min Tjoa, 'Towards the Use of Ontologies for Improving User Interaction for People with Special Needs', in ICCHP 2006 (10th Intl. Conference on Computers Helping People with Special Needs) 12-14th Jul'06, Linz - Austria . Published in LNCS by Springer Berlin / Heidelberg, volume 4061 / 2006, pp. 77-84 ,ISBN: 3-540-36020-4, DOI: 10.1007/11788713_12

Shuaib (2007), Shuaib Karim, Khalid Latif and A Min Tjoa 'Providing Universal Accessibility using Connecting Ontologies: A Holistic Approach , in HCII 2007 (12th International Conference on Human-Computer Interaction) 22-27th July'07, Beijing - China. Published in LNCS by Springer Berlin / Heidelberg, vol. 4556(7) / 2007, pp. 637-646, ISBN: 978-3-540-73282-2

SWRL (2004), Horrocks. I, P.F. Patel-Schneider, Boley. H, Tabet. S, Grosof. B, and Dean. M, “SWRL: A semantic web rule language combining owl and ruleml”, 2004,http://www.w3.org/submission/SWRL/., last visited November 2009

WAI-ARIA (2009), Web Accessibility Initiative-Accessible Rich Internet Applications, W3C Working Draft, February 2009, http://www.w3.org/TR/wai-aria/, last visited November 2009

Watt (2007), Stephen Waat, The evolution of the SOA, Part 2: Situational applications and the mashup ecosystem, IBM developerWorks, http://www.ibm.com/developerworks/webservices/library/ws-soa-mashups2/ , last visited November 2009

Web Services (2009), W3C Web Service Activity http://www.w3.org/2002/ws/, last visited November 2009

WS-BPEL (2009), Business Process Execution Language for Web Services version 1.1, http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel/ , last visited November 2009

WSMF (2002), D. Fensel and C. Bussler: The Web Service Modeling Framework WSMF, Electronic Commerce Research and Applications, 1(2), 2002.

WSMO (2009), Web Service Modeling Ontology, http://www.wsmo.org/index.html, last visited November 2009

XForms (2009), XForms 1.1, W3C Recommendation, October 2009, http://www.w3.org/TR/xforms11/, last visited November 2009

XSWT (2009), SWT is an XML-based GUI-description language for SWT, http://sourceforge.net/projects/xswt/, last visirted November 2009

Yahoo Maps (2009), Yahoo Maps homepage, http://maps.yahoo.com, last visited November 2009

Yahoo Pipes (2009), Yahoo Pipes homepage, http://pipes.yahoo.com, last visited November 2009

Forrester (2008), Young, G., Daley, E., Gualtieri, M., Lo, H., Ashour, M.: The Mashup Opportunity.

Zarnekow (2006), Zarnekow, R., Brenner, W., Pilgram, U.: Integrated Information Management. Applying Successful Industrial Concepts in IT. Springer, Berlin, 2006

Curriculum Vitae

Personal Information

Name	Amin Anjomshoaa
Address	Kammelweg 10/212, A-1210 Vienna, Austria
E-mail addresses	anjomshoaa@ifs.tuwien.ac.at
	andjomshoaa@gmail.com
Homepage	http://www.amininfo.net
Date of birth	24.08.1973

Education

2007 – Present	Doctoral study at the Institute of Software Technology and Interactive Systems, Information & Software Engineering Group, Vienna University of Technology, Austria
2007 – Present	Doctoral study at the Faculty of Architecture, department of Building Physics and Building Ecology, Vienna University of Technology, Austria
2006 – 2007	M.Sc. in Information Management, Vienna University of Technology, Austria
2003 – 2005	M.Sc. in Software Engineering & Internet Computing, Vienna University of Technology, Austria
2003	B.Sc. in Data Engineering & Statistics (Nostrifikation), Vienna University of Technology, Austria
1991 – 1997	B.Sc. In Applied Mathematics and Computer Science, Kerman University, Iran

Work Experience

2003 – Present	Institute of Software Technology & Interactive Systems, Vienna University of Technology; Project Assistant / Faculty
2008 – Present	Secure Business Austria, IT Competence Center; Project Manager / Programmer
2002 – 2003	ISOLAB EDV-GMBH, Vienna, Austria; Chief Technology Officer
2000 – 2001	SWP Office, Tehran, Iran; Project Manager (joint Austrian-Iranian research project)
1999 – 2000	Nemood Inc., Kerman, Iran; Network Manager / Software Manager
1994 – 1999	Kerman University, Kerman, Iran; System Administrator / Programmer / Instructor

Publications

Amin Anjomshoaa, Vo Sao Khue, A Min Tjoa,Edgar Weippl, Michael Hollauf, Context Oriented Analysis of Web 2.0 Social Network Contents MindMeister Use-case, Accepted for presentation in The 2nd Asian Conference on Intelligent Information and Database Systems, 24-26 March 2010, Hue City, Vietnam, Springer LNCS/LNAI

Amin Anjomshoaa, Andreas Hubmer, A Min Tjoa, Combining and Integrating Advanced IT-Concepts with Semantic Web Technology - Mashups Architecture Case Study, Invited paper in The 2nd Asian Conference on Intelligent Information and Database Systems, 24-26 March 2010, Hue City, Vietnam, Springer LNCS/LNAI

Mansoor Ahmed, Amin Anjomshoaa, Muhammad Asfandeyar, A. Min Tjoa, Towards an Ontology-Based Solution for Managing License Agreement Using Semantic Desktop, Accepted for presentation in The Fifth International Conference on Availability, Reliability and Security (ARES 2010), February, 15th – 18th 2010, Krakow, Poland

Amin Anjomshoaa, Gerald Bader, and Amin Tjoa, Exploiting Mashup Architecture in Business Use Cases, NBIS 2009, Indianapolis USA

Asfandeyar, M. A.; Anjomshoaa, A.; Weippl, E. R. & Tjoa, A. M. (2009), Blending the Sketched Use Case Scenario with License Agreements Using Semantics., in Dimitris Karagiannis & Zhi Jin, ed., 'KSEM' , Springer, , pp. 275-284 .

T. Moser, K. Schimper, R. Mordinyi, A. Anjomshoaa:SAMOA - A Semi-automated Ontology Alignment Method for Systems Integration in Safety-critical Environments; 2nd IEEE Intl. Wsh. on Ontology, Fukuoka, Japan; 16.03.2009 - 19.03.2009; in:"2nd IEEE Intl. Wsh. on Ontology", (2009), ISBN: 978-0-7695-3575-3; S. 724 - 729.

Shayeganfar, F., Anjomshoaa, A. (2009). Exploitation of Semantic Building Model in Indoor Navigation Systems, EGU 2009, to be presented in European Geosciences Union General Assembly, April 2009.

Shayeganfar, F., Mahdavi, A., Suter, G., Anjomshoaa, A. (2008). Implementation of an IFD library using Semantic Web technologies: A case study, ECPPM 2008 eWork and eBusiness in Architecture, Engineering and Construction, pp. 539 – 544.

M. Ahmed,A. Anjomshoaa, A. Tjoa:Context-Based Privacy Management of Personal Information Using Semantic Desktop: SemanticLIFE Case Study; iiWAS 2008, Linz; 24.11.2008 - 26.11.2008; in:"Proceedings of the 10th International Conference on Informationb Integration and Web-based Application&Services", Oesterreichische Computer Geselschaft, Band 241 (2008), ISBN: 978-1-60558-349-5; S. 214 - 221.

Amin Anjomshoaa, A Min Tjoa: Integration of Semantic XForms and Personal Web Services as a Tool to Bridge the Gap between Personal Desktops and Global Business Processes, UNISCON 2008

Amin Anjomshoaa, A Min Tjoa: Improving Web Form Accessibility using Semantic XForms for People with Cognitive Impairments, Computers Helping People with Special Needs (ICCHP 2008), Springer

Shayeganfar, F., Anjomshoaa, A., Tjoa, A. (2008). A Smart Indoor Navigation Solution based on Building Information Model and Google Android, Computers Helping People with Special Needs (ICCHP 2008), Springer, pp. 1050 – 1056.

M. Ahmed, A. Anjomshoaa, M. Nguyen, A. Tjoa: Towards an Ontology-based Organizational Risk Assessment in Collaborative Environments using the SemanticLIFE, ARES 2007

H. Hoang, M. Nguyen, A. Tjoa, A. Anjomshoaa: "A Front-End Approach for User Query Generation and Information Retrieval in the SemanticLIFE Framework"; iiWAS2006 - The 8th International Conference on Information Integration and Web-based Applications and Services, Yogyjakarta, Indonesia; 04.12.2006 - 06.12.2006; in: "Proceedings of the 8th International Conference

on Information Integration and Web-based Applications and Services", Austrian Computer Society, (2006), S. 107 - 116.

H. Hoang, A. Tjoa, A. Anjomshoaa: Towards a New Approach for Information Retrieval in the SemanticLIFE Digital Memory Framework;WI2006 - The 2006 IEEE/WIC/ACM International Conference on Web Intelligence, Hong Kong; 18.12.2006 - 22.12.2006; in: "Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence", IEEE Computer Society Press, (2006).

H. Hoang, A. Tjoa, A. Anjomshoaa:VQS - An Ontology-based Query System for the SemanticLIFE Digital Memory Project; SWWS06 - The 2nd IFIP WG 2.12 & 12.4 International Workshop on Web Semantics held conjunction wioth OTM06, Montpellier, France; 01.11.2006 - 03.11.2006; in: "Proceedings of the 2nd IFIP WG 2.12 & 12.4 International Workshop on Web Semantics", Springer, LNCS 4278 (2006), ISBN-978-3-540-48273-4; S. 1796 - 1805.

K. Mustofa, A. Tjoa, A. Anjomshoaa: "Semantic Enrichment of Search Result: the Coupling of Semantic Store and Google Services"; iiWAS2006 - The 8th International Conference on Information Integration and Web-based Applications and Services, Yogyakarta, Indonesia; 04.12.2006 - 06.12.2006; in: "International Conference on Information Integration and Web Based Applications & Services", (2006).

M. Nguyen, A. Tjoa, A. Anjomshoaa, F. Shayeganfar: Utilising Web Service Based Business Processes Automation by Semantic Personal Information Management Systems - The SemanticLife Case; 6th International Conference Practical Aspects of Knowledge Management (PAKM2006), Vienna, Austria; 30.11.2006 - 01.12.2006; in: "6th International Conference Practical Aspects of Knowledge Management (PAKM2006)", Springer LNCS, (2006), ISBN 3-540-49998-9; S. 1 - 10.

A. Tjoa, A. Anjomshoaa, S. Karim: Exploiting SenseCam for Helping the Blind in Business Negotiations; ICCHP 2006, Linz; 07/2006; in: "Computers Helping People with Special Needs", Springer, LNCS 4061 (2006), ISBN 978-3-540-36020-9; S. 1147 - 1154.

A. Tjoa, A. Anjomshoaa, S. Karim, F. Shayeganfar: Exploitation of Semantic Web Technology in ERP Systems; Confenis 2006, Wien; 24.04.2006 - 26.04.2006; in: "Research and practical issues of enterprise infomation systems", A. Tjoa (Hrg.); Springer, (2006), ISBN 0-387-34345-8; S. 417 - 427.

A. Tjoa, A. Anjomshoaa, M. Nguyen, F. Shayeganfar: Using Semantic Personal Information Management Systems - The Semantic Life Case; 6th International Conference Practical Aspects of Knowledge Management (PAKM2006), Wien; 11/2006 - 12/2006; in: "Practical Aspects of Knowledge Management", Springer, LNAI 4333 (2006), ISBN 3-540-49998-9; S. 1 - 12.

A. Tjoa, R. Wagner, A. Anjomshoaa, F. Shayeganfar: Semantic Web: Challenges and New Requirements; DEXA Workshops, Copenhagen, Denmark; 22.08.2005 - 26.08.2005, IEEE Computer Society Press, (2005), S. 1160 - 1163.

Amin Anjomshoaa, ICT Benchmarking Tool Product Report, United Nations Conference on Trade and Development, 2004

Amin Anjomshoaa, A. Schatten, A. Tjoa, H. Shafazand, The Application of Software Agent Technology to Project Management Infrastructure; in: "Proceedings of the International Conference on Information Integration and Web-based Applications and Services", SCS-Publishing House, 2002, (invited), ISBN 3-936150-18-4, S. 1 - 4.

Amin Anjomshoaa Knowledge Representation Using ISODataDreamer; EuroAsia ICT 2002, Shiraz, Iran

Anjomshoaa, A. Schatten, A. Tjoa, H. Shafazand: Building an Web-Based Open Source Tool to Enhance Project Management, Monitoring and Collaboration in Scientific Projects; in: "Proceedings of the International Conference on Information Integration and Web-based Applications and Services" (iiWAS 2001) Linz , 2001, -