sButler: A Mediator between Organizations' Workflows and the Semantic Web

Cecile Aberg
Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping Sweden
cecab@ida.liu.se

Patrick Lambrix
Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping Sweden
patla@ida.liu.se

Juha Takkinen
Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping Sweden
juhta@ida.liu.se

Nahid Shahmehri
Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping Sweden
nahsh@ida.liu.se

ABSTRACT

The Semantic Web will enable business-to-business in new ways. In this context, organizations will need to incorporate the Semantic Web into their work routines. However, this should be done in a way that does not require major changes in the organization. To this aim we propose a model for integrating the usage of the Semantic Web into an organization's work routines. The central component of the model is an sButler, a software agent that mediates between the organization and the Semantic Web. Further, we describe an architecture for one of the important parts of the sButler, show its feasibility with an implemented prototype, and discuss a test using a travel scenario.

1. INTRODUCTION

Internet and the Web have provided an environment to do business-to-business in a virtual world where distance is less of an issue. Providers can advertise their products globally and consumers from all over the world obtain access to these products. However, the heterogeneous and continuously changing environment leads to several problems related to finding suitable providers that can satisfy a consumer's needs. The Semantic Web [5] aims to alleviate these problems. By allowing software agents to communicate and understand the information published on the Web, the Semantic Web enables new ways of doing business and consuming services. Semantic Web technology will provide an environment where the comparison of different business contracts will be made in a matter of minutes, new contractors may be discovered continuously and the organizations' routines may be automatically updated to reflect new cooperations. All this provides an *agility* in the implementation of business processes which will allow for new business visions such as the e-business models described by Timmer [26] or the Real-Time Infrastructure (RTI) envisioned by Gartner's business analysts [14].

The Semantic Web provides an infrastructure for software agent to software agent communication. However, the organization does not necessarily use this infrastructure to communicate internally. The organization does not "talk Semantic Web". Therefore, to be able to gain new advantages by using Semantic Web technology, organizations should not need to change their routines, but it should be possible to integrate the new technology in the existing routines [13]. As a result, the usage of the Semantic Web requires the definition of a proper model of interaction between the internal world of the organization and the external world of the Semantic Web. In this paper we provide for such a model where a software agent (sButler[1]) mediates between an organization and the Semantic Web. We show the knowledge requirements and usage for such a model. Further, we propose an architecture for one of the most important capabilities of the sButler. This capability deals with the generation of process instances for organizations' tasks based on Web services. We performed a feasibility study in the form of a prototype implementation and tested the prototype with a conference travel scenario.

In the next section we present our model. Section 3 describes our proposal for an architecture for the sButler with focus on support for the generation of process instances. The prototype and the test are described in sections 4 and 5. In section 6 we discuss related work and we conclude with section 7.

2. INTEGRATION MODEL

To define an overall model for integrating the Semantic Web into the work routines of an organization, we must define a model for the organization's work routines, a model for the Semantic Web and the actual integration. We adopt Workflows as a model for organizations' work routines, and

[1]sButler stands for Semantic Web Butler.

WWW2005, May 10–14, 2005, Chiba, Japan.
.

Semantic Web services as a model for the Semantic Web. The integration is achieved using an sButler agent that mediates the interactions between the Workflows and the Semantic Web.

We choose Workflows because it is a well established method to describe organizations' routines. To represent Workflows we consider the requirements of the Workflow Management Coalition (WfMC) [30], which establishes standards of Workflow usage and management. In the Workflow Reference Model [16] the WfMC specifies that a *Workflow* (or *Business Process*) is composed of *tasks* (or *activities*), that are performed by *Workflow participants* that may be grouped in *organizational roles*. An organizational role defines the set of capabilities that each of its participants must have. A Workflow is defined, created and its execution managed by a *Workflow Management System* (WfMS) through the use of software, running on one or more Workflow engines, which is able to interpret the process definition, interact with Workflow participants and, where required, invoke the use of IT tools and applications [31]. Once a Workflow is defined, it may be executed. This is done by the *Workflow enactment service* which generates a *process instance* and controls its enactment. A process instance is the representation of a single enactment of a process. As such, a process instance uses its own input and output data. Operations performed during the process instance enactment may be executed by external applications, i.e. applications which are not part of the WfMS. As a result, communication between the WfMS and these applications takes place. The WfMC defines an *invoked application interface* as a standard format for this communication. The interface specifies which information may be passed on to the *invoked applications*, such as the attributes of the tasks that should be executed.

To adopt a representation of the Semantic Web we consider the needs of the organizations. As mentioned before, the organizations need to use the Semantic Web to find and access distant contractors in a dynamic manner. The work of the W3C's Semantic Web Service Interest Group [27], which looks at means of integrating the Semantic Web technology in the Web service architecture, provides for such a need. The group strives to describe the Semantic Web as a source of distant contractors modeled as so called Semantic Web services. Web services are programmatic interfaces for applications and enable application-to-application communication on the World Wide Web [28]. They describe the services that are available on the Web. The current consensus is that these descriptions specify the data format required for calling the actual service (i.e. a port and a set of inputs and outputs described with WSDL [29]). Semantic Web services are Web services that provide descriptions based on the semantics of the services. The descriptions are represented in a Semantic Web language. These languages are machine understandable and enable automated discovery of the services. In the remainder of the paper we use the term Web service for Semantic Web service. We also assume in our model that communication in the Semantic Web is achieved through the use of a Semantic Web language, typically a flavor of OWL [3].

In our model, the integration between an organization's routines and the Semantic Web is performed by a dedicated sButler agent. The sButler agent mediates between the organization's WfMS and the Semantic Web service architecture. Therefore, the sButler has to handle such things as the process instance generation of the parts of the Workflow whose enactment is delegated to external contractors, the enactment of these process instances and the management of the experience accumulated from the usage of the Semantic Web (e.g. association of trust values to service providers). Towards the WfMS the sButler is an invoked application that is called by the WfMS to generate and manage process instances composed of Web services for specific tasks. These tasks are known internally to the organization as tasks performed by external contractors. As a result, in the Workflow representation, they are associated to the `external contractor` organizational role.[2] The sButler further communicates with the WfMS through the invoked application interface. Communication with the Semantic Web is achieved by submitting queries for Web services to the network. Depending on the concrete architecture of the network (i.e. peer-to-peer or client-server), the query is sent to a specific registry node or broadcasted on the network to discover the nodes which provide Web services that match the query.

In its role of mediator the sButler supports the retrieval of Web services that match the organization's need. Therefore, the sButler needs to ensure that the discovery algorithms receive optimal input information. For this purpose, the sButler must define an internal representation of the tasks which strives to capture all the aspects of the task that are important for matching with Web service representations. These aspects may include such things as the kind of transaction the task requires and the kind of result the task is expected to produce. As a result, the sButler performs a two-step translation from the organization's representation of a task to its own internal representation and from there to the Web service representations. As Web services may be represented using different languages, different translation schemes may be needed.

Figure 1 illustrates our model. In the figure an organization's routine is represented by a Workflow of five tasks managed by a WfMS. One of these tasks (`task3`) must be delegated to external contractors. As a consequence, when a process instance for this task must be generated, the generation is handed over to the sButler (step 1 in the figure). The sButler then translates the task description into some query for Web services, submits it to the Semantic Web (step 2), and uses the answer to generate a process instance for the task. The process instance is then returned to the WfMS which uses it according to its management scheme (step 3). Later, the enactment and management of the process instance will also require delegation to the sButler. The sButler will again interact with the Semantic Web and return the results to the WfMS.

3. SBUTLER ARCHITECTURE

The sButler may be seen as the Semantic Web interface of the organization. In practice, the sButler may be either implemented as an application that is installed and running on the organization's platform, or as a Web service that is called by the organization when in need of a mediator to the Semantic Web.

In this section we describe an architecture for an sButler

[2]As we focus on the integration of the Semantic Web into an organization's Workflow, we assume in this paper that external contractors are always related to Web services.

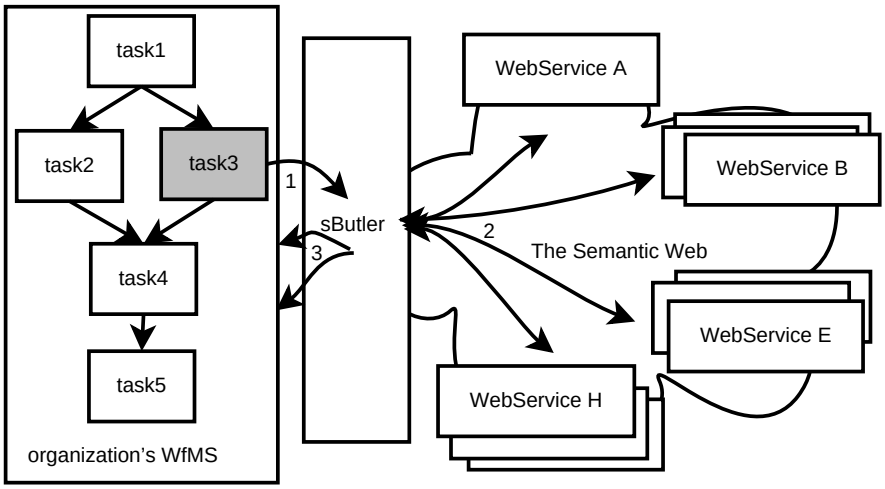

Figure 1: Model to integrate the Semantic Web in an organization's routines.

agent. We focus on one of the most important capabilities of the sButler, namely supporting the generation of process instances for a given Workflow task. This includes the generation of queries towards the Semantic Web based on the given Workflow task, the actual querying to find relevant (compositions of) Web services, the generation of process instances that satisfy the requirements of the Workflow task, the ranking of the process instances with respect to user and organization preferences, and the integration of the process instances into the organization's Workflow.

To generate a process instance for a given task, the sButler requires as input a task description. Further, the sButler uses different kinds of knowledge. Some of this knowledge is represented in ontologies shared by the organization and distant contractors. This allows for better interoperability between their systems. Other knowledge may be organization specific and stored locally in knowledge bases. We identified the following kinds of useful knowledge.

- The `Domain knowledge` provides the vocabulary used by the organization to describe its work and routines. Typically, this domain knowledge describes the different input and output types of the Workflow tasks. In practice, this knowledge is seldomly built from scratch but existing ontologies are used or extended. Examples are ontologies about products, documents and activities.

- The `Transaction knowledge` provides the vocabulary used to describe the different Web transactions that may be established between organizations and Web service providers. This knowledge is represented in shared ontologies. For instance, the MIT Process Handbook defines transactions such as the `Acquire` activity and its specializations [19].

- The `Process knowledge` provides the vocabulary to describe constraints on Web service processes. The MIT Process Handbook [19] is a source of information for such knowledge. Also the quality of service definitions provided by the WfMC in [32] and the METEOR project in [7] are specific cases of process knowledge.

- The `Decomposition knowledge` defines the concepts that may be used to decompose queries and provides explicit rules for the decomposition and reassembly. Some of this knowledge may already be represented more or less explicitly in the `Domain knowledge`. Further, additional decomposition rules (e.g. based on decomposition heuristics) may be included.

- The `User and organization preference knowledge` specifies preferences of the users and organizations. This knowledge is typically expressed in terms of the `Domain`, `Transaction` and `Process knowledge`.

- The `Query knowledge` provides experience data from previous generation and management of queries for Web services. This knowledge typically refers to specific Workflow tasks.

The sButler aims to represent all the aspects of the task that are important for matching with Web service representations. Our internal task model requires the following aspects. First, the required result needs to be represented. This may be, for instance, a concrete object, an action, or information about an object or an action. We use `Domain Knowledge` to represent this information. Further, we represent the kind of process that is performed to create the desired result. This is described using the `Process knowledge`. Finally, another important aspect is the type of transaction the task requires. This may include such things as the delivery mode and the kinds of ownerships that are transferred (e.g. renting vs. buying). This is described using the `Transaction knowledge`. These aspects focus on the organization's requirements on the task. There may be other aspects that are relevant and focus on other areas such as the location of the Web services or technical requirements for the communication with the Semantic Web. These are not considered in this paper.

Figure 2 shows the modules of an sButler architecture for process instance generation. We now describe each module and its use of the different knowledge sources.

Query Generation is the process that supports the generation of a query for Web services and is respon-

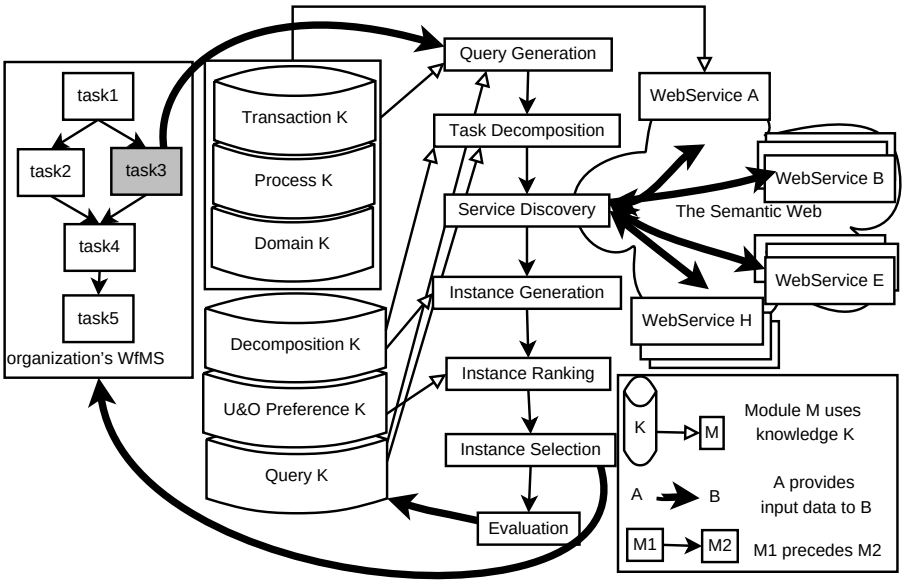

Figure 2: The sButler's modules for process instance generation.

sible for the two-step translation performed by the sButler. During the first translation step the sButler uses the task description provided by the organization to create an internal representation of the task. It uses the organization's `Domain knowledge` as well as `Transaction`, `Process knowledge`, and `Query knowledge`. The sButler may also interact with the organization to obtain as much relevant information as possible. The second step translates the internal representation into Web service representations as required by the existing discovery algorithms. The result is a query for Web services.

Task Decomposition is the process of locally identifying possible decompositions of the task. One possible approach consists of decomposing the desired object and building a new query for each part of the object. In this case some `Decomposition Knowledge` is required. This module also uses `Query knowledge`.

Service Discovery is the actual querying for Web services. Queries for services that perform part of the task (in the case of decomposable tasks) or the whole task are submitted to other service discovery agents available on the Semantic Web. Such agents typically provide for their own Web service composition approach as described in [22] and [10]. The result is a set of candidate Web services for each query submitted.

Instance Generation is the process of putting the parts together to build process instances that produce the whole desired object. It is done using the `Decomposition Knowledge`. The result is a set of candidate process instances.

Instance Ranking consists of ranking the process instances according to some criteria. These criteria are typically represented in a `User and organization preference` knowledge base.

Instance Selection may be done in two modes. In automatic mode, one of the highest ranked instances is automatically selected as the process instance for the task. In manual mode, the user chooses a process instance based on an ordered list of process instance descriptions presented by the sButler. In both cases the sButler returns the process instance to the WfMS which uses it according to its management scheme. Typically, when the time comes to execute the process instance, the WfMS will delegate the enactment to the sButler.

Evaluation aims at improving the performance of the process instance generation by using previous experience. For instance, many process instances may be generated for the same task and information about the selection can be stored. This experience may be used the next time the user has a similar information need. Also during service discovery knowledge may be gained, such as some Web service registries may systematically take too long to answer. All this accumulated knowledge is stored in the `Query knowledge`.

The next section describes our prototype solution for the sButler's process instance generation.

4. IMPLEMENTED PROTOTYPE

Our prototype follows the model and architecture as described in sections 2 and 3. It is implemented in Java, and the knowledge bases are OWL-lite files integrated in an OWLJessKB [15] knowledge base. Since OWLJessKB allows for Horn Rules reasoning using Jess [24], additional rules (e.g. to formulate user and organization preferences) are directly written as Jess triplets. Each Java implemented sButler module may query the Jess knowledge base using the Jess and OWLJessKB Java APIs. Figure 3 illustrates the usage of the different knowledge sources by the sButler prototype's process instance generation modules.

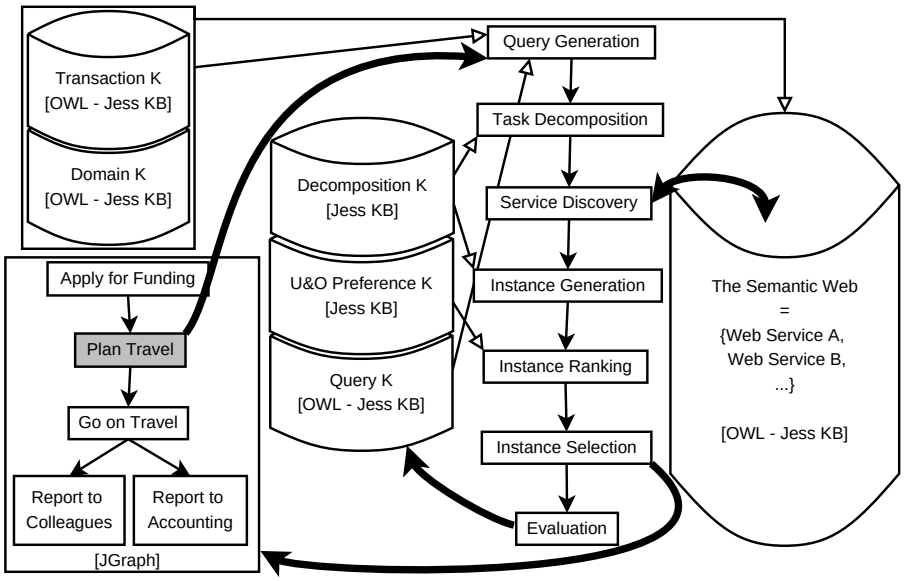

Figure 3: Knowledge sources for the process instance generation.

We simulate the organization's WfMS with JGraph [21], a Java graph visualization library which provides a graphical visualization of the Workflow as a task flow. It also provides an annotation mechanism for the tasks delegated to the sButler that is used to assign task descriptions and attributes. The Semantic Web is represented by a set of Web services. The Web services and sButler's internal representation of the task share the same format composed of two of the aspects mentioned in section 3: the desired result and the kind of transaction the organization wants to enter into. The Web services and Web service queries are all represented by OWL-lite files integrated in the OWLJessKB knowledge base. The service discovery is simulated with the matchmaking algorithm provided by OWLJessKB.

Considering our format for Web service descriptions, we observe that some local knowledge, such as user and organization preferences, may provide information for decomposing the desired result. Therefore, we designed a simple algorithm for using local knowledge to compose services. First, we identify the local knowledge (i.e. `Domain knowledge` or `User and organization preference knowledge`) related to the required result that can be used for decomposition. Secondly, we use the `Decomposition knowledge` to perform the decomposition. Then, we generate a set of queries for each of the subparts of the required result and run the queries. Finally, we use the `Decomposition knowledge` to combine the answers and produce solutions for the task. The first three steps are performed during the `Task Decomposition`, the fourth step is performed during the `Service Discovery` and the last step is performed during the `Instance Generation`.

5. TEST SCENARIO AND DISCUSSION

For testing our prototype we decided to use a conference travel scenario. Travel scenarios have been used by others in the field (e.g. [20, 9, 4]). These scenarios are interesting for several reasons. First, different users have different preferences regarding traveling and therefore the use of user and organization preference knowledge is highlighted. Also, the scenario shows the need for a flexible process instance generation process. Even for the same task for the same user the circumstances and constraints may differ from one time to the next. Further, planning a travel usually requires the use of several Web services and illustrates the need for the management of multiple Web services. It is also easy to modify the scenario to test different capabilities of service discovery algorithms. Finally, it is a scenario for which many people have an understanding of the domain.

Our scenario is as follows. The organization is a university where one category of participants belongs to the research staff and part of the organization Workflow describes the activities of the staff members such as conference travel. This Workflow specifies that when a researcher of this university goes to a conference, she performs a number of tasks. First she applies for funding for travel. This is done by filling in administrative forms and handing them over to the accounting department several weeks before the travel. The second step is to plan the travel. This includes such things as booking transportation to the conference location. Moreover, the travel itinerary produced provides the process instance for the next task as illustrated in figure 4. The third step is to actually travel to the conference and participate. Finally, the researcher must report on the scientific content of the conference to her colleagues and handle administration regarding travel costs for the accounting department. These last two tasks may be performed in parallel. As external contractors are typically required to plan the travel, the organization role for this task is set to `external contractor`. Figure 4 provides the task flow view of the Workflow for conference travel.

The knowledge bases contain concept descriptions as follows.

- The `Transaction knowledge` models the `Acquire` activity specialization hierarchy from the MIT Process Handbook. This includes the activity `Buy` that may be

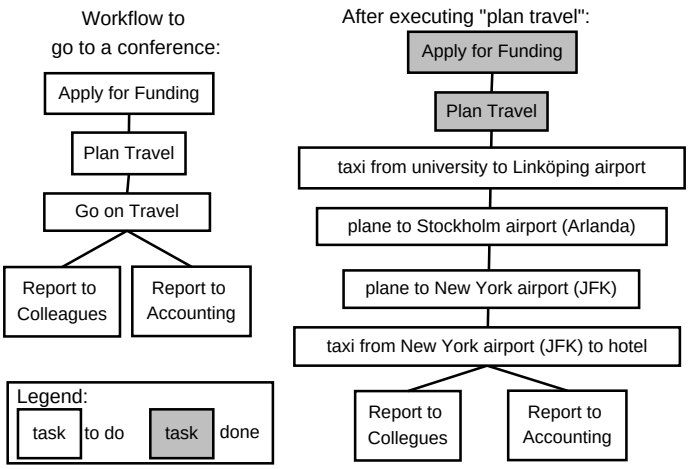

Figure 4: Workflow for the test scenario.

decomposed into the sequence of activities `Identify potential sources`, `Identify own needs`, `Select supplier`, `Place order`, `Receive`, `Pay`, and `Manage suppliers`.

- The `Domain knowledge` models the `OfficeSupply` and `Travel` concepts. The `Travel` concept may be specialized into `RoundTripTravel` or `OneWay- Travel`. A `Travel` is composed of `ItineraryLeg`s which have the properties `hasDeparture`, `hasArrival` and `hasTransportMode`. The domains of the `hasDeparture` and `hasArrival` properties are locations, while the domain of `hasTransportMode` is `TransportMode`. Examples of `TransportMode` are train, boat, plane, and taxi.

- The `Decomposition knowledge` includes information related to the decomposition and assembly of itineraries. An example of a decomposition rule for an `ItineraryLeg` is that an itinerary leg from Linköping University to X using unspecified transportation can be decomposed into two itinerary legs where the first one goes from Linköping University by taxi to Linköping airport and the second leg goes from Linköping airport to X using unspecified transportation. An example of a composition rule is a rule that combines two travels where the final destination of the first travel is the place of departure of the second travel.

- The `User and organization preference knowledge` provides preferences about travels. For instance, we have the preference that if the place of departure is Linköping University and the place of arrival is not in Scandinavia then the travel's first leg should be a taxi ride from Linköping University to Linköping airport.

- The `Query knowledge` contains the definition of the task according to the sButler's internal representation format. In this case we assume that the Workflow has already been used before and that the `Evaluation` module has stored the information that the transaction performs a `Buy` activity and the desired result is a `OneWayTravel`[3].

Further, in our prototype, the `Semantic Web` is represented as a set of six travel agencies providing nine different `ItineraryLeg`s.

To start the application one executes the corresponding Java class (window A in figure 5). The OWL-lite files are loaded into the Jess knowledge bases and the graphical representation of the organization's Workflow appears on the screen (window B in figure 5). In the Workflow interface, the tasks that the sButler supports are highlighted. When the user right-clicks on the `Plan Travel` task, she is presented with a menu and she may choose the *Visualize* option. This provides information on aspects that are associated to the sButler's internal representation of the task. The ontology hierarchies of the `Transaction` and `Domain knowledge` are displayed and the concept that corresponds to the aspects of this task is highlighted. The alternative to the visualization is to start the process instance generation. This is done by selecting *Provide input* in the menu (window C in figure 5). The sButler is invoked and starts performing the process instance generation for this task as follows.

Query Generation Although the aspects of the task are already represented internally (see above), the query generation process still needs to acquire the data that is specific to the process instance. This data is added to the sButler's task description. The type of required data is computed from the specified aspects. The data required for a `OneWayTravel` are the places of departure and arrival. A window is displayed for the user for providing this information (window D in figure 5). The user may enter Linköping University for departure and New York University for arrival. As the prototype assumes that the sButler task representations and the Web service representations are OWL descriptions, the sButler's task descriptions can be used as queries for Web services. Once the query is formulated, the user may proceed by selecting *Execute* in the menu (window C in figure 5).

Task Decomposition tries to decompose the desired result. The sButler identifies the relevant decomposi-

[3]For the sake of brevity we use a one way travel as example instead of a round trip.

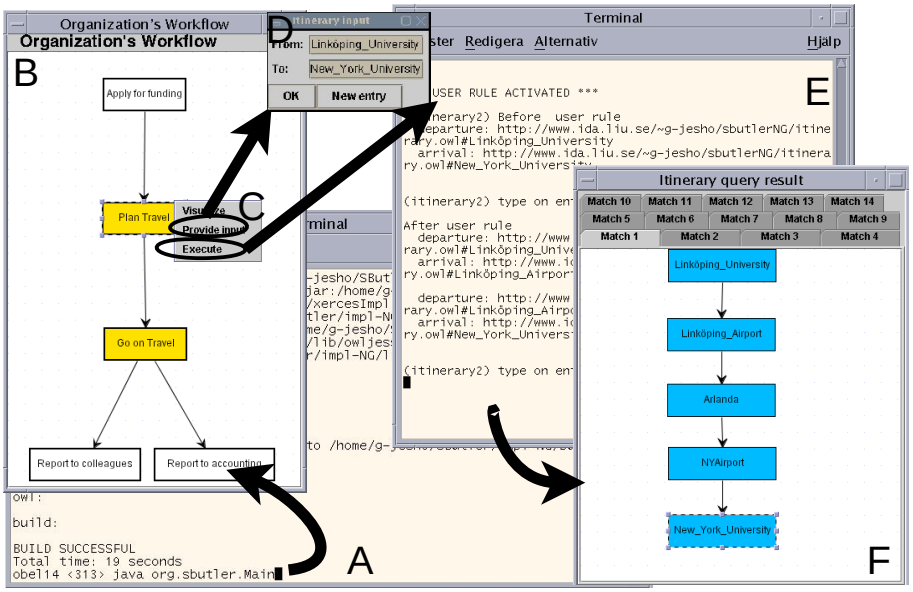

Figure 5: Prototype Screenshots.

tion rules in the `Decomposition knowledge` and applies them (window E in figure 5). The result of the decomposition of the `OneWayTravel` from Linköping University to New York University provides the following `Travel` parts:

1. A taxi leg from Linköping University to Linköping airport
2. A set of legs from Linköping airport to New York University.

Service Discovery submits the queries for the different travel parts to the Jess knowledge base. The answer provides a set of candidate services for the travel parts.

Instance Generation puts the answers to the queries for travel parts together in order to provide complete solutions. The solutions for queries 1 and 2 are combined to provide itineraries for the whole travel. Each itinerary corresponds to a different composition of services.

Instance Selection The user is presented with the different itineraries and selects one. Each itinerary is represented by a graph where the cell is a location and the arrow represents a travel leg (window F in figure 5). Clicking on the leg provides specific information about the leg such as the name of the travel agency and the means of transportation. After the instance selection the chosen itinerary can be used as the sub-Workflow specification for the `Go on Travel` task. For example, the WfMS can expand the Workflow of the `Go on Travel` task as illustrated in figure 4.

The knowledge bases are important components of our architecture. The fact that we used Jess knowledge bases allowed us to do consistency checking as well as verification of the knowledge. We based the domain knowledge on existing ontologies about travel, locations and time. Therefore, the generation of the knowledge was not very time consuming for this application. For larger applications, however, there may be a need to use information from different ontologies and tools for merging and aligning ontologies (e.g. [18]) may have to be used. Another advantage of using ontologies is that the information can be more easily shared between applications.

The decomposition knowledge was manually created based on our knowledge about the domain. The most useful information in the domain knowledge for this purpose was (often implicit) information about part-of relations together with their ranges and domains. It may be interesting to investigate whether parts of the decomposition knowledge may be semi-automatically generated from the domain knowledge. The user and organization preference knowledge contains static rules. These rules may be generated by learning approaches as in adaptive systems. The transaction knowledge is based on the MIT Process Handbook and can be considered as a shared ontology. We downloaded and parsed the information in the handbook and translated it to our internal format. This needs to be done only once and is reused for all tasks.

For our application the system loaded the knowledge bases at start-up. This took ca 30 seconds on a SUN Ultra 10 workstation with a 440-MHz UltraSPARC-IIi architecture. From then on all interaction and processing was instantaneous. Future work will need to investigate whether the same performance can be achieved in larger scenarios as well as when the Semantic Web is not simulated using knowledge bases. As a first step a multi-agent environment may be used as a controlled environment to test the system further.

6. RELATED WORK

There are not many systems that integrate the Semantic Web into an organization's routines yet. One system that aims to do this is METEOR-S [7]. This is an extension of

the METEOR project. METEOR provided a WfMS and dealt with different process management issues such as task dependencies, CORBA-based and Web-based enactment of processes, and quality of service. METEOR-S builds on this expertise to integrate the Semantic Web into the management of Workflow processes and aims to provide a platform that supports the complete life cycle of Semantic Web processes. The life cycle includes the steps semantic annotation and publication of Web services, abstract process creation, semantic discovery of Web services including the integration of Web services into Workflow processes, and orchestration/composition of Web services. One important difference between METEOR-S and the sButler approach is that METEOR-S focuses on service discovery while the sButler aims to mediate between the organization and the Semantic Web.

Further, there are approaches focusing on Web service composition such as SWORD [22] and Racing [10]. Our vision for sButler is to integrate these different approaches to service discovery by translating between the internal representation of the organization and the formats required by the different service discovery approaches.

Another kind of related work are approaches that wrap the organization's services as Web services either by providing a support system for composition design like E-Flow [8], or by providing description languages for composition like BPEL [2], OWL-S [25], and the WSMO standard [23], or by providing a full fledged system to support the development and management of interorganizational Workflows with multi-agent systems (e.g. [6]). These approaches aim to represent complex business processes as Web service composition and deal therefore mainly with the provider side. The current approaches do not deal with the translation between the organization's internal representations and the Semantic Web, but assume that the organization's systems that are connected to the Semantic Web are Semantic Web (or Web service) compliant. These approaches provide the descriptive technology for service composition that the the sButler has to understand.

Finally, the Web Service Management Framework (WSMF) [11] and the sButler have a similar vision when it comes to the need for a strong mediation service that allows anybody to speak to everybody. However, the WSMF focuses on the communications between Web services while our approach focuses on the communication between the organization and the external Web services. Further, the Web Service Modeling Ontology standard (WSMO - Standard) [23] strives to refine the WSMF by defining an ontology which describes the four elements of the WSMF: ontologies, goals, Web services and mediators.

7. CONCLUSION

In this paper we defined a model for integrating the Semantic Web in terms of Web services into an organization's Workflow using sButler agents as mediators. We also described an architecture for the main component of an sButler agent which handles the generation of process instances. Several knowledge bases are used in this process. We performed a feasibility study in the form of a prototype implementation and tested the prototype with a conference travel scenario.

In [1] we introduce an agent-based framework for integrating Web services into an organization's Workflow. This framework allows for publication of Web services, process generation, discovery and composition of Web services, as well as for the integration of Web services into Workflow processes. Within this framework sButlers are agents that mediate between the organization's WfMS and the agents in the Semantic Web. The implementation described in this paper shows the feasibility of the knowledge management and reasoning part of the agent-based model. We now plan a full-scale implementation of our approach. We will use the Jade [17] multi-agent platform as a base and aim to evaluate our approach in agentcities.

Another direction will investigate the interaction of the sButler agents in the context of different Web service formats such as OWL-S [25] and the FIPA service format [12].

Acknowledgements

We thank Johan Aberg for discussions and valuable comments on previous drafts of this paper and Jesper Holmqvist for his help in implementing the prototype. We acknowledge the support of Vinnova (the Swedish Agency for Innovation Systems) and the EU Network of Excellence REWERSE (Sixth Framework Programme project 506779).

8. REFERENCES

[1] C. Aberg, P. Lambrix, and N. Shahmehri. An Agent-based Framework for Integrating Workflows and Web Services. In *IEEE WETICE workshop on Agent-based Computing for Enterprise Collaboration*, 2005.

[2] BEA Systems, IBM, Microsoft, SAP AG, and Siebel Systems. Business Process Execution Language for Web Services. http://ifr.sap.com/bpel4ws/. Accessed 2004-12-08.

[3] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuiness, P. Patel.Schneider, and L. Stein. OWL Web Ontology Language Reference. http://www.w3c.org/TR/owl-ref/. Accessed 2004-12-08.

[4] B. Benatallah, Q. Sheng, and M. Dumas. The Self-Serv environment for Web services composition. *Internet Computing, IEEE*, 7(1):40–48, Jan-Feb 2003.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[6] M. B. Black. Coordinating multiple agents for workflow-oriented process orchestration. *Journal of Information Systems and E-Business Management*, 1(4):387–404, 2003.

[7] J. Cardoso and A. Sheth. Semantic E-Workflow Composition. *Journal of Intelligent Information Systems*, 21(3):191–225, 2003.

[8] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan. eFlow: a Platform for Developing and Managing Composite e-Services. Technical report, Software Technology Laboratory HP Laboratories Palo Alto, 2000.

[9] S. Dalal, S. Temel, M. Little, M. Potts, and J. Webber. Coordinating business transactions on the Web. *Internet Computing, IEEE*, 7(1):30–39, Jan-Feb 2003.

[10] V. Ermolayev, N. Keberle, and S. Plaksin. Towards a Framework for Agent-Enabled Semantic Web Service Composition. *International Journal of Web Services Research,*, 1(3):63–87, July-Sept 2004.

[11] D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[12] Foundation for Intelligent Physical Agents. FIPA Agent Management Specification. Technical report, FIPA, 2001. http://www.fipa.org/specs/fipa00023/XC00023H.pdf.

[13] Gartner. Gartner Says A New Generation of Software Is Emerging for IT to Meet the Agility Demands of Business, Oct 2004.

[14] Gartner. Gartner Says an IT Infrastructure Needs to be Transformed into a Real-Time Infrastructure to Meet Business Requirements, Oct 2004.

[15] Geometric and D. U. Intelligent Computing Laboratory. Owljesskb: A semantic web reasoning tool. http://edge.cs.drexel.edu/assemblies/software/owljesskb/ (Accessed: 2005-03-22).

[16] D. Hollingsworth. The Workflow Reference Model. Technical report, Workflow Management Coalition, 1995.

[17] Jade. Java Agent Development Framework. Open Source.

[18] P. Lambrix and H. Tan. Merging DAML+OIL Ontologies. In J. Barzdins and A. Caplinskas, editors, *Databases and Information Systems (Selected papers from the Sixth International Baltic Conference on DB&IS'2004)*, pages 249–258. IOS Press, 2005.

[19] T. Malone, K. Crowston, and G. Herman, editors. *Organizing Business Knowledge: The MIT Process Handbook*. Cambridge, MA: MIT Press, 2003.

[20] S. McIlraith and T. C. Son. Adapting Golog for Programming the Semantic Web. In *Proceedings of the 5th International Symposium on Logical Formalization of Commonsense Reasoning*, 2001.

[21] Open source. JGraph: The Java Graph Visualization Library. http://www.jgraph.com. Accessed 2004-12-08.

[22] S. Ponnekanti and A. Fox. SWORD: A Developer Toolkit for Web Service Composition. In *11th international World Wide Web Conference*, 2002.

[23] D. Roman, H. Lausen, and U. Keller. D2v02. web service modeling ontology - standard (wsmo - standard). http://www.wsmo.org/2004/d2/v0.2/20040306/, Accessed 2005-03-31, 2004.

[24] Sandia National Laboratories. Jess: The rule engine for the Java platform. http://herzberg.ca.sandia.gov/jess/. Accessed 2004-12-08.

[25] The OWL Service Coalition. OWL-S: Semantic Markup for Web Services (OWL-S 1.0). http://www.daml.org/services/owl-s/1.0/owl-s.html. Accessed 2004-12-08.

[26] P. Timmer. Business Models for Electronic Markets. *Electronic Markets*, 8:3–8, 1998.

[27] W3C. Semantic Web Services Interest Group. http://www.w3c.org/2002/ws/swsig/. Accessed 2004-12-08.

[28] W3C. Web Services Activity. http://www.w3c.org/2002/ws. Accessed 2004-12-08.

[29] W3C. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. http://www.w3.org/TR/2004/WD-wsdl20-20040803/. Accessed 2004-12-08.

[30] WfMC. Workflow Management Coalition (WfMC). http://www.wfmc.org. Accessed 2004-12-08.

[31] WfMC. Terminology and Glossary. Technical report, Workflow Management Coalition, 1999.

[32] WfMC. Workflow Standard - Interoperability Wf-XML Binding. Technical report, Workflow Management Coalition, 2000.