Choreography of Web Services based on Natural Language Storybooks

Kurt Englmeier
LemonLabs GmbH
Ickstattstrasse 16
80469 Munich, Germany
+49 89 2020 8240
KurtEnglmeier@computer.org

Javier Pereira
Universidad Diego Portales, Escuela de Informática
Avenida Ejercito 441, Santiago, Chile
+56 2 6768135
javier.pereira@udp.cl

Josiane Mothe
Université Paul Sabatier, IRIT Lab.
118 Route de Narbonne
F-31062 Toulouse, France
+33 5 61 55 67 65
mothe@irit.fr

ABSTRACT

Universally available services, which communicate in a standardized way, can provide a new generation of middleware. Harnessing the advantages of this promising middleware technology, however, means to be capable to understand and to handle its design language which emerges from standards like SOAP, WSDL, BPEL, etc. These languages are necessary for finding, composing and orchestrating web services. If at all, only IT experts are familiar with these languages.

The key actors, the domain experts of business processes, however, are not IT experts, and thus do not become the main designers. WS-Talk is a research project that encourages the co-existence of Natural Language and Web service technology. It reinforces the role of domain experts in designing business processes without having to resort to their IT colleagues. In our approach business process experts write storybooks in their own language. Their instructions are matched with semantics that represent application logic that, in turn, supports the automatic composition of software components. The WS-Talk products currently support organizations in managing their own and individual information, i.e. to set up their own enterprise search engine.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Clustering, Information Filtering, Search Process.
H.3.5 [**Online Information Services**]: Web-based Services
I.2.7 [**Natural Language Processing**]: Language parsing and understanding, text analysis.

General Terms

Management, Design, Human Factors, Languages.

Keywords

Semantic Web standards, Enterprise Search Systems, Web Service Orchestration, Web Service Choreography, End-user programming

ICEC'06, August 14–16, 2006, Fredericton, Canada.

1. INTRODUCTION

Business processes supported by IT usually manifest themselves in applications comprising a number of software components. Web Services promise the possibility to develop new applications in a dynamic way. The universal availability of services, which communicate in a standardized way, opens new horizons for middleware technology. The standardized and universal middleware layer supported by Web service technology can reduce the integration headaches which are still prevailing in many companies.

Web Services can add a new chapter to the success story of Semantic Web Standards. Open standards like XML foster the universal exchange of information over the Internet. Web Service technology now adds universal interchangeability and thus universal availability of application logic. This availability raises the potential to develop more applications, or more facets of applications, for broader quantity and variety of business processes (business logic). However, this availability does not mean an automatic pathway to new horizons in designing IT-based business processes. Real applications emerge from a complex and dynamic composition of a number of software components or Web Services. A raising availability of application logic also raises the effort to integrate its components.

What organizations expect from Web Services is first of all a reduction of "integration headaches" [11]. A survey – recently published on WebServices.Org – shows that a majority of companies take up Web Service technology in order "to integrate disparate systems". A further motivation for take-up addresses "tangible benefits in terms of reuse, developer productivity, and cost savings" [11]. Web Services, once conceived to facilitate more seamless e-commerce transactions beyond the firewall, get a role that is clearly focused on internal integration. The motivation for take-up emerges from a quest for a standardized platform- and vendor-independent middleware-layer as well as for reusability of existing application logic. Web Services standards and Web Service orchestration languages enable an essential move towards this middleware-layer [5].

Using Web Services, an application is rather a coalition of standardized and almost ubiquitous software modules than the typical monolithic block as we know it since decades. This coalition can be composed as well as adapted on-the-fly. WSDL (Web Service Definition Language), SOAP (Simple Object

Access Protocol) and UDDI (Universal Description, Discovery and Integration) are proven standards to define coalitions of highly interoperable components and to propagate them in a distributed environment [1, 7]. To form dynamic coalitions, they need to be found, composed and orchestrated. [9]
However, there is still some doubt that Web service technology together with web service orchestration standards suffice to unleash the full potential of software components that enable rapid development of versatile and highly adaptable applications. In this paper we investigate how natural language can enhance Web service technology in order to bring the human expert of business processes and the IT expert closer together in the design of IT-based business processes.
Natural language (NL) can provide us with semantics to write storybooks in our language, i.e. to name, to compose, and orchestrate software modules in the way humans think about their everyday work. This rationale is the focus point of the WS-Talk project. The objective of this project is to develop an instrument that enables the experts in business processes rather than the IT expert to define business processes. And the expert uses just natural language for the definition or description of processes. WS-Talk's focus on natural language processing extends towards information retrieval applications or features, by nature. The vision of WS-Talk is to provide companies with natural language interfaces for the management of their own information and to tune their information retrieval systems.
The natural language descriptions are matched with semantic representations of software components and their incoming and outgoing parameters. These representations help to identify the relevant software components or services and to address correctly the information required by these services as well as the information produced by them. The matching process resorts to meta-information which is designed by the IT experts. In the vision of semantic Web services [4], our meta-data focus on generic descriptions of processes as well as on specific ones that fit the application area of the respective company or department. This paper thus focuses primarily on coexistence of natural language and semantic web standards in a layered architecture for applications based on Web Services: in section 2 we present the framework of the layered architecture. The bottom layer comprises the Web Service stack. In the middle resides the choreography stack and on top of that the NL storybook serving the role of the NL modeling layer. Section 3 outlines and illustrates the rationale and advantages of extending service semantics by natural language. It demonstrates how WS-Talk can support enterprise search applications, for instance. Section 4 concludes the paper.

2. THREE-LAYER INTEGRATION ARCHITECTURE

From the cooperation with first users of WS-Talk we have learned that the universal access to services is almost useless if there is no comprehensive view on both, the universally available (and thus more generic) services a company needs and on peculiarities emerging from their specific business objects and processes. And in addition, this comprehensive view must be sharable among business experts and IT experts. Making services available for business purposes (with or without resorting to web service technology) is extremely intertwined with Business Process Management [2, 6]. The WS-Talk approach is thus inclined to Business Process Management where definition and management of business processes rests on two shoulders: the ones of the domain expert and of the IT expert.
This rationale has to be reflected by the architecture for business integration that follows a three-layered approach (see figure 1): the NL storybook at its top layer resides on the orchestration layer which in turn resides on a Web Service stack. Our approach is inclined to the three-layer stack developed by W3C [10]. It considers the top level as the one that still needs to be developed. Standards are available for the middle and bottom layer. Web Service orchestration and Web Service choreography are concepts addressing the middle layer. BPEL (Business Process Execution Language) and WS-CDL (Web Service Choreography Definition Language) are representatives for such a standard language used for Web Service orchestration. In WS-Talk we investigate to what extend natural language can be used to develop the top layer of this architecture and which impact this has on the orchestration layer. Our objective is to make top layer's semantics human-understandable.
To achieve this goal we propose a number of *sub-layers for the top layer*:

1) **Grammatical analysis.** Natural Language instructions are analyzed in order to figure out which roles and actors are represented in a single statement. For instance, the user's travel agent takes the user's chosen flight, sends it to the reservation system and awaits its response. The *actor* in this case is the user's travel agent which can conduct a number of *actions* (functions) which are usually represented by actions like "send" and "await". This means actors are usually identified by *subject* nouns and actions by verbs (*predicates*). The information required by the actor as well as the one produced by the actor are represented as *object* nouns.
2) **Matching with generic process descriptions.** We introduce a *generic thesaurus on processes* that are addressed by the actors of the storybook instructions. The corresponding thesaurus entry provides the natural language instruction with a first possible service representation of the instruction.
3) **Service annotation**. Services need to be annotated. This means we link the service to a number of descriptive terms as contained in the generic thesaurus for processes. The result of annotation is the most relevant branch of the thesaurus for the service under consideration.
4) **Matching with environment specific parameter descriptions.** A *thesaurus for environment specific information* captures the characteristics of the environment where the universal (generic) processes are implemented. Universally applicable processes require information about the environment in which they are applied. For instance, the service for the statistical analysis of time series needs to know about the database where it can retrieve time series.
At the second layer (orchestration) we concentrate on the observable behavior of Web Services in the context of message exchange between them. BPEL or WS-CDL are be used to describe the interdependencies among Web Service operations; they do not define the process driving the message exchange nor the internal behavior of each service. In WS-Talk we do not apply those standards, but developed a simple choreography layer which inclines to their principles. The choreography of storybook statements is the implicit task of the WS-Talk Process

Engine. This includes also error handling and compensation actions as well as the conditional execution of storybook statements. Again, the main focus of the WS-Talk project is on the use of natural language on the top layer of this architecture.
The Web Service Stack defines a set of Web Services as atomic and generic entities. It does not define a choreography language or any other language that helps to coordinate atomic operations. It contains their protocols and message characteristics. The messages themselves may be wrapped in SOAP envelopes.

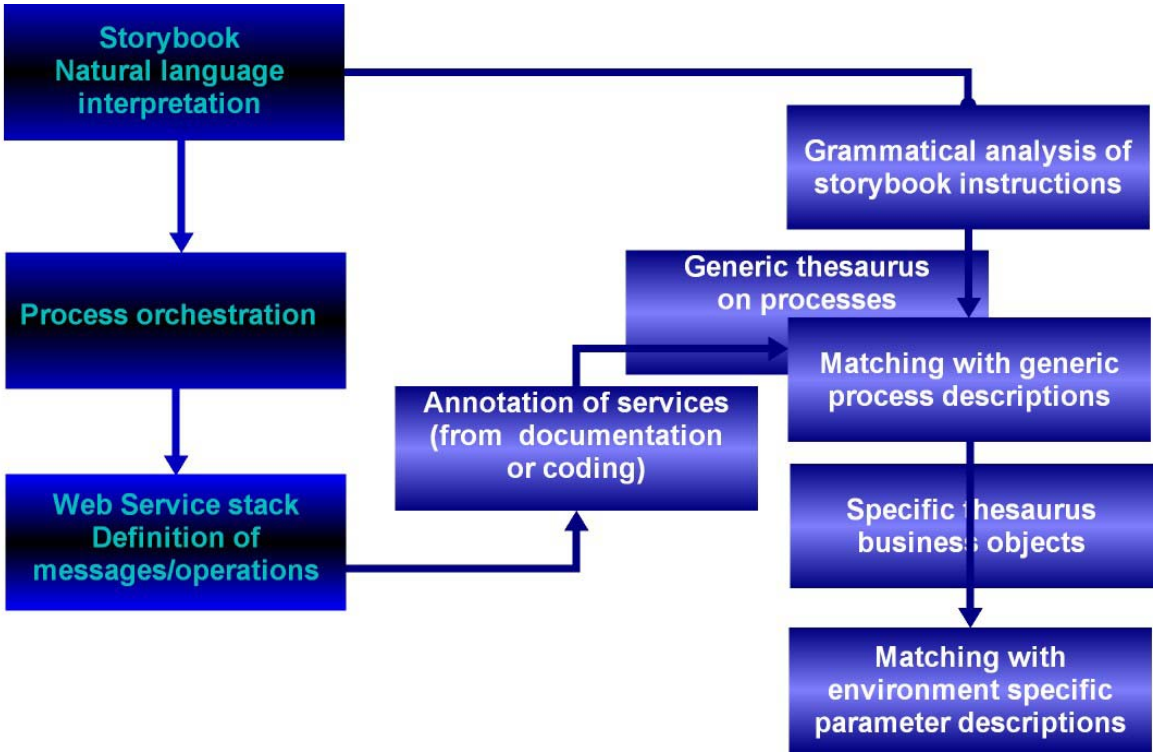

Figure 1: Three layers for a service-oriented architecture (left) including the sub-layers (right) necessary for the coexistence with natural language.

3. BUSINESS PROCESS STORYBOOK AND THE SERVICE EXECUTION ENVIRONMENT

In this paper we concentrate on the storybook that reflects a business process and organizes the workflow across services or software components. In WS-Talk, the Service Execution Environment (similar to a business process engine) finally associates each instruction of the storybook with its corresponding services, executes them, and handles communication and data transfer between them. Thesaurus data and annotations are provided to this engine by the WS-Talk Service Designer.

3.1 Use case

The following example shows a storybook used for a helpdesk application. It refers to a WS-Talk pilot application – an enterprise search system for the Chilean insurance company "Cruz del Sur". The search system uses product descriptions as database, retrieves appropriate documents and extracts text passages from these documents in order to produce tailored retrieval results. The domain expert uses a storybook to describe the application logic – composition of and transactions over software components (or web services) – for a specific retrieval situation (retrieving information from product descriptions, for instance). In this case the helpdesk manager, for instance, describes how incoming user queries have to be handled: First, the user is prompted by the system to enter his query. The query is then processed, i.e. the type of the query is determined and stopwords are eliminated (words with only minor information value for the system's "understanding" of the user's query). Text analysis as applied in retrieval depends on the type of the query (see figure 2). And finally the system prints the result which should be a small number of text passages.

```
How to handle a user request

The user enters a query
Determine type of query
Eliminate stopwords
Use "Cruz del Sur" product descriptions as textbase
In the case of a regular query: Search returns paragraphs
Print the result
```

Figure 2: Example of a storybook. With this storybook the domain expert describes the required behavior of a text retrieval system.

In this example the retrieval system includes the text's structure in its text analysis process. It can indicate the type of information unit that matches the user's query. This is important when we want to restrict retrieval on specific parts of a text like paragraphs, titles, figure captions etc. If the search statement would contain "returns titles and subtitles" instead of "returns paragraphs" the user obtains different retrieval results, for instance.

It should also be pointed out here that an instruction may have a condition under which it is executed. The search feature is carried out if the preceding analysis of the type of query indicates that the user stated a regular query. Otherwise it stops the execution at this stage and returns a message that the storybook could not be executed completely because of this condition.

While analyzing a storybook the WS-Talk process engine treats each natural language statement as a single instruction and considers the whole set of instructions as necessary to complete a particular process (handling a user request, in this case). The process may have different facets (alternative instructions are selected according to the results produced in course of the execution of its instructions). However, for each facet the ACID[1] rationale known from database technology is applied.

3.2 Semantic Coordinates

The Execution Environment is equipped with text interpretation capabilities. Storybooks are analyzed by the engine using a controlled vocabulary reflecting generic processes and specific characteristics addressing the respective business domain where generic processes are applied. We use this vocabulary to describe operations, business objects, collaborations, etc.

Before explaining in more detail how storybook instructions are processed we outline briefly the role of the controlled vocabulary for text analysis. We improve the accuracy of text analysis by classification and taxonomy features resorting to this vocabulary. In WS-Talk, classification is used to set up taxonomies which provide a way to see information around thematic categories. WS-Talk uses taxonomies *and* inverted term lists to annotate an information unit, whatever it may be. The same process can be used for annotating services by resorting to a) the service description section within its WSDL file, b) the documentation associated to a particular software component or service (like the JavaDoc, for instance), c) the programmer's comments, or d) directly to the source code.

Semantic co-ordinates – i.e. controlled vocabularies derived from taxonomies and structured according to concept hierarchies – enable us to develop a context map for the respective domain where the services are applied (the support centre of an insurance company, for instance) or, more general, for the application domain like "text retrieval" or "time series retrieval". These hierarchies can be considered as accepted description standards, at least within the boundaries of a company. Like other language standards they can also be mapped into different natural languages in parallel [3] which support multi-linguality. The process of annotation is thus identifying the adequate semantic co-ordinates for a service, its methods to be applied, and the parameters required and produced by this service. And because of its hierarchical structure the controlled vocabularies are represented in XML format

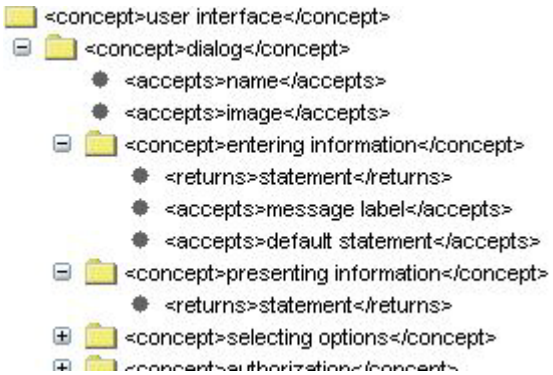

Figure 3. Part of a concept hierarchy as used in the domain "human-computer interaction" to describe a dialog component as part of a generic service supporting different dialog features.

3.3 Setting the scene

The task of the WS-Talk Execution Environment is setting the scene according to the underlying storybook. The process has been described already in chapter 2.

A particular service is available under its title (like "How to handle a user request"). It can be selected and executed by the user (or by other services) like any other application. The process engine operates like a program interpreter. This process starts with a simple analysis of the statement's grammatical structure. In general, the schema subject-predicate-object is applied to identify what actor (user or service) uses a function (represented by the predicate) to operate on a certain object. While a predicate verb is always required, subject nouns and object nouns are optional. Each statement may contain more than just one predicate as well as one predicate-object pairs. The application of a sentence may depend on a condition to be fulfilled.

The grammatical structure of a statement can be represented by (an asterisk indicates that the element may occur iteratively)

[condition] [[subject] [[predicate] [object]]*]

The condition itself can be represented by a simple relationship of "[object] predicate [value]" where "predicate" simply has the quality such as a relationship identifier "is_a". Subjects refer to system components (actors) that reside within a certain application. The statement "The email system sends the results to the user" indicates that the remote mailing system is in charge with passing the retrieved information to the user. Or "The user

[1] The ACID model is one of the oldest and most important concepts of database theory. It sets forward four goals that every database management system must strive to achieve: atomicity, consistency, isolation and durability. No database that fails to meet any of these four goals can be considered reliable. The same holds in a similar way for service orchestration. Even though it is important to note that in service orchestration compensation takes over the importance of roll-back [2]: Instead of rolling back a transaction, orchestration tries to find services which can replace the service which failed. A process spanning over a long period of time is quite often impossible to be rolled back.

enters a request" addresses the subject "user" which in fact refers to the user interface of the system. A statement thus indicates a process represented by a predicate owned by a subject and producing an object, a query for instance, as in our example.

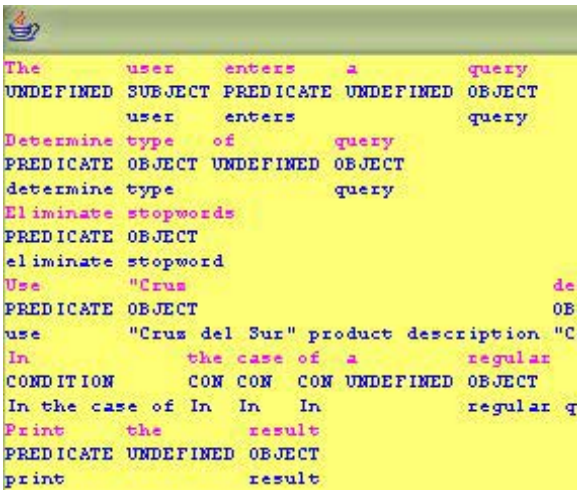

Figure 4. Monitor window showing the results of the grammatical analysis of our example storybook

While processing each statement the Execution Environment looks first for a branch in the concept hierarchy (see figure 3) matching the statement's subject and predicate (eventually including the object). If a corresponding description exists, the engine takes the terms of the generic process representation as synonyms for the natural language expressions. Thus "user – enters – query" is expanded by "user interface – dialog – entering information". The engine registers in this case that "information" and "query" can be treated synonymously in this particular context. The subject "user" is expanded by "user interface" and "dialog". Subjects and predicates represent thus functional semantics, whereas data semantics are represented by objects.

At this moment we assume that there is a software component or service available that matches the same branch of the (hierarchical) representation of the generic process. After successfully matching the generic description ("entering information") with the NL storybook instruction and finally with the annotations of the service the Execution Environment can execute the first instruction.

Things look a bit different at the next instruction "determine type of query", because the engine needs an idea about how to identify different types of a query. This is achieved by providing the engine with information coming from the characteristics from the environment in which it is operating. These environmental variables are made available in the same way like services themselves are made available (see figure 5).

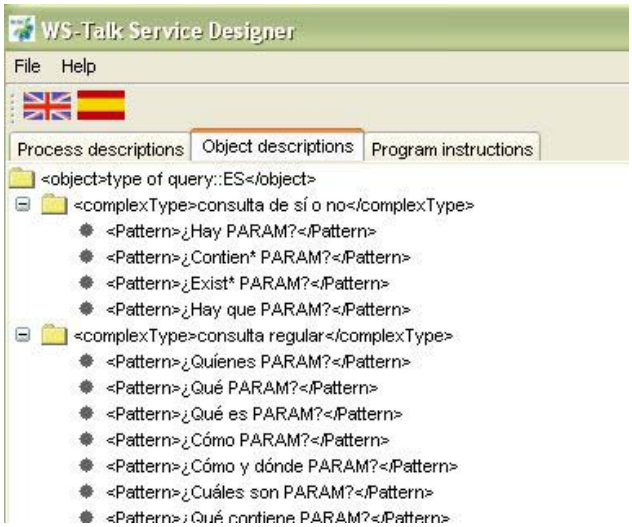

Figure 5. Representation of query types in Spanish

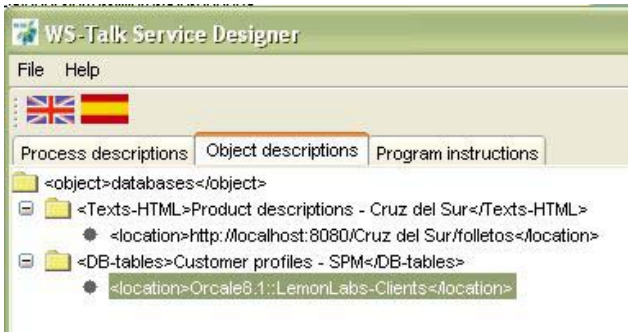

Figure 6. Representation of the access points to different data collections.

If the required environment parameter is available the engine sends the parameters to the process performing the characterization of the query. "Determine type of query" takes the query as input and produces a further object which reflects the nature of the query as stated by the user.2 The service itself makes this information available on request, for subsequent instructions such as search, for instance. The new object produced is used further down to resolve the conditions "in case of a yes/no-query" and "in case of a regular query".

Let us come back to our scenario which demonstrates the application as it results from the storybook presented. The user describes the requested service using natural language. This request, in turn, may also be considered as the description of a particular service that needs to be developed. In WS-Talk, controlled vocabularies are available in different languages in parallel. Even if a particular storybook is written in English, for instance, the retrieval feature as triggered by the storybook mentioned above can handle queries and texts in Spanish.

[2] Nature of a query means: is the answer to that particular query simply yes or no or a more comprehensive reply like a certain paragraph of a document.

Figure 7. The first instruction prompts the user to state a query, in this case: what offers the car insurance called "Cruz Auto"?

Prompting the user to state a query is the first statement of the storybook as mentioned above. From all available product descriptions the system retrieves the most appropriate paragraphs from product descriptions concerning the car insurance "Cruz Auto" (see storybook, figure 2). The annotation process using concept hierarchies is also used to support retrieval of the most suitable information unit. A more detailed description of the corresponding text analysis features is given by Sauvagnat et al. [8]. Printing itself is taken over by the last instruction of the storybook.

3.4 The protocol

The WS-Talk Execution Environment has its own mechanism to handle service orchestration. Every instruction is assigned to a separate, autonomous process which is called agent (inclined to software agent technology). A facilitator serves as the agent manager which monitors the community of agents assigned to a particular storybook. It performs the initial matching process (as described above) which leads to the assignment of software components or services to storybook instructions. In fact, the agent manager transfers the corresponding piece of software to an agent which will execute it in its shell. The agent itself handles the communication between the agents and thus the exchange of input and output parameters between them. Please note that the protocol necessary to execute a service or software components is inclined to that used in Web service technology: each process accepts a number of incoming messages containing the required or applicable input parameters and returns a number of outgoing messages. An agent thus adapts this communication behavior and has in addition a shell for executing the assigned process.

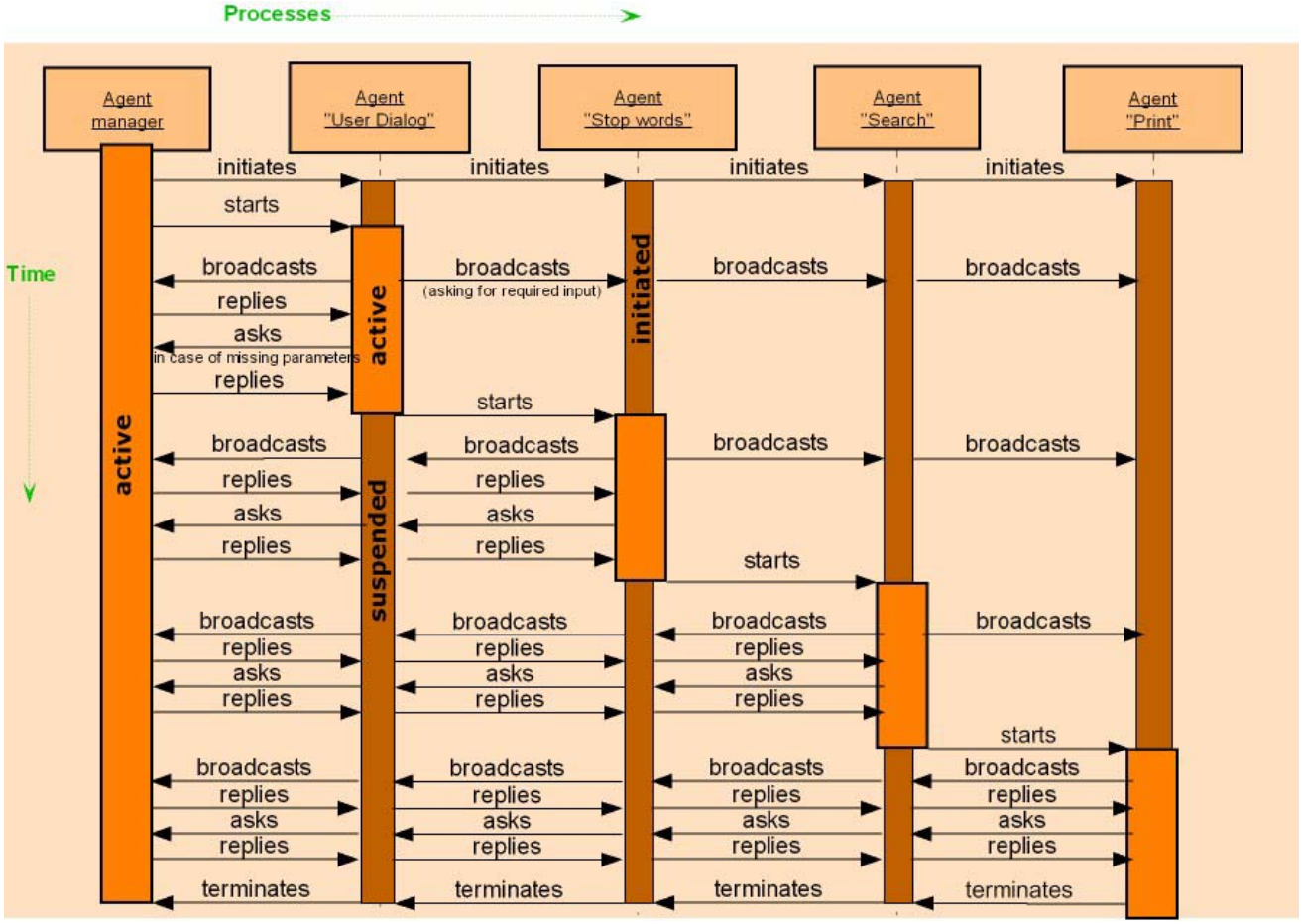

Figure 8. Orchestration protocol as manifested by autonomous processes that execute our example storybook (not all instructions are shown).

The service itself tells the agent about names and types of input and output parameters. By matching these names with the entries of the respective thesaurus the agent can map different names and finds thus corresponding parameters as provided by the facilitator or the other agents. The facilitator is thus also in charge with the environmental parameters which provide external data coming from the application context and required by the application.

In order to get the required input parameters for a service the agent sends a broadcast to all agents and the facilitator asking them if they have variables with certain names available. If an agent has the respective parameter it replies to the agent's request with the corresponding Parameter object. If the processing agent collects all its parameters from the broadcast it starts executing its assigned service. If there are one or more variables missing it asks the preceding agent (i.e. the agent in charge with the preceding instruction) if it has the required variable. And if the preceding one does not have the required information its predecessor is asked and so forth. In this bilateral communication name matching is treated a bit more relaxed. The advantage is that parameter name matching can be almost reduced to type matching. The final instruction in our storybook, for instance, discovers the text to be printed only through this recursive communication along the execution sequence of the agents. No agent provides a variable that is called "result". However, while talking to the "search"-agent both find out, that "paragraphs" and "result" are the things that obviously match here. The complete protocol is shown in figure 9.

Each storybook represents a complete transaction that can be performed only as a whole and must be rolled back completely if an error occurs. In that case the engine can trigger in addition a compensation process that consists at least in a message to the users informing them why the transaction could not be completed and if it will be completed in the future.

The facilitator of the Execution Environment creates an execution stub for each storybook. It lists the sequence of operations to be executed and can thus control the flow of execution and manage error handling.

4. CONCLUSIONS

In the past, Service-oriented Architectures and Web Services had seen a number of successful approaches to integrate application logic and businesses. However, the practice clearly showed that we are far from unleashing their full potential for businesses. There is many a company not using these technologies because they are simple not affordable. Thus, unleashing the full potential means new business opportunities for companies producing and selling new generations of distributed systems and for those using these systems.

Positioned under this thematic umbrella, WS-Talk helps technology end-users, technology designers, and technology providers to perceive and to understand pathways and avenues that lead from current practices to new application areas, to new functionalities.

In WS-Talk we opt for the co-existence of natural language and Web Service technology. Semantic Web representations of objects as well as processes are extended by natural language descriptions. They let users directly interact with web services, business logic representations, or other such objects that are rendered by or operating on Semantic Web standards. The best way for humans to develop their applications is to use their own language. In the vision of WS-Talk, business process experts write storybooks in their own language which are transformed automatically into semantics that can be handled by applications.

5. ACKNOWLEDGEMENT

Research outlined in this paper is part of the project WS-Talk that is supported by the European Commission under the Sixth Framework Programme (COOP-006026). However views expressed herein are ours and do not necessarily correspond to the WS-Talk consortium.

6. REFERENCES

[1] Alonso, G., Casati, F., Kuno, H., Machiraju, V. *Web Services*, Springer (NY), USA 2004

[2] Casati, F., Shan, E., Dayal, U. and Shan, M.-C. Service-oriented computing: Business-oriented management of Web services, *Communications of the ACM* 46 (10), 2004, pp. 55-60.

[3] Englmeier, K. and Mothe, J. Natural language meets semantic web. Retrieved July 16, 2003 from *ktweb.org* website: http://www.ktweb.org/doc/Englmeier-NLP-SW.pdf.

[4] Huhns, M.N. and Singh, M.P. Service-Oriented Computing: Key Concepts and Principles, *IEEE Internet Computing* 9 (1), 2005, pp. 75-81.

[5] Mahmoud, Q.H., *Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI)*. Retrieved November 16, 2005, from *Sun Microsystems* website: http:// java.sun.com/developer/ technical/Articles/WebServices/soa/

[6] Milanovic, N. and Malek, M. Current Solutions for Web Service Composition, *IEEE Internet Computing* 8 (6), 2004, pp. 51-59.

[7] Peltz, C. Web Services Orchestration and Choreography, *IEEE Computer*, (10) 36, 2003, p. 46.

[8] Sauvagnat, K., Hubert, G., Boughanem, M., and Mothe, J. *IRIT at INEX 2003* INitiative for the Evaluation of XML Retrieval (INEX 2003), pp 142-148, 2003.

[9] Shi, X. Sharing Service Semantics using SOAP-Based and REST Web Services, *IT Professional* 8 (2), 2006, pp. 18-24.

[10] W3C, *Web Service Choreography Interface 1.0*. August 2002, Retrieved November 4, 2005, from *W3 Consortium* website: http://www.w3c.org/ TR/wsci

[11] WebServices.Org, *From Web Services to SOA and Everything in Between: The Journey Begin*. Retrieved June 14, 2005, from *WebServices.Org* website: http://www. webservices.org/index.php/ws/content/view/full/63404