

# SOA Governance using Tagging

Harald Meyer and Guido Laures  
Hasso-Plattner-Institute, University of Potsdam, Germany  
harald.meyer@hpi.uni-potsdam.de  
Software AG  
Uhlandstrae 12, 64297 Darmstadt  
guido.laures@softwareag.com

**Abstract:** Finding services in large service landscapes and managing large service landscape is an important problem of service-oriented systems. While semantic Web services promise to overcome this problem, using them is difficult, they only capture a subset of the available meta-data, and are limited to the provider's point of view.

In this paper, we present an approach to service annotation based on tags. Instead of describing service meta-data upfront, it is added to the system in a collaborative manner. We show how this works in principle, identify different use cases for tagging (like describing functional aspects or aspects of service usage), and demonstrate how we integrated tagging into Software AG's registry and repository product Centrasite.

## 1 Introduction

The goal in introducing a SOA is to reduce logical dependencies between IT components and to close the gap between business requirements and their representation in the IT landscape [Bur00, PG03, ACKM04]. The introduction of a service layer to decrease and control logical dependencies, however, comes at the cost of increased technical complexity. The control and steering of IT assets and their life cycle in such a complex SOA landscape is what is referred to as SOA Governance. Governance is key to the success of SOA initiatives and requires support in terms of methodologies and tools. SOA governance tools aim at regulating the life cycle of a service or other reusable IT asset from its inception to its retirement. During its life cycle a service will pass a number of steps that are all defined and controlled in the SOA governance system. An IT asset's life cycle can be divided into two phases: the providing and the consuming phase. The providing phase covers the part of the life cycle that regulates how a service goes into production. The consuming phase defines how it will be consumed once it is in production. In this paper we concentrate on how Web 2.0 mechanisms can be applied to better support the consuming life cycle phase.

Discovering a service as a first step in the consuming life cycle gets more difficult in large service landscapes. Semantic Web services [MSZ01] are a promising approach to find services based on functionality. Service functionality is described through preconditions and effects. But creating them and writing queries to find services according to preconditions and effects is a complex task. Additionally, semantic service specifications only capture functional aspects. Although some standardization efforts like Web Service Modeling On-

tology (WSMO) include quality of service specifications, other aspects are important as well. Functional descriptions and the service interface are in many cases not sufficient decision criteria for a service consumer. Additional information that might be important to know for a consumer are:

- In what business contexts has the service been used (successfully)?
- How often has this service been found and bound to consumers already?
- How do other users of the SOA governance system rate this service?
- What other services have been used by consumers of this service?

These questions can hardly be added to the functional description by the service provider. They depend on the usage of the service in the organization. Therefore, it is of high value to let users add their individual functional and technical descriptions to the service metadata in an easy and informal way like tagging. Furthermore, the SOA governance system can maintain dynamic tags that depend on the usage of a service like its popularity or its performance. The advantage of letting users add their view on the service with individual tags is that the collection of tags provide a common view on the service rather than only the view of the provider. The combination of these tags make it more easy for consumers to discover and bind a service and therefore increase the value and efficiency of an SOA.

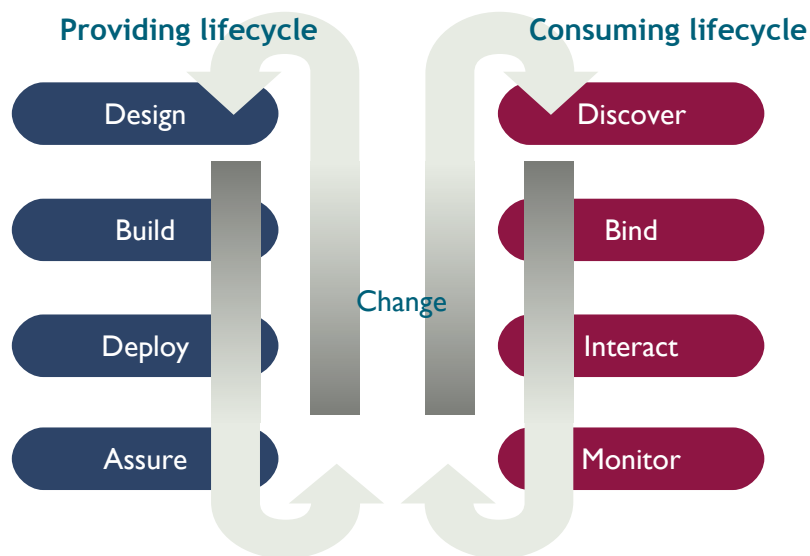


Figure 1: The two phases of an IT asset's lifecycle

The rest of the paper is structured as follows. In the next section, we give an overview of related work. Section 3 provides the formal foundations for tagging services. Section 4 shows possible use cases of tagging in SOA governance. How we implemented our approach using Centrasite is then shown in Section 5 followed by the conclusion.

## 2 Related Work

SOA Governance is an immature and new topic in the industry that came up with the SOA hype starting in the early years of this century. The problem domain, however, is not new. The structuring of IT assets and the definition and enforcement of methodologies and processes to drive their delivery have been around in the industry for a long time. There are several ways how companies are approaching the SOA governance space: enterprise architecture, service management, service delivery processes (such as ITIL) and configuration management. All of these disciplines have their own focus that have its intersection with SOA (see figure below). Hence, tools in this area also related to the SOA governance-related challenges. The idea of SOA as the connecting piece between all these disciplines, however, justifies the introduction of a newly defined discipline that specifies on SOA and its huge set of related standards. Leading IT product vendors like Software AG, HP, IBM and Oracle provide specialized tools to support the SOA governance use case. None of these vendor's tools provide mechanisms like the ones described in this paper.

The approach presented is similar to *service categories* in OWL-S and SAWSDL. In contrast to tags are service category assignments mostly static. During development, the service engineer assigns one or more suitable service categories to the new service. A strict distinction between service development and service usage prevents changes during run-time by people other than the service engineer. Additionally, service categories are mainly used classify a service according to its functionality. Capturing non-functional aspects or aspects of the service landscape is often not envisaged. Looking at existing categorizations or taxonomies of service functionality supports these arguments. Systems like NAICS (North American Industry Classification System), UNSPSC (United Nations Standard Products and Services Code), or RosettaNet have defined processes for changes to their taxonomies. Adding new concepts may take up to 5 years (for NAICS). Categories are statically assigned and describe only functional concepts: For example, it rarely makes sense to change a service categorization in UNSPSC from *4410260214* (Retail of Printers) to *4410310314* (Retail of Toners). Changing an existing service so much is bad service engineering. Instead one would remove the existing service and publish a new one.

Our work is inspired by the recent advent of emergent semantics and folksonomies. Both approaches do not depend on a-priori semantical commitments. Instead, semantic commitment is achieved incrementally through local interactions. Folksonomies are a practical approach of emergent semantics where local interactions are replaced by a central storage for semantic commitments.

## 3 Formal Foundations

### 3.1 Emergent Semantics

Technologies for semantic annotation originally come from the annotation of documents in general (e.g.: author, keywords describing the content). Recently, these technologies

are used to specify service functionality [MSZ01]. When annotating documents in general, annotations are created either by dedicated professionals or the authors of the documents [RF00, Mat04]. While professionally created annotations are of high-quality, their creation is costly and rarely scales for large amounts of documents. Author-created annotations overcome these problems. But both approaches suffer from the same drawback: The annotations are not created by the actual users. This makes it possible that the annotations do not fit the actual usage of the documents. Adaption to changing or unintended uses is not possible. Another problem is the static structure of ontologies. Ontologies represent a-priori agreements on required concepts. These agreements are static and can again not represent changes in the usage of the documents. Hence, a new approach is emerging: emergent semantics [AMO<sup>+</sup>04]. With emergent semantics, a-priori agreements are replaced by incremental, local agreement: new concepts are introduced as required to reach local agreement between two parties. Local agreements are then aggregated to agreements involving multiple parties until a global consensus on the meaning of a new concept is reached. This approach can deal with the *ontological shift*: the meaning of concepts and documents changes over time when users or uses change.

The recent rise in *folksonomies* can be seen as an application of emergent semantics. The term *folksonomy* is a composition of folk and taxonomy. In a folksonomy annotations for documents are created by the users of the system through tagging. The most prominent examples for systems based on folksonomy are del.icio.us (<http://del.icio.us/>) and flickr (<http://flickr.com/>) bookmark and photo management systems. When adding a bookmark in del.icio.us you can add multiple tags or categories to the bookmark. Later, tags can be used to find the bookmark again. Another feature of folksonomies is their community orientation. Bookmarks and tags are shared among all users. Hence, a user cannot only find all the bookmarks he tagged with a given tag, but he can also find all the bookmarks tagged with the same tag by all other users. Folksonomies are a special form of emergent semantics. With emergent semantics, consensus about semantics is reached through peer to peer interactions [AMO<sup>+</sup>04]. For simplicity reasons, folksonomies depend on a central store where the annotations are stored. Hence, no direct interaction among users happens.

The freedom resulting from the usage of tags leads to problems illustrated by Golder and Huberman [GH05]. It is possible that a tag has multiple homonymous or polysemous meanings. While polysemous meanings are closely related, homonyms are words with multiple unrelated meanings. Besides having multiple meanings for one tag, it is also possible that more than one tag exists to represent a specific meaning. Such synonyms can be especially complicated in a tagging-based system as user using one tag, will not find documents tagged with the other tag. Another problem is basic level variation as users do not have to agree upon a common level specificity. As users can create tags at their will, they do not need to adhere to a naming convention. This is problematic as it is unclear whether a tag should be in singular (e.g. *book*) or in plural (e.g. *books*). How tags consisting of more than one words are composed is also undefined: write them as one word, separate them with underscore, two tags, etc. . Similar problems occur on the level of message-level heterogeneities when Web services exchange data [She98, NVS<sup>+</sup>06] and in multidatabase systems [SK92, KCGS93].

Michlmayer [Mic05] identifies spamming and the limited user interface (limited search functionality) of existing folksonomy implementations as further problems. The still unproven assumption why these problems do not interfere with the actual usability of existing systems, is that most of them do not matter if only enough users participate. For example the problems with synonyms is that two different tags for the same meaning lead to separated document landscapes. But if enough users participate, chances are high that most documents get tagged with both tags solving this issue. As a last point in this section we describe a formalization for emergent semantics introduced by Mika [Mik05]. This formalization will later serve us as the basis for our formalization of emergent semantics for service annotation. A folksonomy:

**Definition 1** A folksonomy  $F \subseteq A \times T \times O$  is hypergraph  $G(F)=(V,E)$  with

- $V = A \cup T \cup O$  as the vertices. The disjoint sets  $A = \{a_1, \dots, a_k\}$ ,  $T = \{t_1, \dots, t_m\}$ ,  $O = \{o_1, \dots, o_n\}$  are the set of actors, the set of tags, and the set of objects.
- $E = \{\{a, t, o\} | (a, t, o) \in F\}$  as the hyperedges connecting an actor  $a$  who tagged an object  $o$  with the tag  $t$ .

Figure 2 illustrates an example of a folksonomy with  $A = \{Pete, Mary\}$ ,  $T = \{book, books, news\}$ , and  $O = \{\text{http://amazon.com}, \text{http://cnn.com}\}$ .

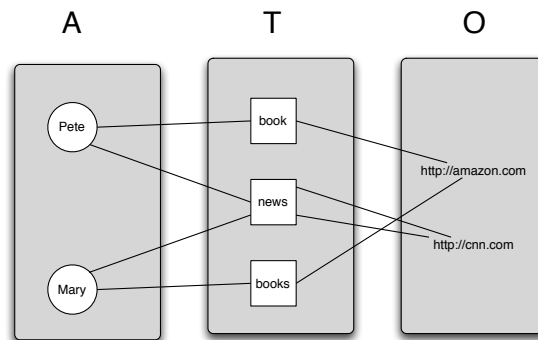


Figure 2: Folksonomy Graph

### 3.2 Emergent Semantics for Service Annotation

A similar situation like for documents in general applies to service descriptions. To find them it is important to annotate them semantically, but existing author-created or professionally created semantic annotations, are costly to produce and have the risk of not fitting the actual usage of the service. Hence, we will apply the concept of tagging in this section as a light-weight approach towards semantic service annotations.

**Definition 2** A service is a discrete business functionality. It is described by a service description. A service description  $s = (N, \mathcal{I}, \mathcal{O})$  is a tuple with

- $N$ : the name of the service
- $\mathcal{I}$ : List of input parameters
- $\mathcal{O}$ : List of output parameters

A service landscape is the set of available services described by service descriptions  $S = \{s_1, s_2, \dots, s_n\}$ .

This definition is rather abstract in comparison to the service definition of WSDL. In WSDL, services can have more than one operation and each operation has at most one input and one output message. But the relation of these operations is unclear. Is each operation an individual service and hence a WSDL service only a collection of services or can the service only be achieved by invoking the operations in a well-defined order? Our service model can be mapped to both approaches. If each operation is an individual service, then the input parameters  $\mathcal{I}$  and the output parameters  $\mathcal{O}$  can be mapped to elements of the input and output messages. If one WSDL service with multiple operations is one service, then  $I$  and  $O$  are the input and output messages of the individual operations.

The consensus of upcoming semantic web service *standards* OWL-S, WSMO, and WSDL-S is that service functionality is described through preconditions and effects. WSMO also introduces assumptions and postconditions. It distinguishes between information space and world space. SAWSDL explicitly excludes the modeling of capabilities. The precondition defines if a service is invocable in the current state. Both the state and the precondition are described through logical expressions. A formal specification of the functional description of services can be found in [KLS06]. If the precondition is satisfiable by the current state, the service is invocable. The effect describes the changes to the current state that result from invoking the service.

With our approach preconditions and effects are no longer necessary. Instead the notion of tagging is applied to semantic services:

**Definition 3** A tagging-based semantic service system with a service landscape  $S$  is a folksonomy where the objects are the service landscape:  $F \subseteq A \times T \times S$ .

This means that the objects that we tag, are now service descriptions. The tags can be used to express the functionality of the service. The actors in such an environment are for example process designers, service landscape managers, and service engineers.

## 4 Using Service Tagging

In this section we describe how tagging can be used for working with large service landscape. We will first present the scenario we are using and then introduce six different use cases for tagging.

## 4.1 Scenario

The scenario we use is taken from a Software AG Centrasite plugin to manage the life cycle of services and other artifacts in a service landscape. As you can see in Figure 3 the service landscape consists of nine typical ERP services. With this, the landscape is rather small. But to illustrate tagging use cases it is sufficient. Another aspect of this service landscape is that its services are high-level services. A service like *Personnel Development* is rather abstract. In reality it will consist of several, more fine granular services.

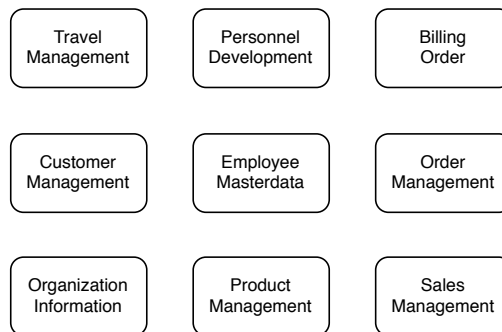


Figure 3: Service Landscape

## 4.2 Use Cases

The use cases for service tagging are use cases for three distinct roles: service engineer, process engineer, and service landscape manager. The service engineer develops new services. A process engineer develops new processes based on the available services and the service landscape manager is for example responsible for knowing what the service landscape looks like, updating services, or making services easy to find. In short, he manages the service landscape.

**Tagging service ownership** The owner of a service can be a person, a department, or an organization. The person could be the service engineer who developed the service or somebody who is responsible for the operation of the service. This can be used to know whom to contact in case of problems. Tagging service ownership on the level of departments can give information about which departments participate in a process or whether services with similar functionality are provided by different departments. Tagging service ownership on the level of individual persons is done by the service engineer and the service landscape manager. On the department level it is done by the service landscape manager. One example of a service ownership tag is a tag *HR* for *EmployeeMasterdata*, *OrganizationInformation*, and *emphTravelManagement* to denote that they are provided by the human resources department.

**Tagging service functionality** Using tags to express service functionality is closest to what you normally do with semantic service specifications. But instead of expressing the functionality in formal way using preconditions and effects, tags represent just keywords on what the service does. An initial tagging of service functionality might be done by the service engineer who developed the service. But tagging service functionality gets to its real strength when it is used by service users (e.g. a process engineer) to describe what a service does for them. This can help to capture unintended usages of services as well as bridging terminology differences between service engineers and service users.

The *OrganizationInformation* service might have been developed with users from human resources in mind, but is actually also used in processes of other departments as well (e.g. to find someone responsible for an approval). This use might not be captured in the tags provided by the service engineer. The first process engineer to use the service in this manner might decide to tag this new, unintended usage to help himself or others to find the service in the future. An example for different terminology between service engineers and user could be the *TravelManagement* service: is it used to approve and book business trips or is it used by employees to request leaves of absence or vacations? Users in the human resources department can tag the service to describe what it does in their own words.

**Tagging service affiliation** Affiliation describes in which contexts a service is used. The context is the business processes or the applications in which a service is used. Tagging affiliation has two advantages. First, it allows impact analysis of what happens when a certain service is no longer available. Combined with information about in which system a service is implemented, the information can be used by service landscape managers to know whether a system is still necessary or which processes need to be changed if a system is shut down. The second advantage of tagging service affiliation is that it gives an overview of which services are often used together. This information can be used by process engineers to find services. Based on already selected services, they can browse the service landscape to see which services are used together with these. In an advanced version, this browsing could be done by the process modeling tool. The tool checks which services might be interesting for the current modeling situation and suggests them. Another elaboration of tagging service affiliation is to perform the tagging automatically based on modeled processes.

A slightly different form of tagging for service affiliation, is tagging negative affiliation. Negative affiliation means that services cannot be used together. Most likely, this would have to be done manually to capture experiences about the service landscape.

**Tagging service quality and characteristics** This use case is done by process engineers and landscape managers to capture experiences of service usage. To a smaller extend, the service engineer can tag non-functional attributes like cost, security, or physical location. This usage is closest to how semantic service specification languages see service quality. These tags are rather static and do not contain information about the actual operation of service. On the other hand, the tagging by process engineers and landscape managers is about how a service operates. This includes the performance and reliability of services.

**Tagging Service Life Cycle** Tagging the life cycle of service is a very easy but rather limited approach to life cycle management. Much more elaborate approaches exist. A tag can represent a phase in the life cycle of service. For example, each service from our scenario is actually available in different versions. As the versions of the services are not aligned, it is often unclear which service versions are the current ones, which are currently in development, or which are already deprecated and should not be used in new processes.

The service landscape manager could for example assign a tag *current* to the current versions of each service. As the names of life cycle management tags are not predefined, the landscape manager can use his own terminology or follow an existing life cycle.

**Tagging for task organization** A service engineer might mark services he needs to work on (e.g. to fix bugs). The process engineer might group services he will use in a future process (without actually modeling the process or knowing whether the services fit together). A service landscape manager might do some *landscaping* of the service landscape and mark which services he still has to look at. For all three, the tags will disappear as soon as the tasks are finished and the tags will normally be of no use for others. In these two aspects, tagging for task organization differs from all other use cases.

## 5 Implementation

We integrated the tagging functionality into Centrasite as a plugin. The plugin has an own UI that allows to tag services, browse services by tags, browse the tag cloud etc. The UI is integrated into Centrasite Control. Figure 4 shows the how this is done architecturally. Another component in Figure 4 is *Oryx*<sup>1</sup>. Oryx is a Web-based modeling tool developed at the Business Process Technology group at the Hasso Plattner Institute. It is currently mainly used to model business processes using the Business Process Modeling Notation (BPMN) but stencil sets exist to, for example, also model EPCs or Petri nets.

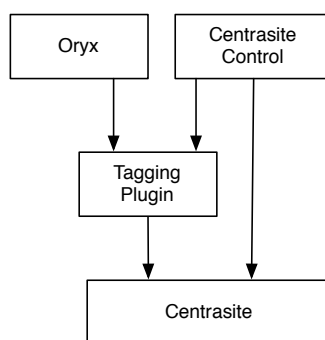


Figure 4: Architecture

<sup>1</sup><http://bpt.hpi.uni-potsdam.de/Oryx/>

The non-UI part of the tagging plugin interfaces with Centrasite using the *Java API for XML Registries (JAXR)*. Hence, this part can be used by an registry that supports JAXR. All the tags of user as well as all tags by all users form classifications schemes. The tags for an individual service are classifications.

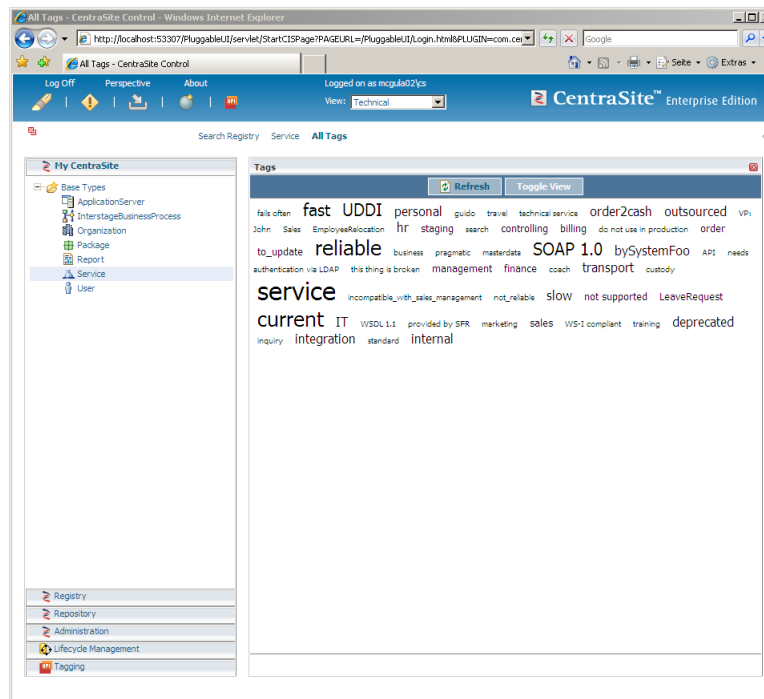


Figure 5: Tagging inside Centrasite

Figure 5 shows a screenshot of the tagging plugin inside Centrasite Control. Displayed is the tag cloud of all tags. The tag cloud lists all tags and the tags are sized according to their frequency. The *deprecated* tag is the biggest as multiple, deprecated versions of each service exist. Using the *current* tag, a user can easily find the newest version of each service. The other tags are the results of the different use cases we presented in the previous section. Besides looking at the tag cloud, the plugin includes screens to edit the tags of services, list services with a specific tag, and search for services based on tags.

Figure 6 is a screenshot of Oryx. On the right hand side, it includes inspectors to access a Centrasite repository. One can browse the repository using tags and if one has found a suitable service, drag it onto the modeling canvas to integrate it into the current process. The aforementioned functionality for suggesting services based on the current modeling context, has not yet been implemented.

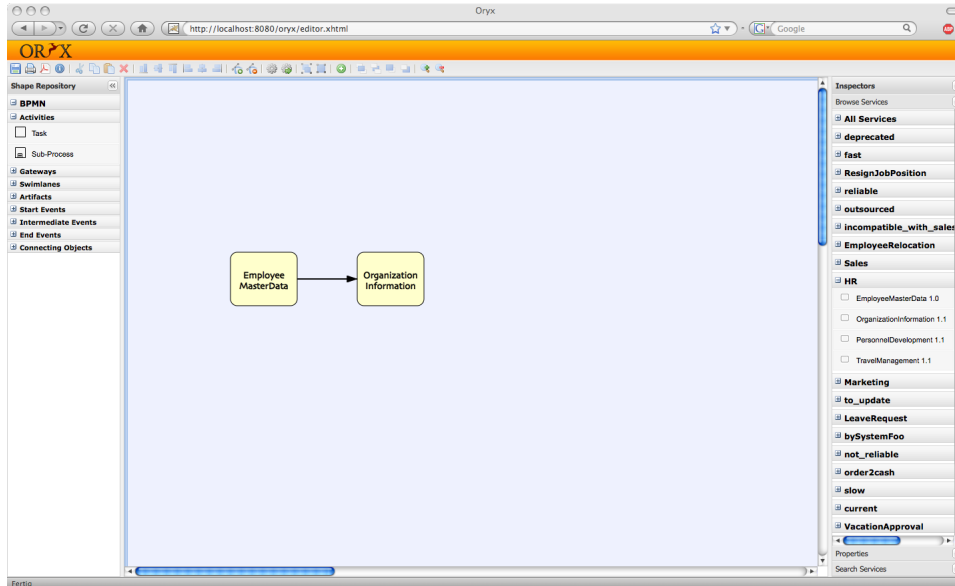


Figure 6: Tags during Process Modeling in Oryx

## 6 Conclusion

In this paper we showed how a lot of the task that come up when working with large service landscapes can be handled using relatively simple techniques from the Web 2.0 area. Tagging allows us to assign keywords to service to categorize them according to their ownership, their functionality, their affiliation, their quality characteristics, their point in the life cycle, or for task organization.

We implemented service tagging as a plugin for the Centrasite repository and registry and showed how this can be used to manage service landscapes. An integration into our Web-based process modeling tool Oryx demonstrated how tags can facilitate process modeling.

We like to further investigate automated tagging and automated tag usage. Automated tagging derives tags from user actions or the operation of services and processes. Automated tag usage can be used in process modeling or landscape management to suggest services and actions. Investigating larger service landscape and how the organizations working with them can benefit from service tagging is another area of future research.

## References

- [ACKM04] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services – Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer, 2004.

- [AMO<sup>+</sup>04] Karl Aberer, Philippe C. Mauroux, Aris M. Ouksel, Tiziana Catarci, Mohand S. Hacid, Arantza Illarramendi, Vipul Kashyap, Massimo Mecella, Eduardo Mena, Erich J. Neuhold, O. De Troyer, Thomas Risse, Monica Scannapieco, Fèlix Saltor, Luca de Santis, Stefano Spaccapietra, Steffen Staab, and Rudi Studer. Emergent Semantics Principles and Issues. In *9th International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, pages 25–38, 2004.
- [Bur00] Steve Burbeck. The Tao of e-business services. *IBM developerWorks*, 2000.
- [GH05] Scott Golder and Bernardo A. Huberman. The Structure of Collaborative Tagging Systems. *Journal of Information Science*, 2005. to appear.
- [KCGS93] Won Kim, Injun Choi, Sunit K. Gala, and Mark Scheevel. On Resolving Schematic Heterogeneity in Multidatabase Systems. *Distributed and Parallel Databases*, 1(3):251–279, 1993.
- [KLS06] Uwe Keller, Holger Lausen, and Michael Stollberg. On the Semantics of Functional Descriptions of Web Services. In *Proceedings of the 3rd European Semantic Web Conference (ESWC2006) (to appear)*, 2006.
- [Mat04] Adams Mathes. Folksonomies - Cooperative Classification and Communication Through Shared Metadata. 2004.
- [Mic05] Elke Michlmayr. A Case Study on Emergent Semantics in Communities. In *Proceedings of the Workshop on Social Network Analysis, International Semantic Web Conference (ISWC)*, November 2005.
- [Mik05] Peter Mika. Ontologies Are Us: A Unified Model of Social Networks and Semantics. In *Proceedings of the 4th International Semantic Web Conference (ISWC2005)*, number 3729 in LNCS, pages 522–536. Springer, 2005.
- [MSZ01] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. Semantic Web Services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.
- [NVS<sup>+</sup>06] Meenakshi Nagarajan, Kunal Verma, Amit P. Sheth, John A. Miller, and Jonathan Lathem. Semantic Interoperability of Web Services – Challenges and Experiences. In *Proceedings of the 4th IEEE Intl. Conference on Web Services*, 2006. (to appear).
- [PG03] M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing: Introduction. *Communications of the ACM*, 46(10):24–28, 2003.
- [RF00] Jennifer Rowley and John Farrow. *Organizing Knowledge: Introduction to Access to Information*. Gower Publishing Limited, 2000.
- [She98] Amit P. Sheth. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. In *Interoperating Geographic Information Systems*, pages 5–30. Kluwer Academic Publishers, 1998.
- [SK92] Amit P. Sheth and Vipul Kashyap. So Far (Schematically) yet So Near (Semantically). In *Conference on Semantics of Interoperable Database Systems*, pages 283–312, 1992.