Personalized Access to Web Services in the Semantic Web

Nicola Henze and Daniel Krause

IVS - Semantic Web Group, University of Hannover,
Appelstr. 4, D-30167 Hannover, Germany
{henze,krause}@kbs.uni-hannover.de

Abstract. We propose a Semantic Web browser which enables users to discover, select and personalize Web services in the Semantic Web. We discuss how such a Semantic Web browser can offer domain specific visualization, and investigate solutions for supporting individual users in finding appropriate Web services, in tailoring Web services to their needs, and in accessing and administering their favorite Web services. As a proof-of-concept, we show the realization of a Semantic Web browser, the personal podcast syndication service *MyEar*, which collects and filters data from various providers of podcast feeds.

keywords: semantic web, semantic web browser, personalization, web services

1 Motivation

The Semantic Web aims at enabling a huge, dynamic and federated network of media entities and information, enriched with machine-processable semantics. Many of today's research initiatives in the Semantic Web focus on developing technologies, languages, reasoning languages and tools for realizing an appropriate backbone for the Semantic Web vision. While these enabling technologies are being developed, the question of how to assist users to benefit from the emerging Semantic Web, thus on how to realize the second part of the vision by T. Berners-Lee et al. [2]: "better enabling computers and people to work in cooperation" currently receives less attention.

To realize one of the most promising advantages of the Semantic Web – the possibility to personalize output according to user's needs – Semantic Web services have to cope with three user-centered issues:

- allowing users to specify their needs (customization)
- optimizing result evaluation according to explicit and implicit needs of the user (adaptation; explicit needs are directly obtained during a particular interaction, implicit needs are derived from previous interactions and are interpreted and consolidated by aid of a user modeling component)
- presenting their results in a way that
  - user-side applications can visualize the results

- transparency and controllability of the result-determining processes and the adaptation steps are guaranteed.

The paper is organized as follows: In section 2 we briefly review related work. Afterwards, we present the architecture of our approach for Semantic Web browsing and describe the different stakeholder within the architecture and their interplay. As a proof-of-concept of our approach, we describe our current prototype of a Personal Music Syndicator which facilitates personalized access to podcasts. Finally, a discussion of our approach, and summary and outlook on current work end the paper.

2 Related Work

Currently, we can distinguish two main strategies for creating Semantic Web browsers: The first strategy visualizes RDF documents without taking into account any particularities of the RDF documents. Examples are Piggy Bank, Longwell[1] or Brownsauce[2]. These browsers are, more appropriately, called RDF browsers.

The second strategy for creating Semantic Web browsing is focusing on a certain domain, which might be narrow (as in the case of DynamicView [4] or mSpace [12]) or broad (Haystack [11] or SEAL [5]). These approaches' architectures are all based on a domain-specific fundament requiring considerable modifications for applying them in other domains. At this time there exists no approach that copes with both issues at the same time: Being generic enough to handle any application domain while offering a domain optimized user interface.

Personalizing the access to Web content is a large area of research, which we cannot describe in this paper with sufficient detail. Instead, we want to point out one particular problem that arose in the recent years in the area of adaptive interactive systems: the open corpus problem [3]. The open corpus problem states that today's adaptive systems code require information for the personalization process within the corpus of documents with the effect that the complete corpus of documents must be known (and thus fixed!) during the design time of the system. One very important effect of the open corpus problem is the limited re-usability of adaptive functionality, which might be one of the reasons why personalized systems have not been so prominent in the Web as might have been expected. For the e-Learning domain, we have shown that the separation of personalization-specific information and the document corpus is possible [7]. Based on these results, we started to develop several Web services for the e-Learning domain, which offer encapsulated and re-usable adaptation functionality for e-Learning, and generalized our concept for the realization of Web services which offer personalization functionality in other domains like publication viewing or music recommendation.

[1] http://simile.mit.edu/longwell/
[2] http://brownsauce.sourceforge.net/

Finally, related work to our approach can be found in projects which apply Web services in the Semantic Web. Current research here focuses more on enabling technologies like Web services discovery, composition and orchestration (cf. [9, 10]) and less on usability and user-centered visualization.

3 A Service-based Architecture for Realizing Personalized Semantic Web Browsing

Our approach for a Semantic Web browsing offers users a uniform entry point to access the Semantic Web, and in particular to Web services in the Semantic Web. It has been realized as part of the *Personal Reader Framework* [1, 6] which offers an environment for designing, implementing and realizing Web content readers in a service-oriented manner (see Figure 1):

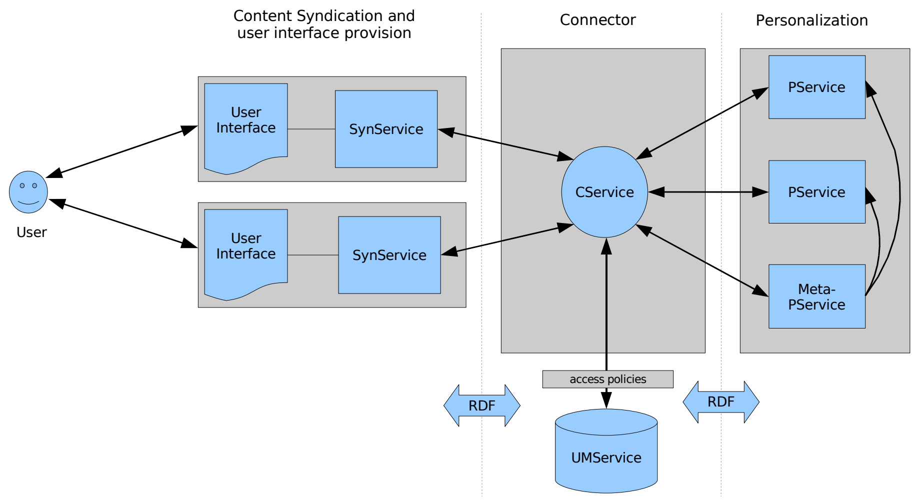


Fig. 1. Overview of Personal Reader architecture

- Web services deliver personalized recommendations for content, extracted and obtained from the Semantic Web. Using Semantic Web techniques they describe their offered functionality in a machine processable format. Optionally, they can return visualization templates that can be used to create user interface snippets for presenting the results of the Web services. We call these kind of Web services *Personalization Services*, *PServices* for short.
- *Meta PServices* can be employed to have single entry points to cooperating or concurrent PServices. For example, a music recommender Meta Personalization Service orchestrates different PServices delivering personalized music recommendations. In an internal processing step the music Meta-Web service

filters and orders all the resulting recommendations of the different PServices into one single result.

- For syndicating the results of PServices and for creating appropriate user interfaces, *Syndication Services* (*SynServices* for short), are responsible for displaying the results of several PServices and/or Meta PServices to the user. Each SynService provides such an user end point to a certain domain or task, and allows the user to benefit from many PServices simultaneously, which are selected, combined and customized as the user wishes. SynServices can be realized as RDF browsers, can implement their own RDF processing interface, or they can make use of visualization templates provided by the different PServices / Meta PServices.
  The *MyEar Music Syndicator* is an example of such a SynService, which provides the user end point to specify requests for podcasts. Throughout the paper, we use the MyEar Music Syndicator to illustrate our ideas of browsing information in the Semantic Web. Other examples of SynServices provide a user end point for viewing learning objects in an embedding context (where each aspect of this embedding context – like recommendations for quizzes, for examples, for further details with respect to the learning object's content, etc., is provided by different PServices), or a user end point for browsing scientific publications of a large European project [1].
- For enabling the whole process, a core information provider – a user modeling service, *UMService* for short, deriving appropriate user profiles – is essential. In our architecture, the UMService realizes a centralized approach for user modeling, maintaining and protecting information about a user on behalf of this user. The main reason for choosing a centralized approach is to realize privacy protection: the user has full control about his user profile, and can define policies on which parts of this user profile might be public, or are available to trusted parties only. One central instance for all this information makes it possible to create awareness about stored information, and on realizing policy-protected access. Furthermore, this central UMService receives updates from SynServices or PServices in different domains, and allows for cross-domain re-use of user profile information (if the user wants that).
- From a technical point of view, another component is required to maintain the communication between the SynServices providing the user interface, the PServices, and the UMService. This is the so-called *Connector Service* (*CService* for short) which harvest Web service brokers, collects information about detected PServices (for discovery, selection, customization, and invocation), and for organizing the communication between all involved parties, including requests to the UMService.

3.1 Using the Personal Reader Framework

First, the user has to specify his current request by formulating a query. For the matchmaking between user's query and provided functionality of detected PServices, the CService offers functionality for the required matching as e.g. described in [8]. The goal is to discover PServices that can incorporate and adapt best to the provided user data from both, the actual user's request and user profile data. N.B.: Not all PServices need to receive the same user profile data, as some of them might be trusted more than others. The necessary negotiation based on the user-defined policies in the UMService and credentials of the PServices have to be executed beforehand.

The discovery is done as following: First, the Syndicator searches for Web services having the required input and output parameters (*exact match*). This search is send as a request to the CService, which executes the search. If the exact match does not return any results, the CService can do a *plug-in match*, returning Web services that require more input parameters than given. These additional parameters have to be entered by the user in the second step, the configuration step. If the plug-in match cannot find appropriate Web services, *subsumed match*, *subsumed-by match* or a *best-match* can be performed. Which of the matching strategies shall be applied, and which order is preferred, is of course user dependent, the default process will take the order exact – plug-in – subsumed – subsumed-by – best.

Afterwards, all matching PServices will be displayed to the user who can choose which Web service(s) shall be invoked. With this selection step, it is ensured that only Web services are invoked that a user trusts, and negotiations about user profile credentials can be controlled by the user if necessary. Afterwards, PServices' customization parameters – if PServices offer them – are displayed to the user who can adjust them according to his requirements.

Every selected and customized PService is executed and returns its content, plus optionally one or several visualization templates. The visualization templates enable the SynServices to reach a high usability by providing domain–optimized visualization. Default visualization strategies are always available by using a RDF browser. The user can interact with the PServices by clicking on links or completing forms in the generated user interface. As these interactions are sent back to the PService it can adapt it's content more precise to the user's requirements and deliver more personalized content, for example displaying a higher level of detail of the relevant informations.

If a PService detects patterns of usage from the user interaction or certain user requirements, it can send an update request to the UMService. Again, not all PServices are allowed to update the user profile, and the update requests are distinguished according to the overall credentials of a PService, and the credentials of the reasoning process used by a PService to interpret user information. The UMService can allow or deny update requests on behalf of the user. Again, the centralized approach shows its benefits, and approved user profile updates are available immediately to not only the current SynService but to all SynServices for further, user-optimized presentation of Web content.

3.2 Visualization and Interface

All user interaction is realized via the Personal Music Syndicator SynService. It provides the interface for searching and configuring Web services, as described above. After Web services were selected, configured and invoked, the SynService displays the results of all PServices which have been invoked and returned results. This separation of content collection and syndication / visualization ensures an easy processing of the PServices' output, and it allow the SynServices to adjust visualization according to user devices' capabilities and limitations, or further user preferences.

By delivering visualization templates, every PService can optimize visualization and usability, as certain domain-specific information can be taken into account for creating the user interface.

4 Proof–of–Concept: Personalized access to Podcasts via the Semantic Web

As a proof–of–concept of our approach, we present the *MyEar Syndication Service*, our current prototype of a Personal Music Syndicator, which provides recommendations for music podcast:

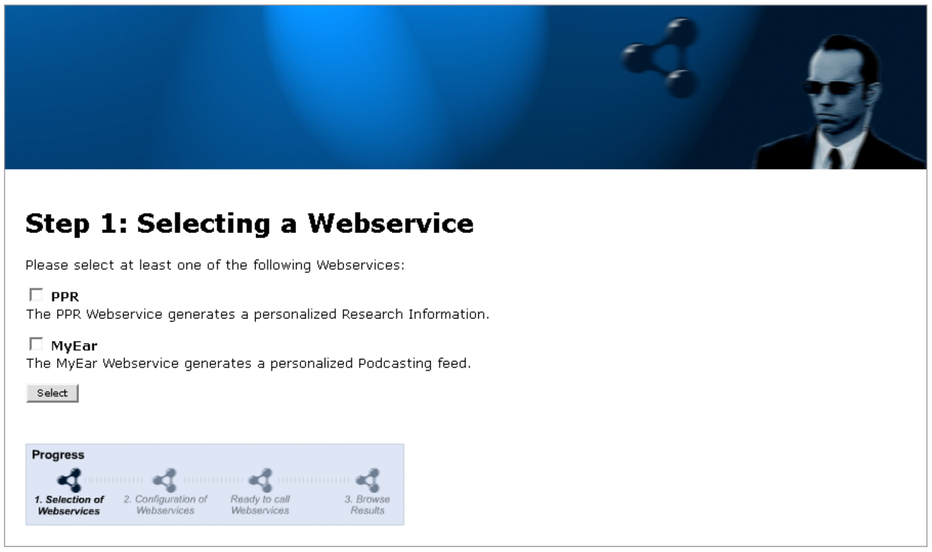
Assume a user who searches for podcasts in the Web. He enters a query and receives a list of appropriate podcast delivery services. He specifies which of these services he wants to launch. The user gets a list of all mandatory and optional parameters which can be used to tailor the services – the MyEar Syndication Service tries to fill all these parameters according to the information it has about the user's preferences. The user can change or simply approve these parameters, eventually the user is requested to enter information that the MyEar Syndication Service was not able to provide. Finally, the user gets the syndicated output of all the services he launched, displayed in his personal Web interface. The appropriate visualization is chosen with respect to the currently used display device of the user.

4.1 Current State of Implementation

In our vision, the user can select detected and appropriate PServices which match his query. As we currently lack a sufficient amount of PServices, the procedures which later will execute the matching process between the user request and the PServices is simply returning all available PServices. The user selects which Web services to use, as depicted in Figure 2).

In the second step, the user can configure selected PServices. For example, the MyEar Syndication Service allows the user to specify keywords, duration and iTunes category of the podcasts he wants to listen to. The description of these customization parameters is provided by the PServices. The user profiling which enables the automatic configuration of the PServices is at the moment a simple one: It stores the parameters the user has entered the last time he used this Web service, and returns them as the default selection in the configuration dialog (figure 3).

After configuration, the MyEar Syndication Service is invoked with the specified parameters. This invocation is passed - via the connector - to the corresponding PServices and MyEar receives the determined content (coded as RDF),


Step 1: Selecting a Webservice

Please select at least one of the following Webservices:

☐ PPR
The PPR Webservice generates a personalized Research Information.

☐ MyEar
The MyEar Webservice generates a personalized Podcasting feed.

Select
Progress
1. Selection of Webservices
2. Configuration of Webservices
Ready to call Webservices
3. Browse Results

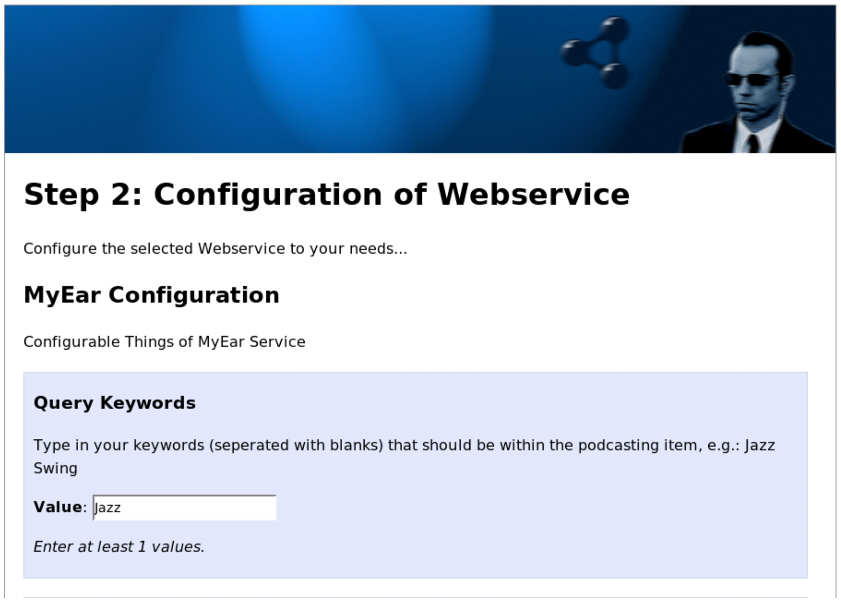
Fig. 2. Dialog for Selecting Personalization Services

as well as visualization templates. Only one visualization template is currently available, which displays the RDF document on PCs within a Web browser, as can be seen in figure 4. The possibility to provide visualization templates by the PServices allows for domain-specific optimization of the user interface, which is not realizable with general-purpose RDF browsing approaches. In the case of the MyEar Syndication Service, for example drag and drop operations are available for selecting podcasts and controlling the audio together with further, music domain-specific gadgets.

4.2 Discussion

A very important issue for improving the usability of Semantic Web browsers is adequate and intuitive visualization of content and it's access. A common approach in today's Semantic Web browsers is to visualize raw RDF files while disregarding the domain and purpose of the RDF document. Therefore, usability of such domain independent Semantic Web browsers is weak. Other approaches focus on some specific domain, and may obtain high usability for this single domain. This procedure has the disadvantage that domain spanning-browsing or domain-spanning content processing is not possible. Consequently, the user has to switch browsers whenever he switches between domains. Synergy effects based on a single user interface and domain spanning user profiling cannot be achieved.

Our approach combines domain specific visualization on the one hand, and a single user interface creation and user profile maintenance on the other. There-


Step 2: Configuration of Webservice

Configure the selected Webservice to your needs...

MyEar Configuration

Configurable Things of MyEar Service

Query Keywords

Type in your keywords (seperated with blanks) that should be within the podcasting item, e.g.: Jazz Swing

Value: Jazz

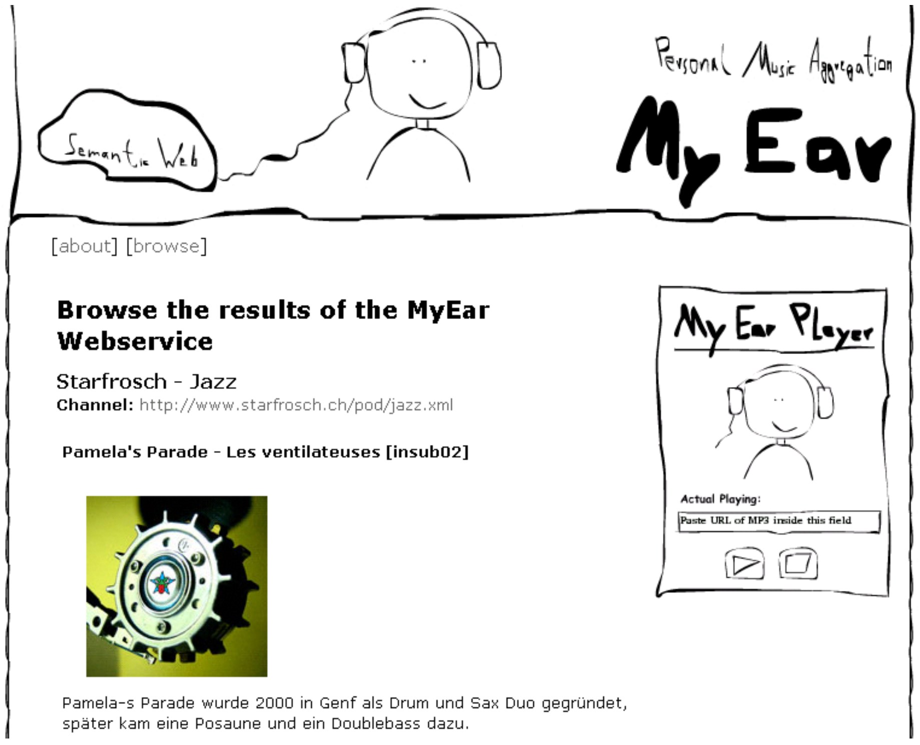
Enter at least 1 values.

Fig. 3. Configuration of the MyEar Syndication Service

fore, we out-source visualization from the Semantic Web browser to the PServices by letting the PServices specify domain-specific and optimized visualization strategies, available via visualization templates. Visualization templates are used by SynServices for creating the final user interface, and – additionally – can optimize the user interface to match the constraints of the currently used device and user specific preferences. To ensure that also results of PServices, which do not provide some visualization template, can be displayed, each SynService implements a generic visualization of RDF documents based on a RDF browser like Piggy Bank[3]. With this dual approach, a domain-optimized user interface will be provided whenever possible, and, as a fall-back solution, general-purpose RDF visualization is possible. This approach has the advantage that domain modeling has not to be done twice (in the Semantic Web browser and in the PServices); furthermore, changes in the output or the visualization of the PService do not require changes in the Semantic Web browser. The user directly interacts with the visualization from the PService, thus PService-specific interactions are enabled, too.

A further advantage of the architecture of the Personal Readers is that access to the user profile can be restricted to trustworthy instances only. This enables high usability as the same information do not need to be given manually to every single Web service, and privacy protection strategies can be facilitated under the full control of the user. By enabling user profile updates from PSer-

[3] http://simile.mit.edu/piggy-bank/



Browse the results of the MyEar Webservice

Starfrosch - Jazz
Channel: http://www.starfrosch.ch/pod/jazz.xml

Pamela's Parade - Les ventilateuses [insub02]

Pamela-s Parade wurde 2000 in Genf als Drum und Sax Duo gegründet, später kam eine Posaune und ein Doublebass dazu.

Fig. 4. Visualization of the MyEar Syndication Service

vices, domain-specific user modeling techniques can and should be implemented in the PServices. The very same argument as in the case of visualization also applies here: domain-specific information should be maintained by those parties which naturally have access to it, other approaches always require duplication or at least exchange of domain models which is error-prone, and at least time consuming.

5 Conclusion and Future Work

We have presented our idea for personalized browsing of Semantic Web data, and presented – as one development of the Personal Reader Project – the *MyEar Music Syndication Service*. Our approach to a Semantic Web browser offers domain-spanning and personalized access to Web services in the Semantic Web. We have enhanced Web services to deliver – beside content – customization and visualization information. Our approach enables domain-optimized usability as user interfaces can be designed on a per Web service base. Combined with a generic RDF browser this approach is simultaneously general and domain-specific, allowing the use of a single user interface and a single user profile to access distributed and personally syndicated Web content.

Our future work will focus on adding more Web services offering personalization for testing the proposed matchmaking strategies within a real world setting.

6 Acknowledgment

We would like the Personal Reader Team at the University of Hannover, and especially Fabian Abel for his engagement in developing the MyEar Syndication Service for the Personal Reader project. This work has been partially supported by the European Network of Excellence "REWERSE - Reasoning on the Web with Rules and Semantics".

References

1. Abel, F., Baumgartner, R., Brooks, A., Enzi, C., Gottlob, G., Henze, N., Herzog, M., Kriesell, M., Nejdl, W., and Tomaschewski, K. The personal publication reader, semantic web challenge 2005. In *4th International Semantic Web Conference* (nov 2005).
2. Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web. *Scientific American* (May 2001).
3. Brusilovsky, P. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction 11* (2001), 87–110.
4. Gao, Z., Qu, Y., Zhai, Y., and Deng, J. Dynamicview: Distribution, evolution and visualization of research areas in computer science. In *Proceeding of International Semantic Web Conference* (2005).
5. Hartmann, J., and Sure, Y. An infrastructure for scalable, reliable semantic portals. *IEEE Intelligent Systems 19*, 3 (2004), 58–65.
6. Henze, N., and Kriesell, M. Personalization Functionality for the Semantic Web: Architectural Outline and First Sample Implementation. In *1st International Workshop on Engineering the Adaptive Web (EAW 2004)* (Eindhoven, The Netherlands, 2004).
7. Henze, N., and Nejdl, W. Adaptation in Open Corpus Hypermedia. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems 12* (2001).
8. Klusch, M., Fries, B., and Sycara, K. Automated semantic web service discovery with owls-mx. In *Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)* (2006), ACM Press.
9. McIlraith, S., Son, T., and Zeng, H. Semantic web services. *Intelligent Systems 16*, 2 (2001), 46–52.
10. Motta, E., Domingue, J., and Cabral, L. Irs-ii: A framework and infrastructure for semantic web services. In *Proceedings of the 2 Intl. Semantic Web Conference* (Florida, USA, 2003).
11. Quan, D., and Karger, D. How to make a semantic web browser. In *Proceedings of the 13th International Conference on World Wide Web* (2004), pp. 255–265.
12. Shadbolt, N. R., Gibbins, N., Glaser, H., Harris, S., and schraefel, m. c. CS AKTive space or how we stopped worrying and learned to love the semantic web. *IEEE Intelligent Systems 19*, 3 (2004).