Personalized Faceted Navigation in the Semantic Web*

Michal Tvarožek, Mária Bieliková

Institute of Informatics and Software Engineering,
Faculty of Informatics and Information Technologies,
Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava, Slovakia
{Name.Surname}@fiit.stuba.sk

Abstract. This paper presents the prototype of an adaptive faceted semantic browser – Factic. Factic implements our novel method of navigation in open information spaces represented by ontologies based on an enhanced faceted browser with support for dynamic facet generation and adaptation based on user characteristics. It is developed as part of a modular framework that supports personalization based on an automatically acquired ontological user model. We describe software tool design and implementation together with discussion on several problems mostly related to the general immaturity of current Semantic Web solutions.

1 Concept overview

The Semantic Web as envisioned by Tim Berners-Lee aims to solve some problems of the current Web related to its constant change and growth by incorporating shared semantics i.e. "meaning" thus e.g. significantly improving interoperability between systems [1]. Although this idea was proposed several years ago, it still remains largely unrealized due to various reasons such as the lack of standards or appropriate software tools. As the need for shared semantics and (web) data integration grows, the demand for common conceptualizations referred to as ontologies is also increasing. Some authors argue that the use of ontologies in the e-science community presages ultimate success for the Semantic Web [1].

Consequently, proper software tools for navigation in the Semantic Web i.e. for navigation in ontologies (e.g., RDF/RDFS, OWL) are required. While these will include new types of tools, adding support for ontologies to "classical" tools is also imperative as these are already widely used in different scenarios. Examples of existing tools include search engines, web portals or faceted browsers [2].

Furthermore, the size and changeability of the Web and consequently the Semantic Web together with their diverse user base make them prime candidates for adaptive web-based systems that take advantage of (automatic) user adaptation in order to increase overall effectiveness, productivity and user orientation.

* This work was partially supported by the Slovak Research and Development Agency under the contract No. APVT-20-007104 and the State programme of research and development under the contract No. 1025/04.

The concept of the adaptive faceted semantic browser was proposed in [3]. As such, it is an enhanced faceted browser with support for:

- Ontological representation of the application domain by means of a domain ontology (e.g., in OWL).
- Logging of user actions with semantics within the browser as defined by an event ontology. The created user action logs are subsequently used for automatic user modeling, which is performed by external user modeling tools [4].
- Personalization based on an ontological user model derived from the domain ontology and created and maintained by the aforementioned user modeling tools. The personalization includes the adaptation of facets, facet restrictions and search results as well as the recommendation of relevant concepts.

Based on its properties, the adaptive faceted semantic browser is suited for effective viewing and navigating in large open information spaces represented by an OWL ontology. It can also be used as an information retrieval tool where the search query is visually created by means of navigation – selecting restrictions in the set of available facets, which are dynamically adapted to users' needs.

2 Faceted browser design and implementation

We designed and implemented the adaptive faceted semantic browser in the form of a software tool called *Factic*. It is a presentation tool that allows users to navigate and search in an information space represented by an OWL ontology. Therefore, we integrated *Factic* into the personalized presentation layer [5] of a web-based information portal [6] (see Figure 1).

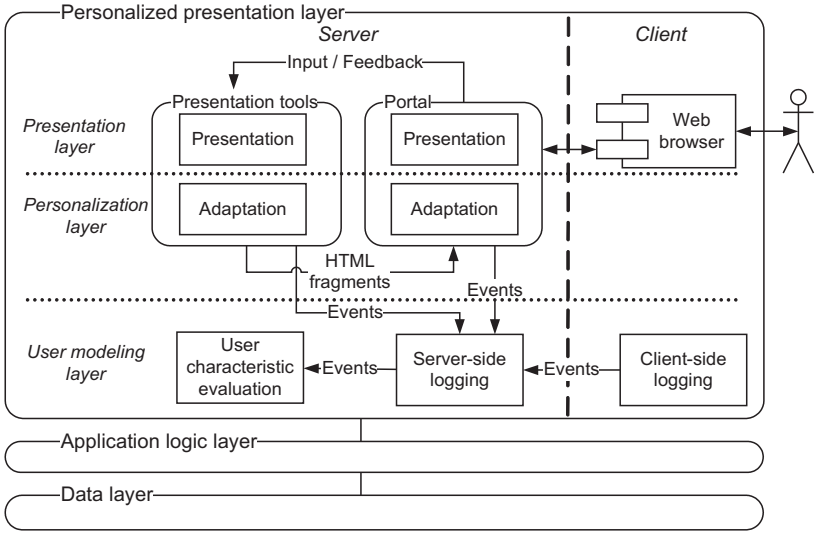

Fig. 1. Architecture of the personalized presentation layer of a web-based portal.

The *Factic* presentation tool is depicted on the top left and can be divided into two parts – the presentation and the adaptation part. As input, *Factic* takes user input/feedback from the Portal module, to which it also sends the results of its processing in the form of (X)HTML fragments. The portal serves for the integration of individual presentation tools (e.g., *Factic*) and acts as a proxy towards the client web browser depicted on the right.

Presentation tools as well as the Portal tool perform user action logging with semantics be means of the user modeling layer depicted at the bottom, which performs both client-side and server-side logging and user characteristic analysis.

Factic is implemented in Java as an *Apache Cocoon* coplet (i.e. Cocoon portlet) and uses *Sesame* to store ontological data, *MySQL* to store relational data and as a back-end for *Sesame*. For the logging service we use *Apache Axis* as a web service container and *Apache Tomcat* as a servlet container.

Cocoon is based on XML and the pipes and filters architectural pattern where every request is processed by a given pipeline. Each pipeline consists of a single generator, zero or more transformers and a serializer. *Factic* itself as a *Cocoon* generator takes full advantage of its XML processing capabilities.

Figure 2 depicts the design and request processing of the *Factic* tool, which employs a two-step transform view, where the initial logical XML output description is transformed by a set of XSL transformations into the final XHTML document (top) and sent to the client web browser (right). Individual user requests are handled as described by the Sequence 1.

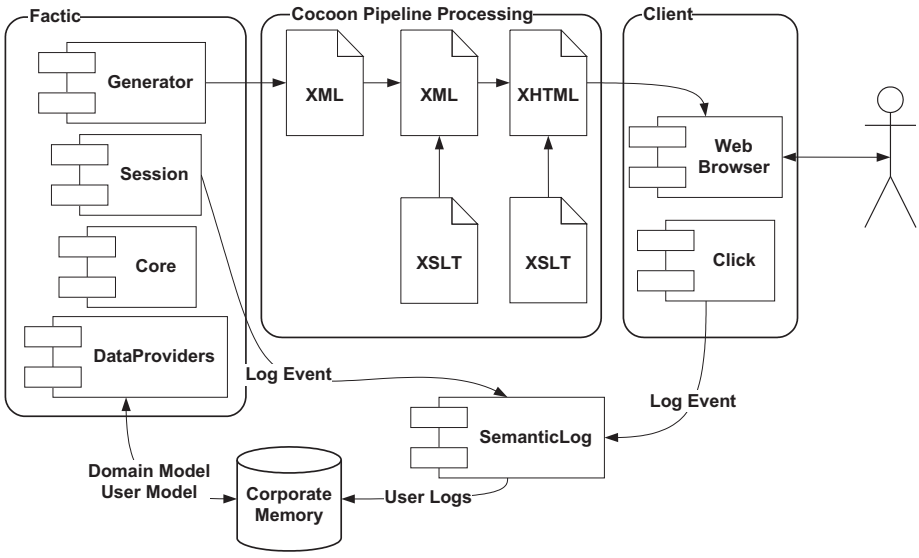

Fig. 2. Design of the Factic presentation tool.

Sequence 1 HandleRequest *Input*: URL request *Output*: XHTML response

1. *Session*:	`Preprocess request, update session state`
2. *Core*:	`Process request, create and execute query`
3. *DataProviders*:	`Retrieve domain and user data`
4. *Core*:	`Process results, evaluate adaptation and annotation`
5. *Session*:	`Log event via the SemanticLog logging service`
6. *Generator*:	`Generate logical output description in XML`
7. *Cocoon*:	`Transform XML output to formatted XHTML response`

3 Discussion and conclusion

We evaluated *Factic* in the domain of online job offers[1] and in the domain of scientific publications[2]. In both cases, we used *Factic* for the presentation of and navigation in the respective domain ontology as well as for information retrieval.

Figure 3 shows the experimental results that we achieved. The time and number of clicks represent the total user effort that was necessary to complete a given scenario i.e. to find a certain set of ontological instances. The results indicate that adaptive selection of active facets can significantly improve total processing time, which depends linearly on the number of displayed facets. Furthermore, recommendation of suitable ontological concepts based on the user model further reduces the number of necessary clicks as well as overall task completion time.

While the results that we achieved are promising we also encountered several problems in the form of several performance bottlenecks:

- The logging of user actions together with the display state of the GUI via web services was very slow because a lot of data had to be serialized/deserialized via SOAP. We solved this by using a hybrid logging approach where some of the data are logged via web services (e.g., client-side logging) and some data are logged directly by means of an API (e.g., display state of the GUI).
- The cost of ontological queries is high and consequently, the processing of ontological queries is slow. We were unable to resolve this problem although we improved overall performance by caching data in *Factic*. Furthermore, the ontological repository *Sesame* is rather immature – it is slow, unoptimized and contains several bugs, which prevent correct evaluation of queries.
- SeRQL – the recommended query language for *Sesame* and thus *Sesame* lack several important features such as COUNT() or ORDER BY. These must thus be emulated by our application which further reduces performance.

The primary advantage o‘f our approach lies in the use of ontologies. The shared conceptualization provided by ontologies improves tool interoperability. E.g., our automatic user modeling is not hard-coded to the output of specific presentation tools nor does it require extensive preprocessing as seen in traditional analysis of web server logs. Furthermore, the stored semantics of user actions are directly used in the user modeling process to estimate user characteristics.

[1] NAZOU Project, `http://nazou.fiit.stuba.sk`
[2] MAPEKUS Project, `http://mapekus.fiit.stuba.sk`

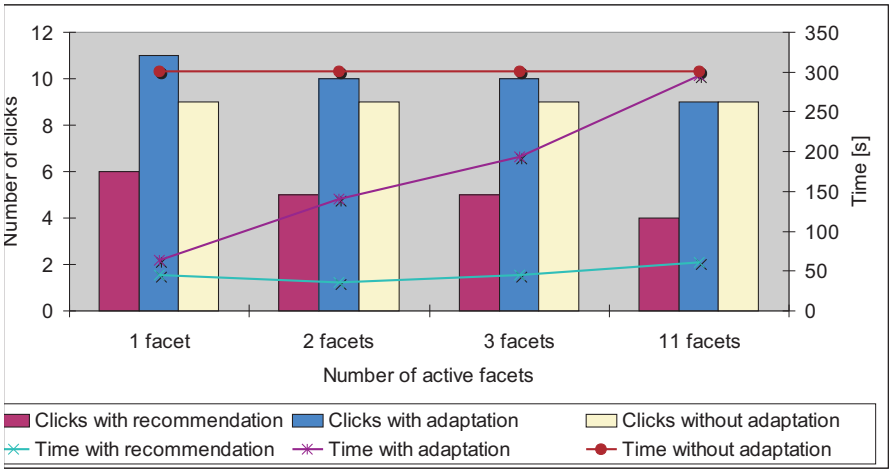

Fig. 3. Evaluation results for different adaptation modes (non-adaptive, with adaptation, with recommendation).

The acquired user characteristics are used in the adaptation process to improve efficiency and overall user experience without the need for direct conscious user involvement. As a result, the user can focus on the tasks at hand without the need to perform tedious system and/or user model settings.

Lastly, the *Cocoon* presentation framework allows us to easily integrate additional presentation tools as well as to further customize the processing pipeline by using additional transformers. Thus future work will include the integration with additional presentation/navigation/information retrieval tools and the implementation of new adaptation functions, such as dynamic facet generation.

References

1. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. IEEE Intelligent Systems **21**(3) (2006) 96–101
2. Instone, K.: How user interfaces represent and benefit from a faceted classification system. In: SOASIST. (2004)
3. Tvarožek, M.: Personalized Navigation in the Semantic Web. In V. Wade et al., ed.: Proc. of 4th Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems, AH'06, Springer, LNCS 4018 (2006) 467–471
4. Andrejko, A., Barla, M., Bieliková, M.: Ontology-based User modeling for Web-based Information Systems. In: ISD 2006, Budapest, Springer Verlag (2006)
5. Tvarožek, M., Barla, M., Bieliková, M.: Personalized Presentation in Web-Based Information Systems. In J. van Leeuwen et al., ed.: Proc. of SOFSEM 2007, Springer, LNCS 4362 (2007) 796–807
6. Barla, M., Bartalos, P., Sivák, P., Szobi, K., Tvarožek, M., Filkorn, R.: Ontology as an Information Base for Domain Oriented Portal Solutions. In: ISD 2006, Budapest, Springer Verlag (2006)