The OntoManager – a system for the usage-based ontology management

Nenad Stojanovic[1], Jens Hartmann[1], Jorge Gonzalez[3]

[1] Institute AIFB, University of Karlsruhe,
76128 Karlsruhe, Germany
{nst,jha}@aifb.uni-karlsruhe.de
[3] SAP AG*
69190 Waldorf, Germany
jorge.gonzalez@sap.com

Abstract

In this paper, we propose an approach for guiding ontology managers through the modification of an ontology with respect to users' needs. It is based on the analysis of end-users' interactions with the ontology-based applications, which are tracked into the usage-log. We proposed two types of the analyses: the ontology evolution and the instance crawling, which lead to the improvement of the structure of the ontology and the expansion of the knowledge base, respectively. The approach has been implemented in the system called OntoManager. We present here the conceptual architecture of OntoManager.

1 Introduction

In an ontology-based information portal ontologies support the process of "indexing" content of an information resource – so called semantic annotation and the navigation through the knowledge repository – so called conceptual navigation. However, ontologies, as a conceptual model for the given business domain, should react to all changes in the business environment. This includes accounting the modification in the application domain or in the business strategy; incorporating additional functionality according to changes in the users' needs; organizing information in a better way etc. If the underlying ontology is not up-to-date or the annotation of knowledge resources is inconsistent, redundant or incomplete, then the reliability, accuracy and effectiveness of the system decrease significantly [Stojanovic et al., 2002a] . In order to avoid these real problems, ontology-based applications have to be supported by a mechanism for the discovering of these changes, analyzing and resolving them in a consistent way [Stojanovic and Stojanovic, 2002].

We have developed such an approach for ontology management and implemented it in the ***OntoManager*** tool. It concerns the truthfulness of an ontology with respect to its problem domain - does the ontology represent a piece of reality and the users' requirements correctly? Indeed, it helps to find the "weak places" in the ontology regarding the users' needs, ensures that generated recommendations for the ontology improvement reflect the users' needs, and promotes the accountability of managers. In this way, the ***OntoManager*** provides an easy-to-use management system for ontologists, domain experts, and business analysts, since they are able to use it productively, with a minimum of the training. As known to the authors, none of the existing ontology management systems offer support for (semi-) automatic ontology improvement in response to the users' needs analysis.

This paper is organised as follows: Section 2 describes the conceptual architecture of our approach. In section 3, we elaborate the modules of the ***OntoManager*** enabling the integration, visualisation and analysis of the users' needs regarding the domain ontology. After a discussion of related work, concluding remarks outline some future work.

2 The conceptual architecture – the MAPE model

Our management system is realised according to the **MAPE** (**M**onitor **A**nalyse **P**lan **E**xecute) model [Kephart and Chess, 2003], which abstracts a management architecture into four common functions: collect the data, analyse the data, create a plan of action, and execute the plan. Indeed, our architecture decomposes the control loop into four parts:

- **Monitor** – mechanism that collects, organises and filters the data about users' interactions with the ontology-based application;
- **Analyse** – mechanism that aggregates, transforms, correlates, visualises the collected data, and makes proposals for changes in the ontology;
- **Plan** – mechanism to structure actions needed to apply the discovered changes by keeping the consistency of the ontology. The planning mechanism uses evolution strategies [Stojanovic et al., 2002a] to guide its work;

* This research was carried out while the author was with the Institute AIFB, University of Karlsruhe

- **Execute** – mechanism to update the underlying ontology-based application according to the changes applied in the ontology.

By monitoring (**M**) the behaviour of users and analysing (**A**) this data, planning (**P**) which actions should be taken and executing (**E**) them, a kind of a "usage loop" is created.

Figure 1 depicts this "usage loop" in an information portal scenario. A user is searching for information by querying and/or navigating through the portal (cf. 1 in Figure 1). All activities the user performed are acquired in the Semantic Log (cf. 2), which is structured according to the Log Ontology, and contains meta-information about the content of visited pages [Stojanovic et al., 2002a] . The process of tracking the users' activities is elaborated in 0. This log data is aggregated and visualised in the ***OntoManager*** (cf. 3). Moreover, the ***OntoManager*** helps ontology managers discover changes in the ontology, which are mostly important for enhancing the usability of the application. The architecture of the ***OntoManager*** is described in the next section in more details. Since the application of a single ontology change can cause the inconsistency in the other part of this ontology and all the artefacts that depend on it [Meadche et al., 2003], we applied the ontology evolution process (cf. 4) that guaranties the transfer of the ontology and dependent artefacts into another consistent state. Moreover, in the case of creating a new concept, OntoCrawler can be started to complete that concept with the most promising instances that can be found in an intranet (or in Internet, in general).

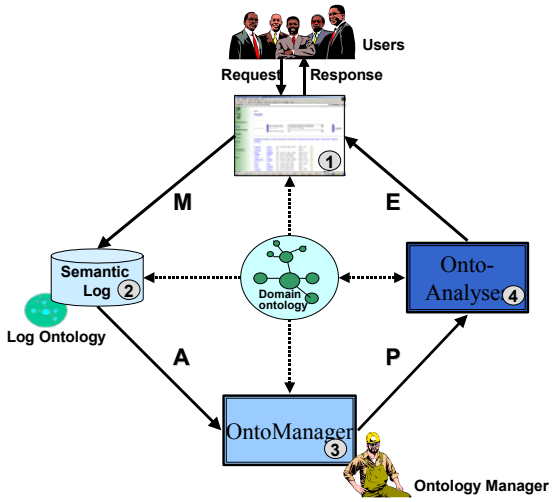

Figure 1 The conceptual architecture of the ontology management system according to the MAPE model

Finally, since the underlying application is ontology-based, all changes in the ontology are reflected on the structure of the portal (cf. 1), by tailoring the portal to the users' needs, which implicitly arose. For example, if none of users were interested in a topic, then the ***OntoManager*** can recommend the ontology manager to remove the corresponding concept from the topic hierarchy. Consequently, new users will be not "bored" by browsing topics, which are useless for the domain shown in the portal. In that way, our management system aims to be a user-friendly platform that integrates the results from the analysis of the usage data with the tools that guide the process of modifying the ontology.

3 OntoManager

The ***OntoManager*** has been designed to provide the methods and tools that support the ontology managers in managing and optimising the ontology according to the users' needs. This system incorporates mechanisms that assess how the ontology (and by extension the application) is performing based on different criteria, and then enable to take action to optimise it.

One of the key tasks is to check how the ontology fulfils the perceived needs of the users. In that way, we obtain an in-depth view of the users' perspective on the ontology and the ontology-based application, since on the top of this ontology the application is going to be conducted. The technique that can be used to evaluate/estimate the users' needs depends on the information source. By tracking users' interactions with the application in a log file, it is possible to collect useful information that can be used to assess what the main interests of the users are. In this way, we avoid asking the users explicitly, since they tend to be reluctant to provide the feedback via filling questionnaires or forms.

In the rest of this section, we firstly describe the inputs into the ***OntoManager,*** and afterwards the structure of the ***OntoManager*** itself.

3.1 Inputs

The ***OntoManager*** has two inputs: the domain ontology that is the backbone of the whole system and the Semantic Log (i.e. Semantic Logs in case a portal is distributed on various web servers, see section 3.2.1).
The description of the model of the ontology we use can be found in [Stojanovic, 2003]. Here we present only the Semantic Log.

Semantic Log
Traditionally, all the activities of users of a portal are captured in the standard web server log. However, the standard web server log fails in the case of reloading pages from cash and spider's crawling, which cause missing, redundant or incomplete data. Moreover, this log file contains only the information about the address of visited pages, which was not enough for the sophisticated analyses we intended to perform. For example, it is not possible to get the information about which query is posted and how many and which results are retrieved. What we need is the information about the semantics of the visited pages. In order to resolve these problems, we do not use traditional web server logs, but rather the application-based logging, so that each user's activity is captured on the level of the application 0. To enhance the quality of the logged information, we have developed the *Log Ontology* as the backbone for structuring information in a *Semantic Log*. Each user's activity is captured in a *Semantic Log* in the form of instances of the *Log Ontology*.

A part of the *Log Ontology* that is relevant for the rest of this paper is presented in 0a. This ontology models what happens in the portal and why, when, by whom, how it is performed. Each user's activity is represented as an instance of one of the subconcepts of the "*Event"* concept.

The structure of the hierarchy of event types reflects the users' activities in an ontology-based portal by including all possible types of interactions (e.g. "*Query*", "*Browse*", "*Read*", etc.). Some additional information, such as the "*date*" and "*time*" of the activity, as well as the identity of the user may be associated through appropriate relations. The information enabling the support for the users' profiling, such as "*sessionID*", "*clientIP*" etc. may also be included. Entities from the domain ontology are related to instances of the "*Event*" concept through the "*relatedTo*" relation. The dependency between events is represented using the "*previousEvent*" relation.

0b shows several users' activities stored in the *Semantic Log*. They are the result of the user's request for "*Projekt*" and successively browsing activities through the concept "*Project*" and its subconcept "*EUProject*". The instance "*Query100*" captures all important information regarding the query activity, whereas the instances "*Broswe123*" and "*Browse456*" correspond to the navigation activities through the hierarchy of the concept "*Project*". Note that "dom#" denotes the namespace of the domain ontology.

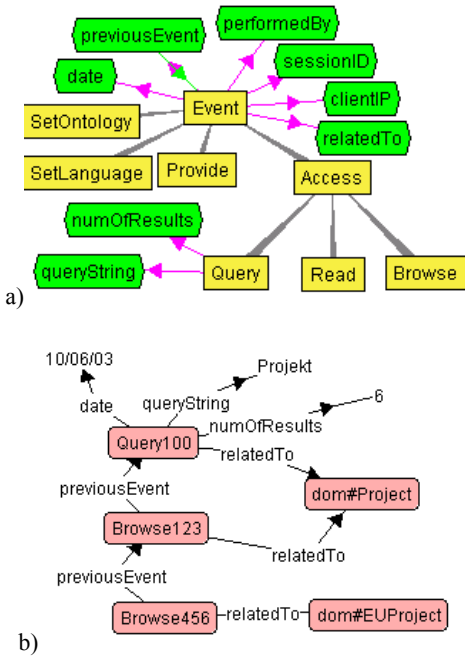

Figure 2. A part of the Log Ontology and the Semantic Log. (a) The conceptual structure of the Log Ontology is represented in the left part. (b) The right part shows several log entries in the form of relation instances.

3.2 Components

Conceptually, the ***OntoManager*** consists of three modules:

- The Data Integration Module that aggregates, transforms and correlates the usage data;
- The Visualisation Module that makes the integrated usage data more useful for human beings by presenting the data in a comprehensible visual form;
- The Analysis Module that provides guidance for adapting the ontology with respect to the users' needs.

Subsequently, we describe these three modules in detail.

Data Integration Module

The *Data Integration Module* has three main functionalities:

- to *collect* data from different, possibly distributed logs in case an ontology-based application is deployed on several web servers;
- to *pre-process* data by transforming disparate data into meaningful information. This phase also covers the cleaning and validation of the data for achieving the required quality;
- to *organise* them in a way that enables a fast and efficient access to the data.

In order to integrate data from various servers, we replicate the *Semantic Logs* of all these servers into a "common" log, so called *OntoLog*. Since all logs are based on the *Log Ontology* and they reference the same domain ontology, the semantic heterogeneity problem doesn't occur. Another possibility for the integration was to integrate the logs virtually (on-the-fly) by accessing them in the time of processing. Such a solution would enable the immediate visibility (actuality) of log data in the ***OntoManager***, but it requires extensive distribute processing and, thus, it is slow and expensive. Since the analyses we want to perform are statistic-based, the actuality of the data is not so critical. However, the update of the *OntoLog* is performed periodically (currently once per week).

Moreover, during this phase, the data is also pre-processed, in order to make it better suited for the further analysis. We perform two types of data pre-processing:

1. **Data abstraction -** Since the interaction of the users with the portal is mainly done on the level of ontology instances, the Semantic Logs (and consequently the *OntoLog*) mainly contain the information about the usage of ontology instances. For example, if a user has seen more details about the project "*SemIPort*", the log file recorded this information explicitly. However, the goal of our system is to improve the ontology and not its knowledge base. Thus, all log entries regarding ontology instances have to be transformed into corresponding ontology concepts. Regarding the previous example, all the appearances of the instance "*SemIPort*" in the *OntoLog* have to be replaced with the concept "*Project*";
2. **Extracting links -** the most important information for the analyses we want to perform is the frequency of browsing relations between two concepts (see section 3.2.3). Since the *OntoLog* does not contain explicit information about the source and the target of a browsing event[1], we extract it in this pre-processing phase by analysing successive events. For example, regarding the part of the log presented in 0b, from two successive browsing events ("*Browsing123*" and "*Browsing456*") our system concludes that the link between concepts "*Project*" and "*EUProject*" was browsed, since the first event is related to the concept "*Project*", and the second one to the concept "*EUProject*".

[1] The *Log Ontology* models the dependency between events through the relation "*previousEvent*" (see 0a).

Finally, the integrated and pre-processed data has to be analysed, in order to enable the ontology manager to manage the ontology efficiently. However, with increasing frequency of the application usage, the log might contain a large quantity of data. Thus, it has to be reengineered, to enable ontology managers to perform sophisticated data analysis through a fast access to a variety of possible views of the underlying information. Further, in order to get a fast response, it would be useful to pre-calculate at least some of the information that will be needed for analysis. Since OLAP techniques [Kimball and Merz, 2000] typically handle huge volumes of data that is interrelated in complex ways, and enable the pre-calculation of everything that may be needed, we decided to transfer the log into *OLAP cubes*. In this way, the *OntoLog* only contains the pre-processed information about the users' interactions, which are needed to improve the ontology, whereas the *OLAP cubes* enable the analysis of this information at an aggregate level.

Indeed, an OLAP cube as a part of the ***OntoManager*** performs various in-advance analyses, in order to speed up the decision making process. The most important data is the number of browsing[2] the *direct hierarchy relation* between two concepts c1 and c2 (denoted as Usage(c1,c2)) and the number of querying[3] for a concept c (denoted as Querying(c)). By processing the *OntoLog,* these values increase. For example, by processing the part of the Semantic Log presented in 0b, the value of Usage("*Project*", "*EUProject*") and the value of Querying("*Project*") will be incremented. See section 3.3 for more information about the analyses we perform.

Due to the lack of space, we omit here the detailed description of the OLAP cube. In the current implementation, the OLAP cube is queried via a web service. An advantage of using web service is that it enables having a thin client that can access the OLAP data in a remote server without threatening the security of the server.

The Visualisation Module

Since "information visualisation is the use of computer-supported, interactive, visual representation of abstract data to amplify cognition" [Card et al., 1999], the graphical representations of the ontology-usage data can help the ontology manager adapt an ontology with respect to the users' needs.

In order to achieve that, the *Visualization Module* combines graphically (transparently and intuitively) the integrated ontology usage data with the ontology itself. Besides, it enables the representation of different aspects of the underlying information. Finally, it allows for easy and flexible presentations of the same information in different ways. By showing different aspects of the underlying information and in different ways (from one or more perspectives), the visualisation mechanisms offer support for analysis tasks.

The presentation of the results of analysis in the form of tables, histograms, charts or other easily comprehensive ways can increase the understanding of the usability of the ontology entities. On the other way, the requirements put on visualisation can considerably vary with different analytical tasks. Thus, the *Visualisation Module* presents information in several different ways:

- **Graph-based representation** of the ontology (see the left part of the screenshot shown in 0), where nodes correspond to the concepts in the ontology, and links correspond to the direct hierarchy (see Definition 3). It enables:
  - easy manipulation with large ontologies. A lot of visual features are implemented in the current version: focus on the part of the ontology (zoom, anti-zoom), rotating the nodes and lines around a selected node, adapting the number of hierarchical levels in the ontology presented on the screen (locality), tracking the path followed to reach the current selected node, the existence of back and forward button to repeat actions;
  - efficient inspection of various "problems" which can be found in an ontology, i.e. not-used concept, very sought concepts. In this version, a suitable colouring is performed as an indicator of the frequency of using an ontology concept and its relations;
- **Table-based presentation** of the results (see the right part of the screenshot represented in 0). It enables a comprehensible two-dimensional view on the data, and supports very fast sorting of data;
- **Bar-based presentation** (e.g. histogram, Pareto diagram, etc.) that shows several measures at the same time by means of a vertical bar. For example, Pareto diagram[4]) enables a very easy detection of the most important concepts, e.g. concepts that take the most of the users' attention.

The application of these visual metaphors supports discovering patterns, trends in the ontology usage data, and, consequently, leads to the new insights into the ontology. Indeed, this module digests the result of the data integration modules and produces the summary reports easily readable by the ontology managers. The added value of our visualisation lies in its **expressivity**. For example, it is very easy to detect unused concepts. In addition, the correlation between two concepts is immediately apparent. The Pareto diagram can show which concepts take the useful information and which can be treated as useless.

Figure 3. illustrates some of the above-mentioned functionalities of the ***OntoManager***. The content is taken from our evaluation study. This screenshot presents the inspection of the concept "*Project*" (cf. 1) and its subconcepts "EUProject", "*RegionalProject*" and "*NationalProject*". The upper left part shows the hyperlinked path of the concept (i.e. tracking). On the left panel, a graphical representation of the ontology is presented, enabling an ontology manager to traverse/navigate the ontology by clicking on the nodes. When the information about the usage of the selected concept is required, a query is submitted to the OLAP via

[2] Browsing is treated as a click on the hyperlink between two concepts that are in a direct hierarchy relation.

[3] The number of queries related to a concept.

[4] According to the Pareto principle, by analysing 20% of most frequently used data 80% problems in the ontology can be eliminated.

a right-click button menu. The information about the number of visitors, visits and times that the sub-concepts have been accessed is presented in the right-most panels. The ontology manager can select between querying and navigation data. The label colour of the nodes will then change according to some user-defined rules (darker colour indicates more visits in Figure 3). The coupling between the structure of the ontology and the aggregated data enables visual highlighting when an entity has not been accessed at all (e.g. "*RegionalProject*" (cf. 2 in Figure 3)).

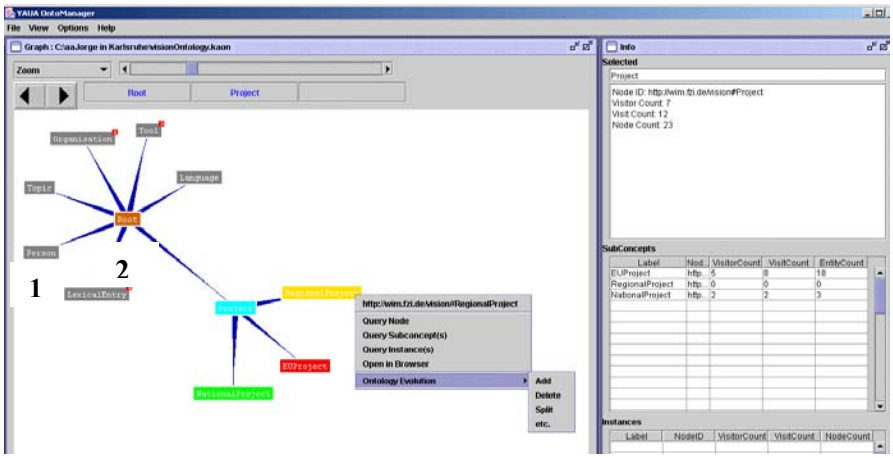

Figure 3. The Visualisation Module in the OntoManager

The Analysis Module

Whereas the Visualisation Module provides the ontology managers with convenient representations of underlying data, the *Analysis Module* suggests them how to improve the ontology. The basic assumption of our system is that management of an ontology should be guided by the real interest of the users. Thus, the ontology should be re-engineered considering this feedback, e.g. adding more granularity to most sought-after concepts by splitting them or grouping less accessed concepts.

In particularly, there are two task of the Analysis module:

- Ontology Evolution that provides guidance in the process of modifying the ontology and ensure the consistency of the updated ontology. This module keeps track of the changes and has the possibility to undo any action taken upon the ontology.

  The OntoManager imports the functionalities related to the ontology evolution process that we elaborated in [Stojanovic et al., 2002a] . We omit here more details.

- Instance Crawling that complete newly created concept with the most promising instances that can be found in an intranet

  Deleting useless concepts in an ontology leads to a new applicable ontology by some steps based on the presented evolution strategies. However, adding new concepts to an ontology leads to a knowledge leak since instances of the new concepts are missed.

  Therefore, OntoCrawler provides semantic capabilities for identifying and extracting new useful instances whereby existing knowledge about concepts, relations and instances can be used as background knowledge for the crawling process. A crawling process results in an ordered set of instances weighted by their assumed semantic relevance. Thus, it is possible to add new concepts within OntoManager to differentiate an actual knowledge state and then to use OntoCrawler to fill these concepts.

  The crawling process [Ehrig, 2002], [Schmitz, 2002] uses the domain ontology which represents specific domain knowledge and a World ontology which describes the environment in which the knowledge is kept. In this example, the World ontology consists of information about hosts, IP-adresses, hyperlinks, etc. In the crawling process new documents are analysed and a semantic relevance for the crawl-task is calculated. This relevance is used to focus the search on potentially highly-relevant instances.

4 Related Work

In [Stojanovic L. et al., 2002b] we made a comprehensive evaluation of most frequently used tools for editing ontologies, Protege[5], OilEd[6] and OntoEdit[7], by comparing them regarding several criteria, including their support for the continual ontology improvement. None of them provides support neither for the integration of the usage data into the ontology evolution process nor for the discovery of changes in an ontology, which are crucial facilities of the ***OntoManager***. Therefore, these capabilities of the ***OntoManager*** are novel in comparison to the existing ontology editors.

Moreover, the ***OntoManager*** is a tool for a comprehensive management of the ontology-based applications, which incorporates the collection, the integration and the analysis of the data needed for the management. In that way, the ***OntoManager*** is a unique tool, since, as known to the authors, such a management tool for ontology-based applications does not exist. However, there are management systems for other types

[5] http://protege.stanford.edu/

[6] http://oiled.man.ac.uk/

[7] http://www.ontoprise.de/com/co_produ_tool3.htm

of the applications, which can be related to our work. For example, an approach for managing changes in a knowledge management (KM) system is given in [12]. The authors consider two types of changes: (i) functional changes that are about new KM-systems in the organization, new versions of a KM-system and new features in one KM-system and (ii) structural changes that deal with new business models, new subsidiaries and new competencies in the organisation. The results of that study show that managing the evolution of KM-systems on an ad hoc basis can lead to unnecessary complexity and KM-systems failures. Both types of changes can be treated as the explicit changes, which can be very efficiently resolved in our system. However, contrary to the ***OntoManager***, this approach does not consider implicit changes, which can be derived from the usage of the system.

5 Conclusion

The possibility to cope with the implicit changes discovered from the users' behaviour seems to be the most important characteristic of an application, which aspires to be useful. Indeed, it enables the continual adaptation of an application to the changes in the users' needs, without demanding the users to provide an explicit feedback about the usability of the application. The most common attribute for discovering changes is the usage of some structures (buttons, options in the menu, etc.), whose analysis enables their fine-tuning to the users' needs.

In an ontology-based application, the domain ontology is used as a conceptual backbone for structuring the domain information provided in the application. Consequently, the data about the usage of the application can be analysed using the ontology as the background knowledge, which alleviates the process of discovering useful changes in the application. The discovered changes lead to the improvement of the ontology, but in the end effect, since the content and layout (structure) of an ontology-based application are based on the underlying ontology, by changing the ontology according to the users' needs, the application itself is tailored to the users' needs.

In this paper, we presented an integrated approach for the usage-based management of the ontology-based applications, which covers capturing and structuring the users' activities with the application, their integration and filtering, then the visualisation of the usage data in the context of the underlying ontology and, finally, the automatic discovery of changes and their systematic resolution by ensuring the consistency of the resulting ontology. The approach has been implemented in the system called OntoManager, a user-friendly platform that integrates the results from the analysis of the usage data with the tools that guide the process of modifying the ontology. The focus of this paper was on the conceptual architecture of the system. The evaluation of the analyses we proposed is out of the scope of the paper. However, we tested the performance of the system, particularly in discovering anomalies in a hierarchy in the domain ontology. This early evaluation study shows the benefits, in time and correctness with respect to ad hoc methods, of supporting the ontology management by our approach.

Acknowledgement

The research presented in this paper would not have been possible without our colleagues and students at the Institute AIFB and the FZI, University of Karlsruhe. Research for this paper was partially financed by BMBF in the project "SemIPort" (08C5939) and by EU in the IST-2000-28293 project "Ontologging".

References

[Stojanovic L. et al., 2002a] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic. *User-driven Ontology Evolution Management*, Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW'02, Madrid, 2002.^

[Stojanovic and Stojanovic, 2002] N. Stojanovic and L. Stojanovic. *Usage-oriented Evolution of Ontology-based Knowledge Management Systems*, Proceedings of the 1st Int'l Conf. on Ontologies, Databases and Application of Semantics (ODBASE-2002), Irvine, CA, 2002.

[Kephart and Chess, 2003] J. Kephart and D. Chess, *The Vision of Autonomic Computing*, IEEE Computer, January 2003., pp. 41-50.

[Stojanovic N. et al., 2002] N. Stojanovic, L. Stojanovic and J. Gonzalez, *On Enhancing Searching for Information in an Information Portal by Tracking Users' Activities*, First International Workshop on Mining for Enhanced Web Search (MEWS 2002), held in conjunction with WISE 2002, Singapore, 2002.

[Meadche et al., 2003] A. Maedche, B. Motik, L. Stojanovic, R. Studer and R. Volz, *Ontologies for Enterprise Knowledge Management*, IEEE Intelligent System, pp. 26-34, March/April 2003.

[Stojanovic, 2003] N. Stojanovic, *On the Query Refinement in the Ontology-based Searching for Information*, the 15th Conference On Advanced Information Systems Engineering, CAiSE'03, Austria, 2003.

[Kimball and Merz, 2000] R. Kimball and R. Merz, *The Data Webhouse Toolkit: Building the Web-Enabled Data Warehouse*, John Wiley & Sons, 2000.

[Card et al., 1999] S. Card, J. Mackinlay and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann, 1999.

[Ehrig, 2002] M. Ehrig, *Ontology-focused Crawling of Documents and Relational Metadata*, Masters Thesis, University of Karlsruhe, 2002

[Schmitz, 2002] C. Schmitz, *Untersuchung der Graphstruktur von Web-Communities am Beispiel der Informatik*, Masters Thesis,University of Trier, 2002

[Stojanovic L. et al., 2002b] L. Stojanovic, B. Motik, *Ontology Evolution within Ontology Editors*, EKAW'02/EON Workshop, Madrid, 2002.

[Hardless et al., 2000] C. Hardless, R. Lindgren, U. Nulden, K., Pessi, *The Evolution of knowledge management system need to be managed*, http://www.viktoria.informatik.gu.se/groups/ KnowledgeManagement/Documents/kmman.pdf, 2000.