Ontology Engineering for Multiway Acquisition of Web Metadata

Vojtěch Svátek[1,2], Jiří Kosek[1], and Miroslav Vacura[1]

[1] Department of Information and Knowledge Engineering,
University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic
svatek@vse.cz, jirka@kosek.cz, vacura@vse.cz
[2] EuroMISE Research Centre – Cardio, Prague, Czech Republic

Abstract. A promising way how to flexibly obtain collections of implicit metadata from heterogeneous and loosely structured web resources is to exploit a distributed architecture consisting of modules specialising in syntactical types of web data: linguistic structures, HTML structures, URLs, explicit metadata, link topology and images. The prototype of such architecture currently under development is based on the web service model, and uses a simple request–answer (SOAP) communication architecture. Similarly to mainstream multi–agent systems, the communication among the modules should respect a shared ontology of the domain. Due to the static character of the architecture, the crucial problem is not the sharing of semantics during the process of problem solving, but rather the (offline) construction of ontologies from multiple datatype–specific domain models. Two partial case studies for this problem are provided.

1 Introduction

For the knowledge present in the World–Wide Web to be processable by machines, it should be converted from its free text form into a structured representation. The idea coined by the Semantic Web initiative [2] is to insert structured knowledge manually into the web pages, in the form of explicit metadata. In the predictable future, however, pages rich in metadata will represent only a fraction of (valuable) web resources. It seems thus necessary to combine the harvesting of explicit metadata with the acquisition of *implicit* metadata hidden in various forms of 'object' data.

Beside standalone applications, the problem of WWW analysis has also been attacked from the multi–agent perspective, in particular, by *open collections of agents*, each of them specialising in particular *semantic* type of information. In this paper, we investigate yet a different possibility – that of a *tightly-coupled* collection of modules specialised in *syntactic* type of data: the closedness of the architecture should reduce the communication overhead, and the syntactic specialisation should enable the reuse of low-level (data-type-specific) routines across different tasks. As in any kind of complex distributed system, *ontology engineering* plays an important role in such architecture. However, due to its static character, the crucial problem is not the sharing of semantics during the process of problem solving, but rather the (offline) construction of ontologies from multiple datatype–specific domain models.

In the rest of this paper, we describe the principles as well as the prototype implementation of the distributed system designed for multiway acquisition of web metadata (section 2), discuss the role and the way of construction of formal ontology in this system (section 3), and present two case studies (section 4); the last two sections briefly compare the approach with existing research, and outline the perspectives for the future.

2 Architecture of the ***Rainbow*** System

The central idea of the *Rainbow*[3] system (Reusable Architecture for INtelligent Brokering Of Web information access) is the separation of different web analysis tasks according to the *syntactical type of data* involved. In this way, the natural complementarity and/or redundancy of information inferable from different types of data co-existing on the web should give way to robust and reusable applications. The prospective suite of metadata-acquisition components will thus consist of specialised modules for

- analysis of the (free-text sentence) linguistic structure
- analysis of the HTML mark-up structure
- analysis of URL addressing
- analysis of explicit metadata (in META tags and RDF)
- analysis of the link topology
- analysis of images.

plus additional modules handling source data as well as interaction with users and external services.

Rainbow can be characterised as a *distributed knowledge-based system* rather than a *multi-agent system*, since the modules have a limited degree of autonomy and their collection is relatively closed[4]. The functionality of the modules is realised by means of *web services* a priori defined by the WSDL [4] interface description. The model of communication is limited to synchronous requests and answers represented by means of the SOAP [3] protocol.

The syntactic (datatype–oriented) labour division has several important advantages – it enables:

1. to reuse *low–level analysis routines*, which do not significantly differ across the tasks and application domains

[3] For more information see `http://rainbow.vse.cz`. Beyond the acronym, the name is motivated by the idea that the individual modules for analysis of web data should synergistically ‘shed light’ on the web content, in a similar way as the different colours of the rainbow join together to form the visible light.

[4] This holds for the ‘internal’ analysis modules; on the other hand, various external services and application-dependent user interfaces are likely to be added.

2. to exploit the possibly deep–and–narrow *specialisation of application developers* (e. g. in linguistics, graphs, or image processing)
3. to perform *data mining* (for important abstraction patterns) on simple, *homogeneous representations* of data, and thus more efficiently[5]

The implementation of the architecture is currently limited to a quite primitive running prototype. The current version of the *linguistic* module is merely able to extract *sentences*[6], namely those potentially containing general information on the organisation offering products/services (cf. section 4.2). The criterion for selecting sentences is the presence of one of a family of 'indicators', which have been collected by offline data mining in a web directory (see [10]). Other modules have either been implemented in a trivial manner (mere extraction of useful types of explicit metadata, without any abstraction step), exist as a stand-alone program (URL analysis, see [13]), as a dedicated, domain-specific application (image analysis, see section 4.1) or merely as design models plus informal knowledge bases (HTML mark-up and link topology analysis).

In addition to the analysis modules, a *source data* component has been implemented, which is responsible for the acquisition (incl. the removal of duplicate pages), cleaning, storage and provision of source data. For the acquisition of data, a usual web spider has been created. The cleaning amounts to a conversion of common HTML to well-formed XHTML. The data are then stored in a relational database and sent to the other modules on request.

Finally, a (testbed) *navigation interface* has been built, in the form of plug-in panel in the open-source Mozilla browser. Every time a new page is open in the browser, the page is downloaded and pre-processed (provided it is not yet in the database), and the analysis modules are invoked. Their results are just displayed, no integration is performed at the moment.

The general diagram of the architecture is at Fig. 1; for simplicity, external services are omitted. Fig. 2 shows a screenshot of the browser equipped with the *Rainbow* navigation pane. The sections of the pane correspond to the listing of 'similar' pages provided by the Google server (as an additional, external web service), to the content of selected META tags (raw explicit metadata), and to the sentences provided by linguistic analysis[7].

The process of building a knowledge base for a particular analysis module, within a particular *Rainbow*-based application, should rely on the analysis of

[5] An obvious risk related to the last point is that of missing some aspects of information perceivable only in combination of multiple types of data. We however hope that a large proportion of such combinations can still be captured at the level of inter-module communication, despite the abstractions performed by the modules prior to this communication.

[6] The version of linguistic module capable of term–level extraction and distinguishing specific metadata classes is forthcoming.

[7] The presence of 'offer' (one of the 'indicators' found by web directory mining [10]) as the main verb in the sentence was basis for its selection. One of the heuristics not yet implemented would immediately extract 'horses' (the syntactical object) as suspected metadata of type 'generic product name'.

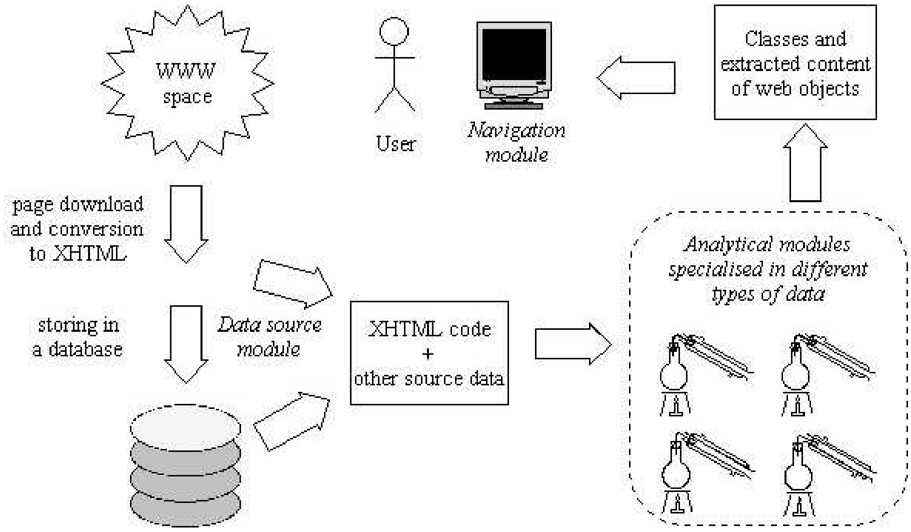

Fig. 1. Diagram of the *Rainbow* architecture

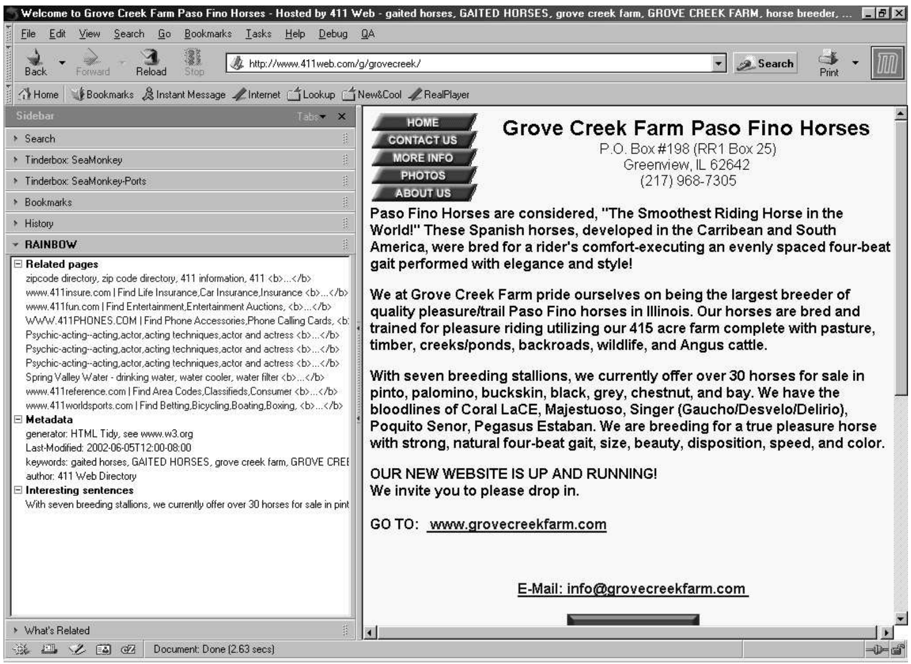

Fig. 2. Screenshot of Mozilla with the *Rainbow* navigation pane

problem domain (including interaction with domain experts) by the responsible developer – knowledge engineer specialised in the particular type of data. The result should ultimately have the form of knowledge base connecting high–level semantic concepts (such as categories of web pages, images, or textual 'messages') to generic datatype–specific features (such as sets of 'synonymous' URL strings, free-text terms, or skeletal HTML structures). In order for the whole collection of knowledge bases to be consistent, the developers should first interact with each other 'on behalf of their modules' and unify their views of the domain.

Formal tools should make this interaction more efficient, among which *ontologies* are to play a crucial role. It is thus, actually, the ontology viewpoint from which *Rainbow* (being itself not a multi-agent system in the usual meaning) brings some aspects of 'negotiations' common for multi-agent systems. These however do not follow from the nature of online *problem solving* but from that of offline *knowledge acquisition*.

3 Ontology Engineering for *Rainbow*

According to [15], ontologies are beneficial in three general aspects of the software system lifecycle: communication between people, inter-operability between systems, and system engineering. While for open multi-agent systems, the support of online interoperability is crucial, this is not the case for a collection of relatively fixed knowledge-based modules such as the *Rainbow* architecture. On the other hand, the *communication between people* is important, since, due to the datatype-based labour division, the individual developers (who should have the skills primarily related to processing the particular sort of data) often deal with similar or even same concepts in different guises. They should take this similarity into account in order to conceive the prospective reasoning services consistently and efficiently. Such ontologies for 'human consumption' may clearly be *informal*, such as glossaries and/or diagrams. In the next phase, however, the *design* of the overall (distributed) application comes into play. An ontology expressed in a *formal* language will then be helpful for checking *beforehand* the conceptual consistency of information to be exchanged – in our case, by means of web services. The same domain ontology could be used for different tasks in the same domain, such as for systematic extraction of product information from company pages as well as for the retrieval of companies offering the desired (generic) product.

We have developed an 'upper-level ontology' of web information objects, named WEB-OBJ; its hierarchy of classes correspond to *syntactical types* of objects. WEB-OBJ constitutes the basis for developing ontologies for more specific *domains* of the web; roughly, the classes in domain ontologies (as illustrated by the following section) correspond to *semantic classes* of objects, and are subclasses of the classes from WEB-OBJ. Ideally, the developers of datatype-specific knowledge bases should create their partial domain models as formal ontologies, which could in turn be *merged* and/or *integrated*[8]. with the help of semi-formal tools. Currently, however, we have experience only with informal partial models, since their developers do not yet have enough experience with formal ontology modelling. The target ontology is at the same time integrated and formalised by the single ontology engineer; problems related to the semantics of concepts are then fed back to the module-developers.

4 Case Studies

Two case studies have so far been tackled in the context of the *Rainbow* project. Both are quite limited in extent and complexity, and their realisation does not fully correspond to the sequence of steps suggested in section 3.

4.1 Case Study 1: Pornography Recognition

Recognition of pornographic ('adult–only') websites and pages is a well–known, binary discrimination task addressed by both commercial and academic research. Techniques used involve the analysis of text, images, as well as maintenance of server 'blacklists'; these are, however, typically used in isolation, or bundled together in an ad hoc manner. In a PhD project, associated with the *Rainbow* project, multiple ways of pornography recognition (differing in the type of data used) have been systematically examined, implemented and tested:

Image analysis. Images on pages have been analysed for the presence of particular (body) colours, and for objects in particular position.

URL analysis. Indicative terms have been sought in the URL, namely in the directory structure and filename. Heuristics for combination of term weights (mutual strengthening of otherwise non-significant terms etc.) have been employed.

HTML structure and link topology analysis. The link structure consisting of a 'selection page' (with fingerprints) pointing to multiple (image) target pages is rather characteristic; moreover, the number of incoming links is also indicative. To distinguish pornography from art galleries, detailed analysis of HTML mark-up (incl. the computation of text-to-image ratio) proved useful.

Explicit metadata analysis. The RSAC rating present in META tags of pages also served as one of the inputs.

The combination of all methods has achieved the accuracy (in recognising the porno sites) of approx. 95%, which was significantly more than that of each individual method. However, since the pornography application was developed before the first version of the actual *Rainbow* architecture, the integration of the

[8] As it seems, this process may be denoted by both these terms as distinguished in [11]: the partial models may significantly overlap and/or contain to separate (complementary) layers of objects.

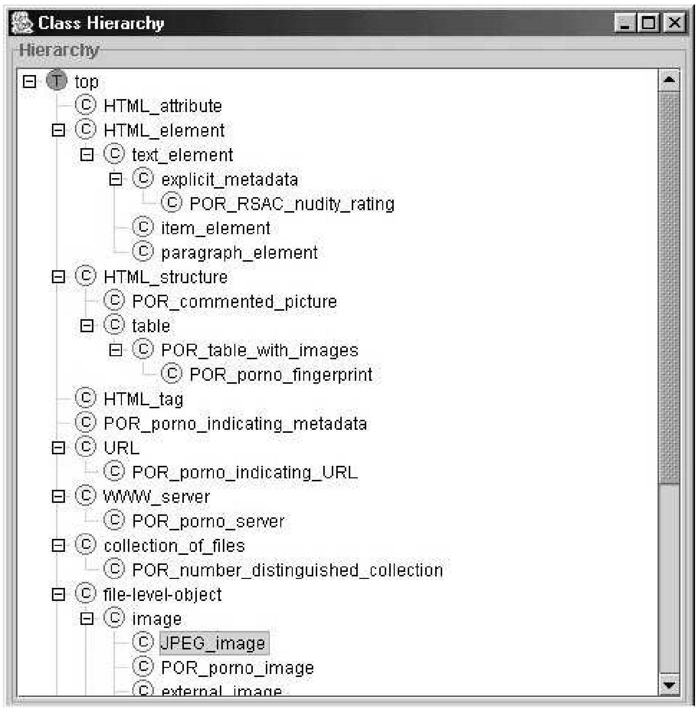

Fig. 3. OILed view of class hierarchy for the web pornography ontology

modules was not based on web services; rather, they have been run on the same data and the results integrated offline. The development of the modules has not been guided by an explicit *domain ontology*, either. Therefore, we have later 're-engineered' such ontology, in order to get a deeper insight into the problem domain, and to prepare a new implementation of the application in the future, within the fully–blown *Rainbow* architecture. We first formulated the informal domain models constrained by the types of data; after the separation of HTML structure and link topology aspects, five very simple models arose. In the process of their integration, semantically equivalent concepts have been unified, and the domain–specific part of the taxonomy was built[9]. The formal language used was OIL [7]. An advantage of this language is the availability of a reasoner (FaCT), which enables to test the *consistency* of the model. The structure of the ontology is depicted in the taxonomy window of the OILed editor[10] in Fig. 4.1. Since OILed does not by itself support ontology inheritance and merging, we have (provisionally) 'grafted' the pornography extension on a copy of the WEB-OBJ ontology; the pornography concepts are distinguished by the POR prefix.

[9] In addition to approx. 30 classes (and 12 slots) inherited from the WEB-OBJ ontology, the (pornography) domain–specific part contains another 12 classes.

[10] Available from `http://oiled.man.ac.uk`.

4.2 Case Study 2: Collection of Information on Products and Services

The incentive for starting the development of *Rainbow* itself was a particular web access problem: acquisition of implicit metadata from web pages of small 'organisations offering products and services' (let us nickname it the 'OOPS' domain). This is reflected in the tuning of knowledge bases developed and in the choice of training and testing data. The problem of ontology construction from the datatype-constrained domain models is more pressing here, since the (six) individual models are developed by different people, and even based on different samples of data[11]. The models vary in size and detail of elaboration; they deal, respectively, with

– *URL-based* concepts, such as 'company description page' recognisable by terms such as 'about-us' in filename.
– *Linguistic* concepts, such as 'generic product name' recognisable by its object-dependency on indicators such as 'our ... includes'.
– *HTML* concepts, such as 'pricelist item' recognisable by containment in particular HTML structures with tabular display.
– *Image* concepts, such as 'page section separator' being an inline image with specific properties.
– *Topology* concepts, such as 'intro page' recognisable by no incoming and single outcoming link.
– *Explicit metadata* concepts, such as 'page offering software' recognisable by structures such as `<meta name='category' content='software'>`

Even here, the partial models are only *informal*, since the individual developers were, at the time of their construction, not familiar enough with OIL or other formal ontology language. Since the number of concepts is substantially higher than in the pornography domain, it seems to be necessary to *filter out* some of them from the shared ontology, to overcome the well known 'hugeness problem'. Candidates for such leave-out are the concepts semantically distant from other modules, or those not contributing to the results of analysis (and included in the model only 'for completeness' , such as that of 'page-access-counter' in HTML mark-up analysis. The process of shared ontology construction (again, in OIL) is still in progress; its result should serve for the future development of web services in the 'OOPS' application of *Rainbow*.

5 Related Work

The architecture of *Rainbow* seems to be rather unique, which also holds for the ontology construction principles described in this paper. Unlike the truly multi-agent approaches, examples of which are [8, 9, 16], we rely on a tightly-coupled

[11] *OpenDirectory* (`www.opendir.com`), *WebTREC* [1] and several smaller ad-hoc samples are used.

collection of modules specialised in *syntactic* type of data. The closedness of the architecture should reduce the communication overhead (the web-service protocols are simpler than agent protocols such as KQML or FIPA ACL), and the syntactic specialisation should enable the reuse of low-level (data-type-specific) routines across different tasks. Freitas [8] reflects the need for such reuse by sharing the low-level routines among the agents by means of an ontology. Such a step is however problematic regarding that an ontology should be an abstract knowledge model rather than a collection of specific routines; our principle of syntactic specialisation (and use of ontology to share purely the declarative, semantic knowledge) seems to avoid such mismatch.

The *Web→KB* [5] project is similar to the ours in attempting to exploit the redundancy of different types of data expressing the same information. Its stress is however on the use of sophisticated *machine learning* techniques while ontologies play only a minor role. Moreover, the separate analysis of different types of data is ad hoc (for the purpose of concrete learning sessions, in the 'co-learning' framework) rather than long-term and systematic.

6 Conclusions and Future Work

In this paper, we have presented the general ideas as well as the provisional implementation of a distributed system for acquisition of implicit metadata from the web, in which different syntactic types of data will be analyses by separate modules. We argue that a formal ontology will play an important role in this architecture, though in a different manner than in mainstream multi-agent architectures. The crucial point is to support the process of constructing a coherent application from a collection of web services that are relatively independent but tend to overlap in underlying concepts. A formal ontology integrated from the datatype-specific domain models could then reveal semantic redundancies and mismatches; languages such as OIL, provided with a sound reasoning mechanism, serve well for this purpose.

An almost immediate technical improvement of the ontology design process should be the adoption of an (existing) software tool more powerful than OILed, and supporting collaborative development and merging of formal ontologies. Provided the developers of datatype-specific models use such tools, a larger portion of the whole process could be automated.

Furthermore, the ontology engineering effort in *Rainbow* has been so far was oriented merely on the sharing of *concepts*. However, concrete *lexical* items also deserve to be shared, namely, among the modules dealing with text (be it in sentences, HTML mark-up, explicit metadata, or even URLs). We thus want to investigate the options of constructing shared domain lexicons in our multiway context.

Besides the domain ontologies, *task ontologies* similar e. g. to template models in CommonKADS [12] could play an important role as reusable skeletal components for different web access applications. We hope that a starting point for building a library of such ontologies could be the generic framework of web information access outlined in [14], with three atomic inferences (*classify*, *retrieve* and *extract*) being the 'flesh' of most information access tasks.

Finally, the evolving prototype implementation of *Rainbow* should constantly serve for empirical validation of (not only) the mentioned research directions.

Acknowledgements

The authors would like to thank their collaborators in the *Rainbow* project, who contributed to the development of ontologies and/or prototype implementation, namely to Martin Kavalec, Pavel Kupka, Jiří Klemperer, Martin Sajal, Petr Berka, Vilém Sklenák and Petr Strossa.

The research has been partially supported by the grant no. 201/00/D045 (Knowledge model construction in connection with text documents) of the Grant Agency of the Czech Republic.

References

1. Peter Bailey, Nick Craswell and David Hawking: *Engineering a multi-purpose test collection for Web retrieval experiments. Information Processing and Management*, 2001.
2. Tim Berners-Lee, James Hendler and Ora Lassila: *The Semantic Web.* Scientific American, May 2001.
3. D. Box et al.: *Simple Object Access Protocol (SOAP) 1.1 – W3C Note.* W3C, 2000. http://www.w3.org/TR/SOAP/
4. E. Christensen et al, *Web Services Description Language (WSDL) 1.1*, W3C Note, 2001. `http://www.w3.org/TR/2001/NOTE-wsdl-20010315`
5. Mark Craven et al.: *Learning to Extract Symbolic Knowledge from the World Wide Web.* In: Proc. of 15th National Conference on Artificial Intelligence (AAAI-98).
6. DeRose S., Daniel R., Maler E.: *XML Pointer Language (XPointer) – W3C Working Draft.* W3C, 1999. http://www.w3.org/TR/xptr.
7. Dieter Fensel et al.: *OIL in a nutshell.* In: EKAW2000, Springer Verlag, 2000.
8. Fred L. G. Freitas and Guilhermo Bittencourt: *Cognitive Multi-agent Systems for Integrated Information Retrieval and Extraction over the Web.* In: Proc. International Joint Conference SBIA-IBERAMIA 2000, LNCS 1952, Springer-Verlag 2000.
9. Stefan Haustein: *Utilising an Ontology Based Repository to Connect Web Miners and Application Agents.* In: (G.Stumme, A.Hotho, B.Berendt, eds.) Semantic Web Mining, Workshop at ECML/PKDD-2001, Freiburg 2001, pp.59-66.
10. Martin Kavalec, Vojtěch Svátek and Petr Strossa: *Web Directories as Training Data for Automated Metadata Extraction.* In: (G.Stumme, A.Hotho, B.Berendt, eds.) Semantic Web Mining, Workshop at ECML/PKDD-2001, Freiburg 2001, pp.39-44.
11. H. Sofia Pinto, Asuncion Gomez Perez, and Joao P. Martins: *Some Issues on Ontology Integration.* In: IJCAI'99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. Stockholm 1999.
12. Guus Schreiber et al.: *Knowledge Engineering and Management. The CommonKADS Methodology.* MIT Press, 1999.

13. Vojtěch Svátek and Petr Berka: *URL as starting point for WWW document categorisation.* In: (Mariani J., Harman D.:) RIAO'2000 - Content-Based Multimedia Information Access, CID, Paris, 2000, pp.1693-1702.
14. Vojtěch Svátek et al.: *Framework and Tools for Multiway Extraction of Web Metadata*. In: Proc. Information Systems Modelling, Roznov 2002.
15. Mike Uschold and Robert Jasper: *A Framework for Understanding and Classifying Ontology Applications*. In: IJCAI'99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. Stockholm 1999.
16. G. Zhong, T. Mine, T. Helmy and M. Amamiya: *The Design and Application of the KODAMA system.* In: (Hruška T., Hashimoto M., eds.:) KnowledgeBased Software Engineering, IOS Press, 2000.