

Web Service Composition for Deductive Web Mining: A Knowledge Modelling Approach

Vojtěch Svátek¹, Annette ten Teije² and Miroslav Vacura¹

¹ Department of Information and Knowledge Engineering,
University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic
e-mail: svatek@vse.cz, vacura@chello.cz

² Department of Artificial Intelligence, Vrije Universiteit Amsterdam,
De Boelelaan 1081A, 1081HV Amsterdam, The Netherlands
e-mail: annette@cs.vu.nl

Abstract. Composition of simpler web services into custom applications is understood as promising technique for information requests in a heterogeneous and changing environment. This is also relevant for applications analysing the content and structure of the web. We discuss the ways the problem-solving-method approach studied in artificial intelligence can be adopted for template-based service composition for this problem domain; main focus is on the classification task.

1 Introduction

Composition of simple *web services* into sophisticated (distributed) applications recently became one of hottest topics in computer science research. Three alternative research streams can be identified:

1. *Programming in the large*, i.e. composition of services by (more-or-less) traditional procedural programming in languages such as BPEL4WS³, inspired by workflow research. This stream is the only one recognised by most of the industrial IT community, to date; its main advantage is perfect control over the choice and linkage of different services, at design time. This however, on the other hand, entails a rather low degree of flexibility at run time.
2. *Planning* in artificial intelligence style, based on pre- and post-conditions of individual services without pre-specified control flows, as in OWL-S[4]. This approach offers extreme flexibility; however, the results may be quite unpredictable if all conditions are not perfectly specified, which may often be difficult in real environments.
3. *Template-based* composition, in which concrete services are filled in run time into pre-fabricated templates [9, 19].

The area of application for (composite) web-services is potentially quite wide. While the focus is most often on B2B transactions and financial services, the

³ <http://www-128.ibm.com/developerworks/library/ws-bpel>

general paradigm appears useful even for less critical tasks such as organisation of scientific events [19] or information harvesting from the *surface web*, which is the focus of the current paper.

The way information is presented on the web typically combines multiple types and representations of data. Free text is interleaved with images and structured lists or tables, pages are connected with hyperlinks, labelled with URLs (often containing meaningful tokens) and endowed with explicit meta-data (in specialised tags). Different methods of web data analysis (focusing each on a different data type/representation) may provide complementary and/or supplementary information. Reducing the analysis on a single method, which is typically done e.g. in text categorisation or information extraction projects, may thus lead to significant information loss. On the other hand, a monolithic application encompassing many methods would be impossible to maintain (in view of permanent changes in web data standards and conventions) as well as reuse in different domains. The only solution thus seems to be to combine multiple, relatively independent, tools, for which web services offer a collection of relatively mature and widespread interface standards (such as SOAP and WSDL). This is the approach taken in the *Rainbow* project [16], in which web-analysis services based on diverse principles (statistics, linguistic, graph theory etc.) have been designed and equipped with hand-crafted or inductively trained knowledge bases; adoption of third-party services is also envisaged.

As the number of available web-analysis tools increases, their composition by traditional programming becomes cumbersome. On the other hand, the space of suitable tools will hardly be as borderless as in semantic-web scenarios of information search, which are assumed amenable to planning approaches. The *template-based approach* thus looks as a reasonable compromise.

Most recently, ten Teije et al. [19] suggested to view web service composition templates as analogy to *problem solving methods* (PSMs), i.e. abstract descriptions of knowledge-based reasoning scenarios, which have been intensely studied in the knowledge modelling community for nearly two decades (see e.g. [5, 12]). In addition, they suggested to view the *configuration* of the template again as a kind of reasoning task, namely, that of *parametric design*. Each concrete application is assumed to be specified (in sufficient detail) merely as combination of values assigned to a fixed set of parameters. The configuration process is carried out by a so-called *broker* tool, and employs the *propose-critique-modify* (PCM) reasoning method⁴, taking advantage of *background knowledge* of the broker. Independently, Svatek et al. [18] designed a collection of *PSMs* abstracted from real *deductive web mining* applications, with individual components (services) positioned in a *multi-dimensional space*. This space could play a similar role as the space of template parameters from [19], no reasoning method had been however formulated for automated configuration.

The goal of the current paper is thus to evaluate the possibility to adapt the parametric design approach from [19] to the (specific features of) web analysis PSMs from [18]. Main focus is on *classification*, which is the only task considered

⁴ I.e. a ‘meta-level’ PSM with respect to that incorporated in the template itself.

in [19] and also one of tasks studied in [18]. The nature of the research, being joint venture of two related but independent projects, influences the structure of the paper. Section 2 explains the idea of web service composition based on parametric design [19], while section 3 describes the multi-dimensional framework and web-analysis PSMs from [18]. Section 4 then suggests a modification of the parametric design approach suitable for DWM, and shows examples of templates and background knowledge to be possibly used by a broker. Finally, section 5 surveys some related projects, and section 6 wraps up the paper and suggests directions for future research.

2 Configuration of Web Services as Parametric Design

2.1 Motivations

Current approaches to Web service configuration are often based on pre/post-condition-style reasoning. Given descriptions of elementary Web services, and the required functionality of the composite Web service, they aim to try to construct a ‘plan’ of how to compose the elementary services in order to obtain the required functionality. Planning techniques are heavily investigated for this purpose [13]. In [19], we instead proposed a *knowledge intensive* approach to the creation of composite Web services. We described a complex Web service as a fixed template, which must be configured for each specific use. Web service configuration can then be regarded as *parametric design*, in which the parameters of the fixed template have to be instantiated with appropriate component services. During the configuration process, we exploit detailed knowledge about the template and the components, to obtain the required composite web service. Whereas in other work the main metaphor is “Web service configuration = planning” (i.e. generalised reasoning based on only component specifications), our approach is based on the metaphor “Web service configuration = brokering” (i.e. reasoning with specialised knowledge in a narrow domain). A *planner* is assumed to be *domain-neutral*: it is supposed to work on any set of components, simply given their descriptions. A *broker* on the other hand exploits specific knowledge about the objects it is dealing with. In the remainder of this section, we describe how such a broker can be equipped with configuration knowledge on how to combine these web services.

2.2 Parametric Design

Parametric design is a simplification of general configuration. It assumes that the objects to be configured (in our case: complex Web services) have the same overall structure that can be captured by templates. Variations on the configuration can only be obtained by choosing the values of given parameters within these templates. We will show that for specific type of web services, namely classification services, this is indeed possible.

An existing reasoning method (PSM) for parametric design is *Propose-Critique-Modify*, or PCM for short [6]. The PCM method consists of four steps:

- The *propose step* generates an initial configuration. It proposes an instance of the general template used for representing the family of services.
- The *verify* step checks if the proposed configuration satisfies the required properties of the service. This checking can be done by both pre/post-condition reasoning, or by running the service.
- The *critique* step analyses the reasons for failure that occurred in the verification step: it indicates which parameters may have to be revised in order to repair these failures.
- The *modify* step determines alternative values for the parameters identified by the critique step. The method then loops back to the verify step.

The propose-critique-modify method for Parametric Design requires specific types of configuration knowledge to drive the different steps of the configuration process. The question is whether this configuration knowledge (PCM knowledge) can be identified for large classes of Web services. It turns out that this is indeed possible for a specific class of web services, namely, *classification* ones.

2.3 Application on Classification Services

The common definition of classification is [15]: "Classification problems begin with data and identify classes as solutions. Knowledge is used to match elements of the data space to corresponding elements of the solutions space, whose elements are known in advance." More formally, classification uses knowledge to map observations (in the form of $\langle feature, value \rangle$ -pairs) to classes.

We address the question whether classification services can be described in a single template. [10] does indeed present such a general template:

1. First the observations have to be verified whether they are legal (*Check*).
2. All legal observations ($\langle feature, value \rangle$ -pairs) have to be scored on how they contribute to every possible solution in the solution space (*MicroMatch*).
3. Individual scores are then aggregated (*Aggregate*).
4. Candidate solutions are determined via aggregated scores (*Admissibility*).
5. Final solutions are selected among candidate solutions (*Selection*) .

This structure constitutes the overall template for classification services, which can be easily captured in current Web service description languages such as OWL-S [4]. Example values of *Admissibility* parameter are (see [19] for more):

- *weak-coverage*: All $\langle feature, value \rangle$ pair in the observations are *consistent* with the feature specifications of the solution.
- *strong-coverage*: All $\langle feature, value \rangle$ pair in the observations are *consistent* with the feature specifications of the solution and *explained* by them.
- *strong-explanative*: All $\langle feature, value \rangle$ pair in the observations are *consistent* with the feature specifications of the solution, *explained* by them, and all features specified in the solution are *present*.

The value of *Selection* parameter then decides whether e.g. the number of unexplained and missing features is considered in ranking candidate solutions.

The broker may employ e.g. the following pieces of knowledge:

- Propose knowledge for the *Admissibility* parameter: if many $\langle \text{feature}, \text{value} \rangle$ pairs are irrelevant then do not use *strong-coverage*.
- Critique knowledge for the *Selection* parameter: if the solution set is too small or too large then adjust the *Admissibility* or the *Selection* parameter.
- Modify knowledge for the *Admissibility* parameter: if the solution set has to increased (reduced) in size, then the value for the *Admissibility* parameter has to be moved down (up) in the following partial ordering: *weak-coverage* \prec *strong-coverage* \prec *strong-explanative*.

A prototype PCM broker has been successfully applied on real data in the domain of conference paper classification (for reviewer assignment).

3 Framework and PSMs for Deductive Web Mining

3.1 The *TODD* Framework

In [18], *deductive web mining* was defined as ‘all activities where pre-existing patterns are matched with web data’; the patterns may be either hand-crafted or learnt. We proposed a framework that positions any DWM tool or service within a space with four dimensions:

1. Abstract *task* accomplished by the tool:
 - *Classification* of a web object into one or more pre-defined classes.
 - *Retrieval* of one or more web objects.
 - *Extraction* of desired information content from (within) a web object.

The *Classification* of an object takes as input its identifier and the list of classes under consideration. It returns one or more classes. The *Retrieval* of desired objects takes as input the (syntactic) *type* of object and *constraints* expressing its class membership as well as (part-of and adjacency) relations to other objects⁵. It outputs the *addresses* (based on URIs, XPath expressions and the like) of relevant objects. The *Extraction* task takes as input the class of information to be extracted and the scope (i.e., an object) within which the extraction should take place⁶. It outputs some (possibly structured, and most often textual) *content*. In contrast to Retrieval, it does not provide the information about location from where the content was extracted.

2. Type of *object* to be classified or retrieved⁷. The types, such as Document, Hyperlink, or Phrase, represent an upper-level of abstraction of web objects, and are defined by the Upper Web Ontology (see below). The basic assumption is that the type of object is always known, i.e. its assignment is not by itself subject of DWM.

⁵ For example: “Retrieve (the XPath addresses of) those HTML tables from the given website that are immediately preceded with a possible ‘Product Table Introduction Phrase’ (containing e.g. the expression `product*`)”.

⁶ For example: “Extract the occurrences of Company Name within the Website”.

⁷ *Extraction* is not unambiguously associated with a particular object.

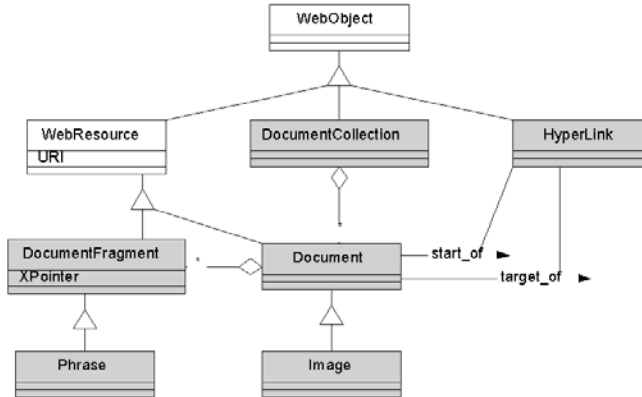


Fig. 1. UML diagram of Upper Web Ontology

3. *Data type and/or representation*, which can be e.g. full HTML code, plain text (without tags), HTML parse tree (with/without textual content), hyperlink topology (as directed graph), frequencies of various sub-objects or of their sequences (n-grams), image bitmaps or even URL addresses.
4. *Domain* in which the service is specialised.

We thus denote the framework as ‘*task-object-data(type)-domain*’ (TODD). Its dimensions are to high degree independent, e.g. *object type* is only partially correlated with *data type*. For example, a document may be classified based on its HTML code, URL, META tag content or position in topology. Similarly, a hyperlink can be classified based on its target URL or the HTML code of source document (e.g. the menu structure containing the respective `<a>` tag).

The TODD framework by itself does not offer any added value to DWM application design until augmented with appropriate *ontologies*. Due to lack of space, we only show the *Upper Web Ontology* (see Fig. 1), which acts as general basis for more specific ontologies. Within the *Rainbow* project [16], domain ontologies (for bicycle product information and for web pornography) as well as ontologies related to analysis of specific types of web data have been designed.

3.2 Problem-Solving Methods

A characteristic feature of the web space is lack of clear object-feature-value structures, since, in a sufficiently comprehensive model, most features deserve to become objects of their own. As consequence, *structural* (say, recursive) PSMs arise. Eight PSMs have been formulated in [18]. Here we only concentrate on the (three) *Classification* and (two) *Retrieval* ones, and omit the (three) *Extraction* PSMs, for the sake of brevity.

Look-up based Classification amounts to picking the whole content of the given object, and comparing it with content constraints (such as look-up table),

which yields the class; for example, a phrase is a Company Name if listed in business register. *Compact Classification* also corresponds to a single inference, it is however not based on simple content constraints but on some sort of computation (e.g. Bayesian classification), which is out of the scope of the knowledge modelling apparatus. Finally, *Structural Classification* corresponds to classification of an object based on the classes of related objects (sub-objects, super-objects and/or neighbours). It is thus decomposed to *retrieval* of related objects, their *individual classification*, and, finally, evaluation of *global classification patterns* for the current object. It is therefore *recursive*: its ‘inference structure’ typically contains full-fledged (Direct) Retrieval and Classification tasks. Compared to the generic Classification template from section 2, this notion of classification is slightly simplified and more goal-driven. Some parts of Structural Classification PSM can be mapped on the generic template: classification from lower level of recursion is similar to MicroMatch, while evaluation of global pattern unites the Aggregate, Admissibility and Selection steps. There is no Check step (since no observations are known a priori), but an extra step of Retrieval (since objects relevant for classification of current object have first to be determined).

In *Direct Retrieval*, relevant objects are first retrieved based on parthood and adjacency constraints and then classified. Objects whose classes satisfy the class constraints are output. In *Index-based Retrieval*, (abstract) class constraints are first operationalised so that they can be directly matched with the content of objects. Then the objects are retrieved in an index structure (which is separate from the web space itself), possibly considering structural constraints (provided structural information is stored aside the core index). Corresponding CommonKADS [12] inference diagrams can be found in [18].

As an example of use of such PSMs, Table 1 shows the pseudo-code of pornography-recognition application consisting of tools developed within the PhD thesis [20]; descriptions of other applications can be found in [18]. The pseudo-code is assumed to clarify the structure of an application for a human user rather than to be run by machine. The ‘predicate’ corresponds to the first dimension in the TODD framework, i.e. *task name*; an extra letter is used to distinguish the PSMs. The first ‘argument’ is a variable referring to the *current object* of the task instance: input object in the case of Classification and output object/s in the case of Retrieval. The second to fourth ‘arguments’ reflect the three remaining TODD dimensions: *object type*, *data type* and *domain*. Finally, the fifth ‘argument’ contains additional specifications: list of classes distinguished in the classification task, and list of logical expressions determining the set of objects to be retrieved, respectively.

The upper level of the pornography-recognition process is an instantiation of the *Structural Classification* PSM. In order to classify the whole website (i.e. document collection), symptomatic ‘out-tree’ topology structures are first sought; their sources (local hubs) can possibly be identified with ‘index’ pages with image miniatures. To verify that, the hub is examined for presence of ‘nudity’ PICS rating in META tags (Look-up Classification PSM), for presence of indicative strings in the URL, and its whole HTML code is searched for ‘image gallery’-like

Table 1. PSM-based pseudo-code representation of pornography application

```
Clas(DC, DocCollection, _, Pornography, [PornoSite,@other]) :-
  RetD(D1, Document, topology, General, [D1 part-of DC, LocalHub(D1)]),
  Clas(D1, Document, _, Pornography, [PornoIndex,@other]),
  RetD(D2, Document, topology, General, [D2 follows D1]),
  Clas(D2, Document, _, Pornography, [PornoContentPage,@other]).
% classification of index page
Clas(D, Document, _, Pornography, [PornoIndex,@other]) :-
  ClaL(D, Document, meta, Pornography, [PornoResource,@other]),
  Clas(D, Document, url, Pornography, [PornoResource,@other]),
  RetD(DF, DocFragment, html-txt, General, [DF part-of D, ImgGallery(DF)]),
  ClaC(DF, DocFragment, freq, General, [ScarceTextFragment,@other]).
% classification of content page
Clas(D, Document, _, Pornography, [PornoContentPage,@other]) :-
  ClaL(D, Document, meta, Pornography, [PornoResource,@other]),
  RetD(Im, Image, html-txt, General, [Im referenced-in D]),
  ClaC(Im, Image, image, Pornography, [PornoImage,@other]).
```

structures with low proportion of text (which distinguishes pornography from regular image galleries). The analysis further concentrates on individual pages referenced by the hub, and attempts to identify a single dominant image at each of them. The images are then analysed by (bitmap) image analysis methods; in particular, the proportion of body colour and the central position of a dominant object are assessed. In the description, we omit the ‘evaluation of global classification pattern’ subtasks, for brevity; their inclusion would be straightforward.

4 DWM Service Configuration as Parametric Design

4.1 Limitations of Fixed Template

The methods presented in the previous two sections share the interest in *configuring* a wide collection of *web services* into a functional application, and both rely on application templates in the form of *problem-solving methods*. The inventory applied on general classification PSMs (section 2) is more advanced, as it already comprises an operational meta-level, itself based on the problem-solving modelling paradigm. The *propose-critique-modify* method of *parametric design*, implemented by means of a configuration broker, enables to effectively search the space of the classification method family with respect to the task at hand. It thus seems obvious to apply a similar approach in the area of deductive web mining, which is of equally *analytic* nature and even comprises *classification* as one of underlying tasks. However, as we outlined in section 3, the PSMs for deductive web mining tend to involve *recursion*: a reasoning process starting at one object is successively redirected to other objects in its parthood or neighbourhood. This more-or-less disqualifies reasoning methods relying on a *single and entirely fixed*

feature template, of which parametric design is a typical representative. There seem to be at least two possible solutions to this problem:

1. to allow for *multiple templates per task*, differing in the number of ‘sibling’ sub-tasks and degree of recursion, and to include *heuristics for template selection* as part of broker knowledge.
2. to modify the *parametric design algorithm* to involve, in addition to setting parameter values, also *template-restructuring operations* such as subtask replication and recursive unfolding (i.e. replacement of parameter with a whole template for processing a different object).

In the rest of this section, we outline the first solution, since it is easier to design and implement in its rudimentary form; it obviously oversimplifies many aspects of real-world settings.

4.2 Multiple-Template Solution

Table 2 shows four versions of template for the classification task: the first one amounts to single way of (presumably, Look-Up or Compact) classification of the current object, the second aggregates two different ways of classifying the current object, the third relies on another object in order to classify the current object, and the fourth combines (presumably Look-Up or Compact) classification of current object with its structural classification (via classification of another object). The template must have its arguments filled for concrete use, such as in the pornography recognition application in section 3.

For simplicity, we again omit the evaluation of *global classification pattern*, which would be an additional component in the template. For example, the presence of sub-object of certain class determines the class of the super-object in a certain way; similarly, classification of the same object by different methods has to be compared and the result computed (e.g. by voting or weighing).

Let us formulate, analogously to section 2, some examples of broker knowledge. We will limit ourselves to *Propose* knowledge, which is relevant for initial setting of parameters. Note that, in our multiple-template version, broker knowledge relates to *template selection* as well as to *specification of arguments* for all subtasks within the template:

- Templates with lower number of distinct objects (X, Y, Z, ...) should be preferred.
- Non-recursive templates should be preferred; moreover, look-up classification should be preferred to compact classification.
- Default partial ordering of data types with respect to object classification, for *Document* object (may be overridden in a domain context):
frequency > *URL* > *topology*, *free_text* > *metadata*
- URL-based or topology-based classification (as rather unreliable kinds of services) should never be used alone, i.e. can only be filled into a template with ‘parallel’ classification of same object, such as SC2 or SC4

- Default partial ordering of types of relations (`@rel`) to be inserted into classification template (may be overridden in a domain context):
part-of \succ *is-part* \succ *adjacent*
- Preference of domains used in structural classification, with respect to the domain of current object: *same domain* \succ *super-domain* \succ *other-domain*.
- The class of object determined by a Classification sub-task should be (according to domain knowledge) sub-class of the class of objects determined by the immediately preceding Retrieval sub-task in the template.

These heuristics are merely tentative, to illustrate the variety of possible broker knowledge for DWM applications. Let us further show a hypothetical scenario of their use, in connection with the pornography-recognition application from section 3. Imagine a web pornography ontology⁸ containing among other the following description-logic axioms:

```
PornoSite same-class-as (WebSite and (has-part some PornoIndex))
PornoIndex same-class-of (LocalHub and (followed-by >1 PornoContentPage))
```

For an application recognising pornography websites, the broker would select the template SC3, which is simpler than SC4; neither SC1 nor SC2 would be applicable (assuming no service were able to recognise `PornoSite` by Look-Up or Compact Classification). In attempting to fill SC3 in, it would seek a class of related object that could help determine the class of current object. With the help of the first axiom, it finds out that `PornoIndex` could serve for the purpose (as part of sufficient condition); it will thus accordingly instantiate the Classification sub-task. Then it will determine, by the second axiom, a suitable class of objects to be retrieved in the preceding (Retrieval) sub-task as `LocalHub`; since this is not a pornography concept but generic concept, `Domain1` will be set to `General`. Finally, it finds out that `LocalHub` cannot be recognised as `PornoIndex` merely by Look-Up or Compact Classification. It will thus have to create another SC3 template, on the second level, in order to recognise `PornoIndex` by means of `PornoContentPages` following it in the link topology.

5 Related Work

In the *IBrow* project [1], operational PSM libraries have been developed for two areas of document search/analysis: Anjewierden [3] concentrated on *analysis of standalone documents* in terms of low-level formal and logical structure, and Abasolo et al. [2] dealt with information search in multiple external resources. Direct mining of websites was however not addressed; *IBrow* libraries thus do not cope with the problem of web heterogeneity and unboundedness, which motivated the development of the TODD framework. In contrast, the Armadillo system [7] attempts to integrate many website analysis methods; it currently relies on workflows manually composed from scratch by the user, although a template-based solution is also being envisaged.

⁸ This ontology has actually been developed in DAML+OIL, by the authors [17].

Table 2. Sample templates for classification task

```
SC1: Cla(X, TypeX, _, Domain, GoalClass) :-  
      Cla(X, TypeX, DataType1, Domain1, Class1),  
  
SC2: Cla(X, TypeX, _, Domain, GoalClass) :-  
      Cla(X, TypeX, DataType1, Domain1, Class1),  
      Cla(X, TypeX, DataType2, Domain2, Class2).  
  
SC3: Cla(X, TypeX, _, Domain, GoalClass) :-  
      Ret(Y, TypeY, DataType1, Domain1, [Y @rel X | OtherExps]),  
      Cla(Y, TypeY, DataType2, Domain2, ClassY).  
  
SC4: Cla(X, TypeX, _, Domain, GoalClass) :-  
      Cla(X, TypeX, DataType1, Domain1, Class1),  
      Ret(Y, TypeY, DataType2, Domain2, [Y @rel X | OtherExps]),  
      Cla(Y, TypeY, DataType3, Domain3, ClassY).
```

6 Conclusions and Future Work

Configuration of web services can be considered as *parametric design*, which enables to use the *propose-critique-modify* PSM. Thanks to *templates* we avoid configuring a webservice from scratch. Furthermore, such knowledge-intensive approach does not need complete functional descriptions of the components and of the required composite service but ‘only’ *configuration knowledge*. We attempted to apply this framework on service composition in the restricted domain of *deductive web mining*, in connection with a generic framework of DWM services. Due to the specific nature of web as underlying data structure, service templates tend to involve *recursion*, which also impacts the process of template-filling. Examples of specialised broker knowledge are shown, and a scenario of composing part of a pornography-recognition application is outlined.

Future research includes specification of templates for other DWM tasks, in particular those with nature of *extraction*, taking models of applications from [18] as starting point. The next step would be the development and evaluation of simple prototype of *problem-solving broker*. We will probably not be able to test it on real data (as was done in [19] for parametric design over the original classification template), since processing real-world web data would require functionalities (cleaning, complex parsing) not worth implementing in a throw-away prototype. We will rather rely on artificial, simplified data, and only proceed to real data when switching to a *functional architecture* incorporating independently-developed (often third-party) tools, as envisaged in the *Rainbow* project [16]. We also expect to implement and test the solution based on *automatic template restructuring*. Finally, we would like to compare the efficiency of both template-based variants with pure pre/post-condition *planning* approach.

The research is partially supported by the grant no.201/03/1318 of the Czech Science Foundation. The authors would like to thank Frank van Harmelen and Martin Labský for comments on drafts of this paper.

References

1. IBROW homepage, <http://www.swi.psy.uva.nl/projects/ibrow>
2. Abasolo, C. et al.: Libraries for Information Agents. IBROW Deliverable D4, online at <http://www.swi.psy.uva.nl/projects/ibrow/docs/deliverables/deliverables.html>.
3. Anjewierden, A.: A library of document analysis components, IBrow deliverable D2b. Online at <http://www.swi.psy.uva.nl/projects/ibrow/docs/deliverables/deliverables.html>.
4. Ankolekar, A. et al.: DAML-S: Semantic markup for web services. In: Proc. ISWC 2002, LNCS 2342, pp. 348–363.
5. Benjamins, R., et al. (eds.): *IJCAI Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*, 1999.
6. Brown, D., Chandrasekaran, B.: Design problem solving: knowledge structures and control strategies. *Research notes in AI*, 1989.
7. Ciravegna, F., Dingli, A., Guthrie, D., Wilks, Y.: Integrating Information to Bootstrap Information Extraction from Web Sites. In: IJCAI'03 Workshop on Intelligent Information Integration, 2003.
8. Kiefer, M.: Message to swsl-committee@daml.org, May 14, 2003.
9. Mandell, D. J., McIlraith, S. A.: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. In: Proc. ISWC2003.
10. Motta, E., Lu, W.: A Library of Components for Classification Problem Solving. In: Proceedings of PKAW 2000, Sydney, Australia, December 2000.
11. Narayanan, S., McIlraith, S.: Simulation, verification and automated composition of web services. In: Proc. WWW 2002.
12. Schreiber, G., et al.: Knowledge Engineering and Management. The CommonKADS Methodology. MIT Press, 1999.
13. Sheshagiri, M., desJardins, M., Finin, T.: A planner for composing service described in DAML-S. In: Workshop on Planning for Web Services, at ICAPS 2003.
14. Sirin, E., Hendler, J., Parsia, B.: Semi-automatic composition of web services using semantic descriptions. In: Web Services: Modeling, Architecture and Infrastructure workshop at ICEIS2003.
15. Stefik, M.: *Introduction to knowledge systems*, Morgan Kaufmann, 1995.
16. Svátek, V., Kosek, J., Labský, M., Bráza, J., Kavalec, M., Vacura, M., Vávra, V., Snášel, V.: Rainbow - Multiway Semantic Analysis of Websites. In: 2nd International DEXA Workshop on Web Semantics (WebS03), IEEE Computer Society 2003.
17. Svátek, V., Kosek, J., Vacura, M.: Ontology Engineering for Multiway Acquisition of Web Metadata. LISP-2002-1 Technical Report, 2002. Available from <http://rainbow.vse.cz/papers.html>.
18. Svátek, V., Labský, M., Vacura, M.: Knowledge Modelling for Deductive Web Mining. In: Proc. EKAW 2004, Springer Verlag, LNCS, 2004.
19. ten Teije, A., van Harmelen, F., Wielinga, B.: Configuration of Web Services as Parametric Design. In: Proc. EKAW 2004, Springer Verlag, LNCS, 2004.
20. Vacura, M.: Recognition of pornographic WWW documents on the Internet (in Czech), PhD Thesis, University of Economics, Prague, 2003.