

## MODELLING WEB SERVICE COMPOSITION FOR DEDUCTIVE WEB MINING

Vojtěch SVÁTEK, Miroslav VACURA, Martin LABSKÝ

*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, vacuram@vse.cz, labsky@vse.cz*

Annette TEN TEIJE

*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 characterised as deductive web mining (DWM). We suggest to use problem-solving methods (PSMs) as templates for composed services. We developed a multi-dimensional, ontology-based framework, and a collection of PSMs, which enable to characterise DWM applications at an abstract level; we describe several existing applications in this framework. We show that the heterogeneity and unboundedness of the web demands for some modifications of the PSM paradigm used in the context of traditional artificial intelligence. Finally, as simple proof of concept, we simulate automated DWM service composition on a small collection of services, PSM-based templates, data objects and ontological knowledge, all implemented in Prolog.

**Keywords:** Web services, web mining, problem-solving methods, ontologies.

## 1 INTRODUCTION

Composition of simple *web services* into sophisticated (distributed) applications recently became one of hot topics in computer science research. The area of application for composite services is potentially wide. While the focus is often on customer (e.g. travel) services, B2B transactions and financial services, the general paradigm appears as applicable even to back-end tasks such as gene analysis in bioinformatics [14] or web mining, the latter being the focus of this paper. *Deductive web mining* (DWM) as particular species of web mining was first introduced in [18]; it covers all activities where *pre-existing patterns* are matched with web data, be they of textual, graph-wise or, say, bitmap nature. DWM thus subsumes web information extraction, and differs from inductive web mining (such as association mining in web text), which aims at discovery of *previously unseen, frequent* patterns in web data. This does not mean that the ‘pre-existing patterns’ in DWM have necessarily been hand-crafted: inductive learning of patterns (or analogous structures/models) is merely viewed as an activity separate from DWM (‘reasoning’). Our current research attempts to combine both areas (web service composition and DWM), which, each of them separately, encompass a huge amount of research, while their intersection has been surprisingly left untouched. We attempt to show that abstract knowledge models are helpful for capturing the essence of DWM tasks. Starting from generic models (ontologies and problem-solving methods), we continue with their manually designed, semi-formal combinations and instantiations, and finish with automatically building operational simulation prototypes.

The structure of the paper is as follows. In section 2 we outline the history of our own DWM project named *Rainbow*, as initial motivation for DWM knowledge modelling. Section 3 presents our four-dimensional descriptive framework for DWM, called TODD, and a collection of ontologies associated with it. Section 4 explains the notion of problem-solving methods (PSMs) and shows its instantiation for DWM service modelling. Section 5 shows manually created models of existing applications, based on TODD and PSMs. Section 6 discusses the possibility of automatic, template-based (to read, PSM-based) web service composition in the DWM domain. Section 7 describes concrete simulation experiments done in this direction for the restricted task of pornography recognition. Finally, section 8 surveys some related projects, and section 9 wraps up the paper.

## 2 BACKGROUND: THE RAINBOW PROJECT

The *Rainbow* project (<http://rainbow.vse.cz>) represents a family of more-or-less independent DWM projects. Their unifying principles are commitment to *web service* (WSDL/SOAP) front-end and agreement on shared *upper-level ontology*. Furthermore, for each *application*, the developers also agree on a *domain* and share the source of training/testing *data*. Otherwise, the *formal principles* of analysis methods vary (from linguistic through statistical to e.g. graph theory), and so does the *representation of data* (such as free text, HTML trees or link topology). The overall

goal of the project is to verify the possibility of building web mining applications from semantically described components. There have been three use cases to the approach, each of them integrating several analysis tools:

1. Different *pornography-recognition* services, specialised in image bitmap analysis, HTML structure analysis, link topology analysis, META tag analysis and URL analysis, have been executed more-or-less standalone. Empirical tests however proved that the synergy of different methods reduces the overall error (from 10% for the best individual method to 6% for a combination of methods) [22]
2. Very simple analysis of *company information* (at the level of single pages) was designed to be executed and integrated via a web browser plug-in, which displayed the structured list of extracted information in a side bar [17]
3. Last, an application specialised in *bicycle offer extraction* has been sewn together, including (in addition to 'core' DWM tools): the *XML/full-text database engine AmphorA*, storing web pages as XHTML documents as source-data backend; a simple *control procedure* calling individual DWM tools, and integrating and saving the results; an instance of RDF repository *Sesame* (<http://www.openrdf.org>) for storing the results corresponding to a 'bicycle-offer' ontology (RDF Schema); finally, an (HTML+JSP) *semantic query interface* with pre-fabricated templates, shielding the user from the underlying RDF query language (SeRQL) and enabling a simple form of navigational retrieval [10].

Although the application-building effort itself has essentially been manual to date, the experience collected lead us to preliminary design of a semi-automatic composition method presented later in this paper. The first and the third application of *Rainbow* (beside other applications reported in the literature) have been re-described in terms of our novel knowledge modelling inventory, and the first (pornography recognition) was eventually subject of simulated experiments in service composition.

### 3 CONCEPTUAL FRAMEWORK AND ONTOLOGIES FOR DWM

#### 3.1 The TODD Framework

Based on experience from *Rainbow*, we proposed a framework that positions any DWM tool or service within a space with four dimensions:

1. Abstract *task* accomplished by the tool. So far, we managed to characterise any concrete DWM task as instance of either:
  - *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 semantic classes under consideration. It returns one or more semantic classes to which the object belongs.

The *Retrieval* of desired objects takes as input the *syntactic class* of object and *constraints* expressing its semantic class membership as well as (part-of and adjacency) relations to other objects; 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\***)”. *Retrieval* outputs the *identifiers* (addresses based on URIs, XPath expressions and the like) of relevant objects.

The *Extraction* task takes as input the semantic class of information to be extracted and the scope (i.e. an object) within which the extraction should take place; for example: “Extract the occurrences of Company Name within the scope of given Company Website”. *Extraction* outputs some (possibly structured, and most often textual) *content*. In contrast to Retrieval, it does not provide the information about precise location from where the content was extracted.

2. Syntactic class of *object* to be classified or retrieved, or from which information is to be extracted. Basic syntactic classes are defined in the Upper Web Ontology (see section 3.2). The assumption is that the syntactic class of object is always known, i.e. its assignment is not by itself subject of DWM.
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* to which the task is specific.

We thus denote the framework as ‘*task-object-data(type)-domain*’ (TODD). Its dimensions are to high degree independent, e.g. *object class* 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. Clearly, not all points of the 4-dimensional space are meaningful, for instance, a META tag content cannot directly be used to classify a hyperlink.

### 3.2 The Collection of *Rainbow* Ontologies

In parallel with development of *Rainbow* tools, abstract ontologies were also designed. In general, there are two kinds of classes in *Rainbow* ontologies — *syntactic* and *semantic*. Syntactic classes currently considered are e.g. *Document*, *Document-Fragment*, *HyperLink* or *Phrase*; semantic classes are e.g. *ProductCatalogue*, *Leaf-Document*, *ProductDescription* or *Sentence*. As outlined above, semantic classes differ from syntactic ones in the sense that their identification is subject of analysis, while the identification of syntactic classes is assumed to be known in advance (say,

no *Rainbow* tool should be developed in order to distinguish a physical page from a collection of pages). Every semantic class is subconcept of some general syntactic classes. In addition to concepts, there are also relations. Among the most widely used relations in *Rainbow* ontologies is the transitive *part-of* relation, e.g. *Product-Description* may be *part-of* a *ProductCatalogue*. Concepts can also be *adjacent* to each other, they may be *identified-by* some other concepts etc.

The three dimensions of the TODD model (i.e. apart from the task dimension), namely the distinction of *data types*, *object (syntactic) classes* and application *domains* suggest a natural decomposition of the system of ontologies into four layers as depicted in Fig. 1: the Upper Web Ontology (UWO), partial generic web models, partial domain-dependent web models, and integrated domain web ontologies. The upmost layer (i.e. UWO) contains the *object syntactic classes* themselves. Furthermore, the upper two layers ('generic models') are domain-independent and therefore reusable by applications from all *domains*, while the lower two layers ('domain-specific models') add information specific to the domain of analysis, e.g. OOPS ('organisations offering products and services') sites or web pornography. Finally, the outer two layers<sup>1</sup> contain concepts that are independent of the *data type* used for their recognition, while the inner two ('partial models') contain data-type-dependent concepts. Let us now characterise each of the four layers, in turn.

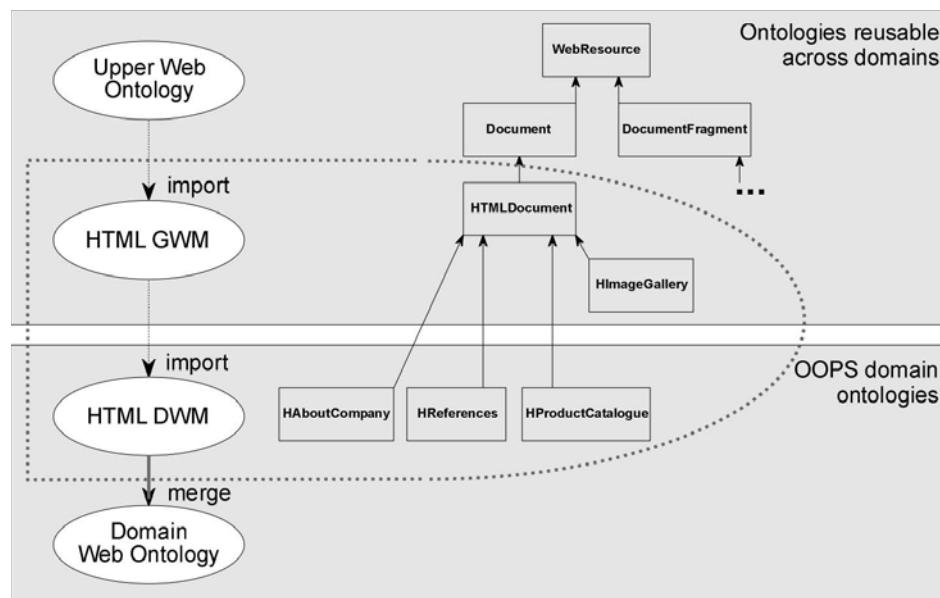


Fig. 1. Structure of the *Rainbow* ontology system shown on HTML analysis example

<sup>1</sup> We might jointly call them e.g. 'holistic models', in contrast to 'partial' ones.

**Upper Web Ontology.** The abstract *Upper Web Ontology* (UWO) provides a hierarchy of common web-related concepts and relations that are shared by different analysis types and application domains. It defines the most generic syntactic classes, which are likely to be frequently reused across individual analysis tools: *Document*, *Document Collection*, *Document Fragment*, *Hyperlink* and the like.

**Partial Generic and Domain(-Specific) Web Models.** For each way of analysis, *partial web models* (generic and domain-specific) occupy the middle layers of the *Rainbow* ontology system. Concepts and relations defined in these models represent the (syntactic and semantic) classes specific to one data type. The partial web models consist of a *generic* and *domain-dependent* part. Elements introduced in the generic model are based on the UWO and are reusable across different application domains. On the other hand, for each data type there might be domain models specialised in different application domains. All of these domain models are then based on a single generic model and the common UWO. Concepts from the generic and domain models mostly correspond to *semantic classes* of resources, but new *syntactic classes* may be defined as well. In Fig. 1, the generic model and OOPS domain model for HTML analysis are depicted within the dashed area. Examples of concepts from these models and the UWO are shown on the right; class names are prefixed with corresponding data types: 'H' for HTML structure, 'T' for topology etc. The HTML Generic Web Model (i.e. the generic model considering HTML code as subject of analysis) contains domain-independent concepts such as *HTMLDocument* or *HImageGallery*. For the OOPS domain, these are further refined to concepts related to product and service offering, such as 'HTML document with company profile' (*HAboutCompany*), 'HTML document with references' (*HReferences*) or 'HTML document with product catalogue' (*HProductCatalogue*).

**Domain(-Specific) Web Ontologies.** In our approach, domain-specific web ontologies can be built via *merging* the class hierarchies from domain-specific partial web models. We studied the possibility of using *Formal Concept Analysis* (FCA) for this purpose, which potentially yields new classes in addition to those inherited from the merged ontologies. The resulting class hierarchy is no longer restricted to a single data-type view and is included in a *Domain Web Ontology* (DWO), as depicted in Fig. 1.

For example, in the graph output by FCA based on web objects annotated with HTML and Topology concepts, we identified one new (no-name) class which was a combination of *THub* and *HProductCatalogue*. It represents the common notion of product catalogue referring to child documents with detailed information about individual products, and may be a meaningful addition to the ontology. More details about the merging method are in [9].

The *Rainbow* collection of ontologies was implemented in DAML+OIL (predecessor of OWL), mostly using the expressivity of RDF/S only. It contains 20 classes in UWO, over 100 classes in partial generic models, and 24 classes in partial

(OOPS) domain models. The ontologies have never been used operationally; they rather served as proof of concept for the TODD model and also as starting point for building a smaller but operational ontology (in Prolog) for the purpose of service composition simulations, see section 7.

## 4 PROBLEM SOLVING METHODS FOR DWM

### 4.1 Problem-Solving Modelling Classics

The first abstract model of knowledge-based problem solving (Problem Solving Method – PSM), from which many successors took inspiration, was probably the model of *heuristic classification* formulated in mid 80s by Clancey [6], see Fig. 2. It represents an abstraction over the reasoning structure of numerous diagnostic expert systems from different domains. Its essence are three ‘primitive’ *inferences* called ‘abstract’, ‘match’ and ‘specialise’, whose inputs/outputs are denoted as *knowledge roles*: ‘Observables’, ‘Variables’, ‘Solution Abstractions’ and ‘Solutions’. The knowledge roles are, in a concrete application, mapped on domain concepts. For example, a medical expert system for treatment recommendation might acquire patient lab tests and other findings as ‘Observables’, it would *abstract* more general notions such as ‘obesity’ or ‘hypertension’ from them, *match* these with general categories of drugs such as ‘diuretics’ or ‘ $\beta$ -blockers’, and, finally, *specialise* the drug groups to concrete substances, drug brands and dosage, according to the context, e.g. availability on market and coverage by insurance.

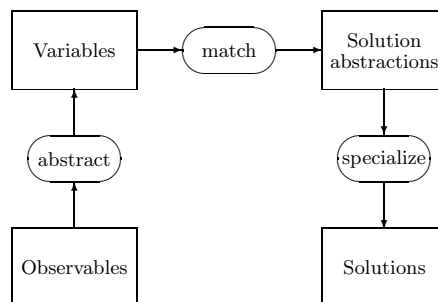


Fig. 2. Inferences and knowledge roles in heuristic classification model

Later, the CommonKADS methodology [15], fully developed in mid 90s, formulated complex guidelines for the design and usage of PSMs in the context of knowledge-based system development. The knowledge-level description of a KBS is viewed as consisting of three interconnected layers. (1) The *domain layer* describes the relevant domain concepts and relations independent of their use for reasoning. (2) The *inference layer* specifies the flow of inferences and data but not the control

flow. It is typically expressed using inference diagrams such as that of heuristic classification. Knowledge roles in the diagram are mapped to concepts from the domain layer. (3) The *task layer* specifies the decomposition of tasks to subtasks and the algorithmic control structure. The lowest level of tasks in the decomposition tree corresponds to inferences from the previous layer.

#### 4.2 Library of Deductive Web Mining PSMs

Let us now present a collection of eight PSMs for DWM (namely, for the Classification, Retrieval and Extraction tasks), in a style inspired with CommonKADS. It is rather tentative, yet seems to cover a large part of realistic cases; examples will be given in section 5.

For *Classification* we consider three PSMs. *Look-up based Classification* amounts to picking the whole content of the given object (cf. the Overall Extraction PSM below), 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.

For *Extraction*, there will be again three PSMs, rather analogous to those of Classification. *Overall Extraction* amounts to picking the whole content of the given object. *Compact Extraction* corresponds to a single inference based on possibly complex computation, which directly returns the content of specific sub-object/s of the given ‘scope’ object. Finally, *Structural Extraction* corresponds to extraction of information from an object via focusing on its certain sub-objects. Such objects have first to be *retrieved*, then lower-grained *extraction* takes place, and, finally, multiple content items possibly have to be *integrated*. Structural Extraction is thus equally recursive as Structural Classification.

Finally, let us first introduce two PSMs for the *Retrieval* task. The upper inference structure<sup>2</sup> in Fig. 3 corresponds to Direct Retrieval and the lower one to Index-Based Retrieval, respectively. The names of inferences (in ovals) are mostly borrowed from the CommonKADS library [15], while the knowledge roles are more DWM-specific. In *Direct Retrieval*, potentially relevant objects are first retrieved based on structural (parthood and adjacency) constraints, and then classified. Ob-

---

<sup>2</sup> We do not show inference structures for Classification and Extraction, due to limited space as well as due to incompatibility of their structural variants with the CommonKADS notation, see below.

jects whose classes satisfy the class constraints are the output of the method. In the absence of class constraints, the method reduces to the ‘specify’ inference. In *Index-based Retrieval*, the (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 considered as separate from the web space itself), possibly considering structural constraints (provided structural information is stored aside the core index).

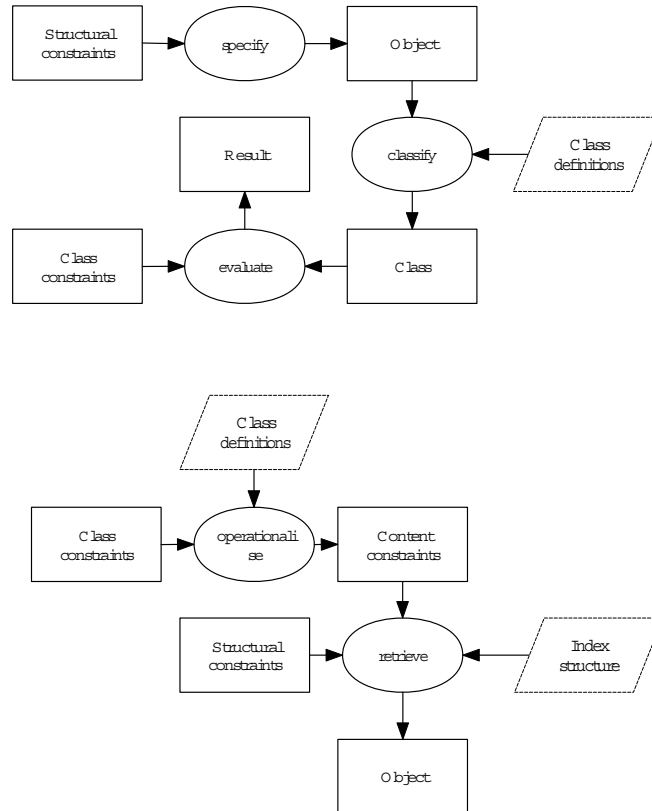


Fig. 3. Inference diagrams of Retrieval PSMs

An interesting issue related to the representation of above PSMs is the possible interaction of different ‘time horizons’ in one application; static roles may become dynamic when changing the time scale. For example, a typical DWM application may first build an index of a part of the website (or learn class definitions from a labelled subset of objects), and then use the index to efficiently retrieve objects (or use the class definitions to classify further objects). This interaction deserves further study.

### 4.3 Traditional vs. DWM Classification

Among the three tasks, it is *Classification* that is most appropriate for comparison with existing PSM research. Classification problem solving was recently systematised by Motta&Lu [13]. Their taxonomy of classification problems is mainly derived from the presence (or absence) of a few key features:

1. *Whether the goal is to find one, all or the best solution.* This distinction can well be ported to the DWM context.
2. *Whether all observables are known at the beginning or are uncovered opportunistically (typically at some cost) during the problem solving process.* In DWM, the latter is typically the case (provided we interpret ‘observables’ as the web objects themselves); the cost is however only associated with download/analysis time, and its increase is smooth—unlike e.g. medical applications, where addition of a single examination may lead to abrupt increase of (financial or social) cost.
3. *Whether the solution space is structured according to a refinement hierarchy.* Presence of class hierarchy is quite typical in DWM; in the *Rainbow* project, it is reflected in concept taxonomies that constitute our ontology, see Section 3.
4. *Whether solutions can be composed together or each presents a different, self-contained alternative.* We believe that in DWM, elementary classification will mostly be carried out over disjoint classes, but can be superposed by multi-way classification with non-exclusive class taxonomies. We discuss this option below, in connection with the *refine* inference of Heuristic Classification.

Motta&Lu [13] also formulated a generic task-subtask decomposition *template*, which can be instantiated for different task settings:

1. First the observations have to be verified whether they are legal (*Check*).
2. All legal observations (*(feature,value)*-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*).

Compared to this generic Classification template, our notion of DWM 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).

We can also compare Structural Classification with Clancey’s *Heuristic Classification* (HC) mentioned earlier. In (DWM) Structural Classification, the *abstract*

inference is replaced with *classify* inferences applied on related (contained and/or adjacent) objects; this is due to the ‘object-relation-object’ (rather than ‘object-feature-value’) character of web data representation. The *match* inference from HC corresponds to ‘evaluation of global classification patterns’. Finally, a *refinement* from general to case-specific solution might rather have the form of classification according to *multiple hierarchies* in DWM (e.g. in data-type-specific ontologies). The object is then assigned to the class that is defined as intersection of both original classes. For example, in the pornography application (see section 5), an object classified as Image Gallery may also be independently classified as Scarce Text Fragment, which yields the class Porno Index.

## 5 RE-ENGINEERING TODD-BASED DESCRIPTIONS

Let us now describe concrete applications in terms of the TODD framework, including the mapping of tasks to PSMs. We only describe the *Rainbow* pornography-recognition application [22] (Table 1) and the bootstrapping approach to website information extraction by Ciravegna et al. [5] (Table 2). More such descriptions (for the *Rainbow* bicycle application and for two more third-party applications) are in [18].

### 5.1 Syntax of the Semi-Formal Language

We use an ad hoc semi-formal language with Prolog-like syntax. Its building blocks are *decompositions* of tasks (‘heads of clauses’) to ordered sequences of subtasks (‘bodies of clauses’). Individual task descriptions (‘literals’) look as follows, respectively:

```

Cla?(<obj_var>, <obj_class>, <data_type>, <domain>, <classes>)
Ret?(<obj_var>, <obj_class>, <data_type>, <domain>, <constraints>)
Ext?(<obj_var>, <obj_class>, <data_type>, <domain>, <content>)

```

The ‘predicate’ (task name) corresponds to the first dimension in the TODD framework. An extra letter is used to distinguish the PSMs introduced in the previous sections: **ClaS** for Structural Classification, **ClaL** for Look-up based Classification, **ClaC** for Compact Classification; **RetD** for Direct Retrieval, **RetI** for Index-based Retrieval; **ExtS** for Structural Extraction, **ExtC** for Compact Extraction and **ExtO** for Overall Extraction. From the nature of the PSMs follows that each ClaS task can be decomposed to a structure including (among other) one or more subtasks of type Classification; analogously, each ExtS task can be decomposed to a structure including one or more subtasks of type Extraction. In the examples, the ‘unification’ of a ‘goal’ with a ‘clause head’ is always unique; the representation is only ‘folded’ for better readability.

The remaining three dimensions of the TODD model are reflected by the ‘arguments’ <obj\_class>, <data\_type> and <domain>. Finally:

- **<obj\_var>** is 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. We use object variables (and object classes) even for Extraction; however, here they only refer to the scope of extraction, not to a ‘current’ object as in Classification and Retrieval.
- **<classes>** is the list of classes distinguished in the classification task (beside named classes, we use the symbol **@other** for a ‘complement’ class).
- **<constraints>** is the list of logical expressions determining the set of objects to be retrieved; they correspond to the knowledge roles Class Constraints (class membership restrictions) and Structural Constraints (parthood/adjacency restrictions).
- **<content>** is the list of types of content information (datatype properties in semantic web terminology) to be extracted.

For simplicity, we ignore strictly procedural constructs such as selections or iterations, as well as the cardinality of input and output.

## 5.2 Descriptions of Applications

The upper level of the *pornography-recognition* process is an instantiation of the *Structural Classification* PSM as discussed in the previous section. 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 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.

The approach to information extraction described in [5] and implemented in the *Armadillo* system heavily relies on knowledge reuse, thanks to the well-known redundancy of WWW information. We only describe the most elaborated part of the method, targeted at extraction of person names (additionally, various personal data and paper titles are extracted for the persons in question). First, potential names are cropped from the website, and checked against binary classification tools such as context-based named-entity recognisers (Compact Classification), as well as against public search tools (namely, online bibliographies, homepage finders and general search engines) that produce the same binary classification (person name -

Table 1. TODD-based description of pornography application

```

ClaS(DC, DocCollection, _, Pornography, [PornoSite,@other]) :-
  RetL(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]).

```

yes/no) as by-product of offering information on papers or homepages (i.e. Index-based Retrieval). Furthermore, for the results of general web search, the page from the given site is labelled as homepage if the name occurs in a particular (typically, heading) tag. The seed names obtained are further extended by names co-occurring in a list or in the same column of a table. Finally, potential person names from anchors of intra-site hyperlinks are added.

### 5.3 Discussion

The models of third-party applications such as the presented *Armadillo* example were created based on the text of research papers. We cannot naturally view them as complete, as the textual descriptions in papers are usually simplified (e.g. due to space limitations) as well. The whole modelling exercise was (apart from reading the text itself) always a matter of 20-30% minutes, and no critical problems have been encountered. In general, the TODD model and our collection of PSMs proved well-applicable on pre-existing DWM tasks and tools. Typically, the core task in the applications was either classification or extraction, which occurred recursively for different objects and was interleaved with retrieval of appropriate objects. The only model encompassing all three task types was indeed that of *Armadillo* application; there phrases of a certain semantic class ('potential person name') are first retrieved, then false candidates are filtered out via classification, yielding true person names, and finally the textual content is extracted from the names so as to be used in subsequent tasks.

Table 2. TODD-based description of an Armadillo application

```

ExtS(DC, DocCollection, _, CSDept, [names]) :-
  RetD(P1, Phrase, text, General, [P1 part-of DC, PotentPName(P1)]),
  % named entity recognition for person names
  ClaC(P1, Phrase, text, General, [PName,@other]),
  % use of public search tools over papers and homepages
  RetI(P2, Phrase, freq, Biblio, P1 part-of P2, PaperCitation(P2)),
  RetI(D, Document, freq, General,
    [P1 part-of D, D part-of DC, PHomepage(D)]),
  RetD(DF1, DocFragment, freq, General,
    [Heading(DF1), DF1 part-of D, P1 part-of DF1]),
  ExtO(P1, Phrase, text, General, [names]),
  % co-occurrence-based extraction
  RetD(DF2, DocFragment, html, General,
    [ListItem(DF2), DF2 part-of DC, P1 part-of DF2]),
  RetD(DF3, DocFragment, html, General,
    [ListItem(DF3), (DF3 below DF2; DF2 below DF3)]),
  ExtS(DF3, DocFragment, text, General, [names]),
  RetD(DF4, DocFragment, html, General,
    [TableField(DF4), DF4 part-of DC, P1 part-of DF4]),
  RetD(Q, DocFragment, html, General,
    [TableField(DF5), (DF5 below DF4; DF4 below DF5)]),
  ExtS(DF5, DocFragment, text, General, [names]),
  % extraction from links
  RetD(DF5, DocFragment, html, General,
    [IntraSiteLinkElement(DF5), DF5 part-of DC]),
  ExtS(DF5, DocFragment, text, General, [names]),
  ...
% extraction of potential person names from document fragments
ExtS(DF, DocFragment, text, General, [names]) :-
  RetD(P, Phrase, text, General,
    [DF contains P, PotentialPersonName(P)]),
  ExtO(P, Phrase, text, General, [names]).

```

## 6 TEMPLATE-BASED COMPOSITION OF DWM SERVICES

### 6.1 Template-based Approach to Web Service Composition

While the abstraction of ('decomposable') models from legacy applications is definitely a task to be carried out by a human, the construction of such models (and even the on-the-fly design of operational applications) from properly described components might be, to some degree, within the reach of automated service composition methods. In the research on web service composition, three alternative research

streams can currently be identified:

1. *Programming in the large*, i.e. composition of services by (more-or-less) traditional procedural programming in languages such as BPEL4WS (<http://www-128.ibm.com/developerworks/library/ws-bpel>). The main advantage is perfect control over the choice and linkage of different services at design time. This however, on the other hand, entails a lower 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 unpredictable if all conditions are not perfectly specified.
3. *Template-based* composition, in which concrete services are filled in run time into pre-fabricated templates [11, 21].

More specifically, [21] suggested to view web service composition templates as analogy to PSMs, and to view the *configuration* of the template again as a kind of reasoning task, that of *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. The configuration process is carried out by a so-called *broker* tool, and employs the *propose-critique-modify* (PCM) reasoning method, taking advantage of *background knowledge* of the broker. 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.

Based on the work by Motta&Lu [13], we assume that classification services can be described in a single template. This template (see Section 4.3) consists of five steps: *Check*, *MicroMatch*, *Aggregate*, *Admissibility* and *Selection*.

Example values of *Admissibility* parameter are (see [21] for more):

- *weak-coverage*: All  $\langle \text{feature}, \text{value} \rangle$  pairs in the observations are *consistent* with the feature specifications of the solution.
- *strong-coverage*: All  $\langle \text{feature}, \text{value} \rangle$  pairs in the observations are *consistent* with the feature specifications of the solution and *explained* by them.
- *strong-explanative*: All  $\langle \text{feature}, \text{value} \rangle$  pairs 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).

## 6.2 *Rainbow* Applications as Composite Web Services

For the first truly composite application of *Rainbow*, a few hundred lines of Java code sufficed to weave together the tools (web services) cooperating in the analysis of bicycle websites [10]. However, with increasing number of available tools, composition by traditional programming soon 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. The collection of *PSMs* abstracted from real *deductive web mining* applications, explained in section 4.2, could be basis for templates. Furthermore, individual components (services) can be positioned in the *TODD space*, which could, among other, play a similar role as the space of template parameters from [21].

An important point is to evaluate the possibility to adapt the parametric design approach from [21] to the (specific features of) web analysis *PSMs*; this is the subject of the next subsection. Main focus will be on *classification*, which is the only task considered in [21] and also one of tasks studied in this paper.

### 6.3 DWM Service Configuration as Parametric Design

As we outlined in section 4.2, 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 fixed feature template* such as parametric design. 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 the paper, we mostly focus on the first solution. Although it obviously oversimplifies many aspects of real-world settings, it is easier to design and implement in its rudimentary form and also remains more faithful to the original parametric design concept. Table 3 shows five templates for the classification task (encoded in Prolog): the first amounts to single classification of the current object, the second aggregates two different ways of classifying the current object, the third and the fourth rely on another object (sub-object or adjacent object) in order to classify the current object, and the fifth combines direct classification of current object with its structural classification (via classification of another object). The arguments of the `templ` clauses amount to the following: template *identifier* (`sc#`), composed service *signature*, list of component services *signatures* (one for each ‘empty slot’), list of ontological *constraints* among object classes. Each signature (i.e. `s()` structure) first defines the *task type* accomplished by the service; the numbers (0, 1, ...) have the semantic of variables that either refer to objects or to slots themselves (0 being the ‘start-up’ object of the composed service), and the Prolog variables `C#` correspond to classes of these objects.

In addition to classification (`cla`) and retrieval (`ret`) services types, the templates also include slots for *auxilliary services* needed to accomplish the target classification task. As types of auxilliary services, we so far considered aggregation (`agr`), transformation (`tsf`) and iteration (not shown here). For example, the presence of sub-object of certain class determines the class of the super-object in a certain way. In particular, the certainty factor of classification of sub-object is *transformed* to certainty factor of classification of super-object; the data flow between the services is indicated by the `ref(SourceService,SourceObject)` construct. Similarly, classification of the same object by different methods has to be compared and the result computed via *aggregation* (e.g. combining the certainty factors).

In more detail, the body of third template declares that, given an input object `no.0` of class `C1`, we can (1) apply a service that can retrieve a ‘target’ object (`no.1`)

of class **C4** within some ‘source’ object of class **C3** (subclass of **C1**); we instantiate the ‘source’ object with object no.0; (2) then apply on the (retrieved) object no.1 a classifier that is capable of classifying any object of class **C5** (superclass of **C4**) into class **C6** or its complement; (3) and finally, transform the result of classification of object no.1 (into class **C6** or its complement, via the second service in the sequence) into the result of classification of object no.0 into class **C2** (as target class to be determined) or its complement.

Table 3. Sample templates for classification task

```
templ(sc1,s(c1a,0,0,C1,C2),
  [s(c1a,0,0,C3,C4)], [subclasseq(C3,C1),subclasseq(C4,C2)]).
templ(sc2,s(c1a,0,0,C1,C2),
  [s(c1a,0,0,C3,C4),s(c1a,0,0,C5,C4),s(agr,[ref(1,0),ref(2,0)],0,C4,C4)],
  [subclasseq(C3,C1),subclasseq(C5,C1),subclasseq(C4,C2)]).
templ(sc3,s(c1a,0,0,C1,C2),
  [s(ret,0,1,C3,C4),s(c1a,1,1,C5,C6),s(tsf,ref(2,1),0,C6,C2)],
  [subclasseq(C3,C1),rel(part,C4,C3),subclasseq(C4,C5)]).
templ(sc4,s(c1a,0,0,C1,C2),
  [s(ret,0,1,C3,C4),s(c1a,1,1,C5,C6),s(tsf,ref(2,1),0,C6,C2)],
  [subclasseq(C3,C1),rel(adj,C4,C3),subclasseq(C4,C5)]).
templ(sc5,s(c1a,0,0,C1,C2),
  [s(c1a,0,0,C3,C4),s(ret,0,1,C5,C6),s(c1a,1,1,C7,C8),
  s(tsf,ref(3,1),0,C8,C4),s(agr,[ref(1,0),ref(4,0)],0,C4,C4)],
  [subclasseq(C3,C1),subclasseq(C5,C1),rel(part,C6,C5),
  subclasseq(C6,C7),subclasseq(C4,C2)]).
```

## 7 SIMULATION OF TEMPLATE CONFIGURATION & EXECUTION

### 7.1 One-Shot Setting Without Broker Knowledge

As seen from the above discussion, there are two main differences from the original approach to web service composition using parametric design (section 6.1):

- We do not have a single template but a choice of multiple ones
- For the individual template slots, we don’t deal with a clearly defined family of different methods (variations of a method) but with a theoretically borderless space of applicable tools.

It was therefore natural to start with a fragment of the original Parametric Design model only, namely, with its *Propose* and *Verify* (in the sense of service execution) phases only. Although *broker knowledge* would be desirable (and was used in previous work [21]) for the *Propose* phase, it was not indispensable, and we could

perform service configuration based on the *signatures* in the templates only. The use of broker knowledge is only discussed in the following subsection.

We implemented a collection of simple programs in Prolog consisting of:

1. the five templates discussed in the previous sections
2. four simulated 'websites' (inspired by real ones), in clausal form, an incomplete example is in Table 4
3. simplified services (incl. auxilliary ones) equipped with meta-data
4. a *configuration tool* that selects and fills in the templates based on service meta-data
5. an *execution tool* that executes the filled template for a given data object
6. an 'ontology' (derived from that described in section 3.2) containing definitions of basic concepts needed for the composition and/or execution phase.

Table 4. Incomplete example of simulated 'website' in clausal form

```

site(s2). % website
class(s2,nonporno).
page(p23). % page with 2 html fragments and 1 picture
url_of(u23,p23).
url_terms(u23,[hot]).
part(p23,s2).
linkto(p21,p23).
textprop(p23,0.8). % proportion of text on page
part(f231,p23).
html_frag(f231). % fragment 1
part(i2311,f231).
image(i2311).
body_color(i2311,0.1).
part(f232,p23).
html_frag(f232). % fragment 2

```

The whole setting is very rudimentary. The service slots in templates are limited to a single object on input and on output. The classification services only perform binary classification, i.e. they output a certainty factor for a single class on output (distinguishing it from its complement). The classes amount to pornography-relevant ones, such as pornography-containing site or pornography content page.

Table 5 shows two examples of service *composition*. The first one suggests two ways of classifying a document as `pornoContentPage`, based on two different templates: either by directly classifying the document or by first retrieving and classifying its follow-up document and then transforming the certainty factor. The second one suggests to classify a site by retrieving and classifying its hub page.

Table 5. Service composition dialogue

```
?- propose(cia, document, pornoContentPage).
Number of solutions: 2
Template:      sc1
Configuration:
  s(cia, 0, 0, document, pornoContentPage, cia_por_url)
Template:      sc4
Configuration:
  s(ret, 0, 1, document, document, ret_follows)
  s(cia, 1, 1, document, pornoContentPage, cia_por_url)
  s(tsf, ref(2, 1), 0, pornoContentPage, pornoContentPage, tsf_porno2)

?- propose(cia, doc_coll, porno_coll).
Number of solutions: 1
Template:      sc3
Configuration:
  s(ret, 0, 1, doc_coll, localhub, ret_localhub)
  s(cia, 1, 1, document, pornoContentPage, cia_por_url)
  s(tsf, ref(2, 1), 0, pornoContentPage, porno_coll, tsf_porno1)
```

The composed services can then be *executed*. For example, we can call the already configured template `sc4` using the ID of input object, its initial class (e.g. just `document` as syntactic class) and the certainty factor of this class (it should be 1 in this case). The execution engine returns the ID of output object (for a classification task, it is identical to input object), its suggested class (`pornoContentPage`), and the certainty factor of this refined class. The results can be compared with 'gold standard' data and thus provide a simple form of *verification* of the configuration.

## 7.2 Towards a Complete Parametric Design Cycle

While the initial configuration of the template (Propose phase) could be accomplished using 'semantic signatures' of individual services only, its subsequent automated *modification* requires additional knowledge. Tentative examples of such knowledge (albeit still meant for the Propose phase) have been formulated in [19]. Compared to broker knowledge from [21], they also include template selection and reformulation knowledge in addition to slot-filling knowledge. 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*  $\succ$  *URL*  $\succ$  *topology*, *free\_text*  $\succ$  *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.

Let us further show a hypothetical scenario of the use of broker knowledge, in connection with the pornography-recognition application discussed in section 5. Let us assume a web pornography ontology grafted upon the *Upper Web Ontology* and 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 was 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.

## 8 RELATED WORK

In accordance with the structure of the paper, we divide the related work overview into two parts, related to conceptual models of web space and to PSM-based modelling of web analysis, respectively.

In the *OntoWebber* project [8], a ‘website ontology’ was designed. It was however biased by its application on portal building (i.e. ‘website synthesis’), and thus did not fully cover the needs of automated analysis; moreover, the problem-solving side of modelling was not explicitly addressed. The same holds for semantic conceptual models of web space used for adaptive hypermedia design, see e.g. [16], which generally rely on a combination of domain model, user model and adaptation model; automated analysis, on the other hand, cannot have ambitions to reconstruct such models and has to rely on lower-level features.

Until recently, PSMs have been understood as specific for knowledge-intensive but ‘data-temperate’ tasks. A few PSMs for *data-intensive tasks* have however also been designed. 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. In contrast, the *Armadillo* system [5] attempts to integrate many website analysis methods; it currently relies on sequences manually composed from scratch by the user, although a template-based solution is also being envisaged. Besides, PSM-based solution has also been developed for task configuration in Knowledge Discovery in Databases (KDD) [7]; however, although some aspects of modelling are similar, the nature of web data is significantly different from that of tabular data.

## 9 CONCLUSIONS AND FUTURE WORK

We demonstrated that web service composition, and specifically its variant based on problem-solving modelling, can be applied to deductive web mining. However, the task demanded a significant modification of principles used in previous domains. In particular, due to the nature of web as underlying data structure, service templates tend to involve recursion, which impacts the process of template-filling. On the other hand, the *TODD* framework, although originally developed independently, easily became the cornerstone of service descriptions created manually as well as of the tentative method of automated composition.

The current prototype of composition tool was only meant for the sake of initial experiment on (semi-)artificial data. We plan to proceed to real data when switching to a functional architecture incorporating independently-developed (often third-party) tools, as envisaged in the *Rainbow* project. In addition to the multi-template model, we also expect to implement and test the solution based on automatic template restructuring. Future research also includes specification of templates for other DWM tasks, in particular those with nature of *extraction*, taking models of applications from section 5 as starting point. Finally, we consider to align our approach with the *WSMO* project (<http://www.wsmo.org>), which also partially applies the PSM paradigm to web service composition.

The research follows up with results obtained in the CSF project no. 201/03/1318 (“Intelligent analysis of WWW content and structure”), and is partially supported by the Knowledge Web Network of Excellence (IST FP6-507482).

## REFERENCES

- [1] IBROW homepage, <http://www.swi.psy.uva.nl/projects/ibrow>
- [2] ABASOLO, C. et al.: Libraries for Information Agents. IBROW Deliverable D4, IIIA, Barcelona, March 2001. 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] 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.
- [6] CLANCEY, W. J.: Acquiring, representing, and evaluating a competence model of diagnostic strategy. In: The Nature of Expertise, Lawrence Erlbaum Press 1988.
- [7] ENGELS, R.—LINDNER, G.—STUDER, R.: Providing User Support for Developing Knowledge Discovery Applications; A Midterm report. In: S. Wrobel (Ed.) *Themenheft der Künstliche Intelligenz*, (1) March, 1998.
- [8] JIN, Y.,—DECKER, S.,—WIEDERHOLD, G.: OntoWebber: Model-Driven Ontology-Based Web Site Management. In: 1st International Semantic Web Working Symposium (SWWS’01), Stanford University, Stanford, CA, July 29-Aug 1, 2001.
- [9] LABSKÝ, M.—SVÁTEK, V.: Ontology Merging in Context of Web Analysis. In: Workshop DATESO03, TU Ostrava, 2003.
- [10] LABSKÝ, M. et al.: Information Extraction from HTML Product Catalogues: from Source Code and Images to RDF. In: 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI’05), IEEE Computer Society, 2005.
- [11] MANDELL, D. J.—MCLRAITH, S. A.: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. In: Proc. ISWC2003.
- [12] MARTIN, D. et al.: OWL-S 1.0 Release. Online at <http://www.daml.org/services/owl-s/1.0/>.
- [13] MOTTA, E.—LU, W.: A Library of Components for Classification Problem Solving. In: Proceedings of PKAW 2000, The 2000 Pacific Rim Knowledge Acquisition Workshop, Sydney, Australia, December 2000.
- [14] SABOU, M.—WROE, C.—GOBLE, C.—MISHNE, G.: Learning Domain Ontologies for Web Service Descriptions: an Experiment in Bioinformatics. In: The 14th International World Wide Web Conference (WWW2005), Chiba, Japan.
- [15] SCHREIBER, G., et al.: Knowledge Engineering and Management. The CommonKADS Methodology. MIT Press, 1999.

- [16] SEEFELDER DE ASSIS, P.,—SCHWABE, D.: A Semantic Meta-model for Adaptive Hypermedia Systems. In: Adaptive Hypermedia 2004, LNCS 3137, 2004.
- [17] SVÁTEK, V.—KOSEK, J.—LABSKÝ, M.—BRÁZA, J.—KAVALEC, M.—VACURA, M.—VÁVRA, V.—SNÁŠEL, V.: Rainbow - Multiway Semantic Analysis of Websites. In: 2<sup>nd</sup> Int'l DEXA Workshop on Web Semantics (WebS03), IEEE 2003.
- [18] SVÁTEK, V.—LABSKÝ, M.—VACURA, M.: Knowledge Modelling for Deductive Web Mining. In: Int'l Conf. Knowledge Engineering and Knowledge Management (EKAW 2004), Whittlebury Hall, Northamptonshire, UK. Springer Verlag, LNCS, 2004.
- [19] SVÁTEK, V.—TEN TEIJE, A.—VACURA, M.: Web Service Composition for Deductive Web Mining: A Knowledge Modelling Approach. In: Znalosti 2005, High Tatras, 2005.
- [20] SVÁTEK, V.—VACURA, M.: Automatic Composition of Web Analysis Tools: Simulation on Classification Templates. In: First International Workshop on Representation and Analysis of Web Space (RAWS-05), online <http://CEUR-WS.org/Vol-164>.
- [21] TEN TEIJE, A.—VAN HARMELEN, F.—WIELINGA, B.: Configuration of Web Services as Parametric Design. In: Int'l Conf. Knowledge Engineering and Knowledge Management (EKAW 2004). Springer Verlag, LNCS, 2004.
- [22] VACURA, M.: Recognition of pornographic WWW documents on the Internet (in Czech), PhD Thesis, University of Economics, Prague, 2003.

**Vojtěch Svátek** is lecturer at the Department of Information and Knowledge Engineering, University of Economics, Prague. His main areas of research are knowledge modelling and knowledge discovery from databases and texts.

**Miroslav Vacura** is post-doctoral researcher at the Department of Information and Knowledge Engineering, University of Economics, Prague. His main areas of research are foundational ontologies and classification of text and images.

**Martin Labský** is PhD student at the Department of Information and Knowledge Engineering, University of Economics, Prague. His main area of research is web information extraction.

**Annette ten Teije** is lecturer at the Department of Artificial Intelligence of Vrije Universiteit Amsterdam. Her interests include approximate reasoning, formalisation of medical knowledge, configuration of reasoning methods, and diagnosis.