Automatic Composition of Web Analysis Tools: Simulation on Classification Templates

Vojtěch Svátek and Miroslav Vacura

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

Abstract. Template-based composition of web services is considered as useful middle-way between their manual 'programming in the large' and fully automatic 'AI-planning-style' composition. This is also relevant for applications analysing the content and structure of the web space. As simple proof of concept, we simulate this approach on a collection of services, templates, data objects and ontological knowledge, all implemented in Prolog. The underlying task is multi-way recognition of sites containing pornography, understood as instance of classification task.

1 Introduction

Composition of *web services* into distributed applications recently became one a hot topics in computer science research. Three alternatives can be identified:

1. *Programming in the large*, i.e. composition of services by procedural programming in languages such as BPEL4WS[1]. Its main advantage is perfect control over the choice and linkage of different services, at design time. This however 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.
3. *Template-based* composition, in which concrete services are filled in run time into pre-fabricated templates [8]. Ten Teije et al. [14] suggested to view composition templates as *problem solving methods* (PSMs), i.e. abstract descriptions of knowledge-based reasoning scenarios, which had been studied in the knowledge modelling community for nearly two decades (see e.g. [5, 10]). In addition, they suggested to view the *configuration* of the template again as a kind of reasoning task: that of *parametric design*. The configuration process is carried out by a *broker* tool, and employs the *propose-critique-modify* (PCM) reasoning method, taking advantage of *background knowledge*.

[1] http://www-128.ibm.com/developerworks/library/ws-bpel

The way information is presented on the web combines multiple types and representations of data. Different methods of web data analysis may provide complementary and/or supplementary information. Reducing the analysis on a single method may thus lead to significant information loss. On the other hand, a monolithic application encompassing many methods would be impossible to maintain and reuse. The approach taken in the *Rainbow* project[2] was to combine multiple independent tools through a web service interface. As the number of available tools increases, their composition by traditional programming however becomes cumbersome. Since we want to avoid the complexity of planning from scratch, the *template-based approach* looks as a reasonable compromise. Svátek et al. [12] designed a collection of *PSMs* abstracted from real web mining applications, with individual services positioned in a *multi-dimensional space*. This space could play a similar role as the space of template parameters from [14].

The goal of the paper is to demonstrate the applicability of generic templates for building web analysis applications. A collection of templates, simplified services and ontological knowledge were formulated and processed with a service composition tool and a service execution tool, all implemented in Prolog. The templates describe different variants of classification, and the services either perform generic web-space operations (such as finding a 'hub' page) or specialise in analysis of web pornography; the later are derived from an existing application [15].

2 Configuration of Web Services as Parametric Design

Web service configuration can possibly be regarded as *parametric design*, in which the parameters (slots) of a fixed template have to be instantiated with appropriate component services. During the configuration process we exploit knowledge about the template and the components so as to obtain the required composite web service. An existing reasoning method (PSM) for parametric design is *Propose-Critique-Modify* (PCM) [6] consisting of four steps:

- The *propose step* generates an initial configuration.
- 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.
- The *modify* step determines alternative values for the parameters identified by the critique step. The method then loops backto the verify step.

Our specific focus were *classification* services, for which a general template had been presented by Motta [9]:

1. First the observations have to be verified whether they are legal (*Check*).

[2] `http://rainbow.vse.cz`

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. A prototype PCM broker exploiting this template has been successfully applied in the domain of conference paper classification (reviewer assignment) [14].

3 Framework and PSMs for Deductive Web Mining

In [12], *deductive web mining* (DWM) 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, typically one of:
   - *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.
2. Type of *objects* on input and output. 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).
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.

The 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. The 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.

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 [12]. Here we only concentrate on the *Classification* task, which can be carried out either within a single inference (with several variations), or in a structural way. *Structural Classification* PSM 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

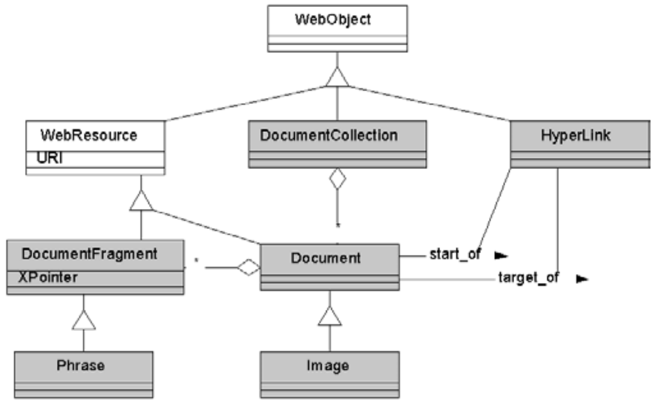

Fig. 1. UML diagram of Upper Web Ontology

of *global classification patterns* for the current object. Compared to the generic Classification template from section 2, this notion of classification is slightly modified. Some parts of Structural Classification PSM can be mapped on the generic template: classification from lower level of recursion is similar to Micro-Match, 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).

The *propose-critique-modify* method of *parametric design* (section 2), implemented by means of a configuration broker, was proven 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, the recursive nature of PSMs for DWM 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 algoritm* 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).

The first approach is demonstrated in Section 4. The second approach would be more demanding but would offer more flexibility.

Table 1 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

Table 1. Sample templates for classification task

```
templ(sc1,s(cla,0,0,Tp1,Tp2),
  [s(cla,0,0,Tp3,Tp4)],[subclasseq(Tp3,Tp1),subclasseq(Tp4,Tp2)]).
templ(sc2,s(cla,0,0,Tp1,Tp2),
  [s(cla,0,0,Tp3,Tp4),s(cla,0,0,Tp5,Tp4),
  s(agr,[ref(1,0),ref(2,0)],0,Tp4,Tp4)],
  [subclasseq(Tp3,Tp1),subclasseq(Tp5,Tp1),subclasseq(Tp4,Tp2)]).
templ(sc3,s(cla,0,0,Tp1,Tp2),
  [s(ret,0,1,Tp3,Tp4),s(cla,1,1,Tp5,Tp6),s(tsf,ref(2,1),0,Tp6,Tp2)],
  [subclasseq(Tp3,Tp1), rel(part,Tp4,Tp3), subclasseq(Tp4,Tp5)]).
templ(sc4,s(cla,0,0,Tp1,Tp2),
  [s(ret,0,1,Tp3,Tp4),s(cla,1,1,Tp5,Tp6),s(tsf,ref(2,1),0,Tp6,Tp2)],
  [subclasseq(Tp3,Tp1),rel(adj,Tp4,Tp3),subclasseq(Tp4,Tp5)]).
templ(sc5,s(cla,0,0,Tp1,Tp2),
  [s(cla,0,0,Tp3,Tp4),s(ret,0,1,Tp5,Tp6),s(cla,1,1,Tp7,Tp8),
  s(tsf,ref(3,1),0,Tp8,Tp4),s(agr,[ref(1,0),ref(4,0)],0,Tp4,Tp4)],
  [subclasseq(Tp3,Tp1),subclasseq(Tp5,Tp1),rel(part,Tp6,Tp5),
  subclasseq(Tp6,Tp7),subclasseq(Tp4,Tp2)]).
```

rely on another object (sub-object or related 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 types (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 `Tp#` correspond to types (or 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[3]. 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).

[3] This method of combining control flow and data flow is rather cumbersome and is likely to be replaced with a smarter one in a real web service composition language.

Table 2. Service composition dialogue

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

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

4 Simulation of Template Configuration and Execution

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
3. simplified services (incl. auxilliary ones) operating on 'website clauses' and 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 the UWO and its sub-models) containing definitions of basic concepts required for the composition and/or execution phase.

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. The classes amount to pornography-relevant ones, such as pornography-containing site or pornography content page. Table 2 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.

The composed services can then be *executed*. For example, we can call the already configured template `sc4` from above using the ID of input object, its initial class (e.g. just `document` as generic type) 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 (here, `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.

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 have been formulated in [13]. Compared to broker knowledge from [14], it also includes template-selection/reformulation knowledge in addition to slot-filling knowledge

5 Related Work

In the *IBrow* project [1], operational PSM libraries have been for 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 [7] 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.

6 Conclusions and Future Work

Configuration of web services can be considered as *parametric design*, which allows 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 nature of web as underlying data structure, service templates tend to involve *recursion*, which impacts the process of template-filling. The simulation described in this paper dealt with classification of web pornography.

The current prototype is only meant for the sake of initial experiment. 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 [11]. Future research also includes specification of templates for other DWM tasks, in particular those with nature of *extraction*, taking models of applications from [12] as starting point. Finally, we consider to align our approach with the WSMO project[4], which, as successor of IBrow project, could be open to the use of PSMs for web service composition.

Acknowledgements

The research is partially supported by the grant no.201/03/1318 of the Czech Science Foundation "Intelligent analysis of the WWW content and structure" and by the Knowledge Web Network of Excellence, IST FP6-507482. The authors would like to express their thanks to Annette ten Teije for her repeated explanations on PSM-based web service configuration principles.

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. Mandell, D. J., McIlraith, S. A.: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. In: Proc. ISWC2003.
9. Motta, E., Lu, W.: A Library of Components for Classification Problem Solving. In: Proceedings of PKAW 2000, Sydney, Australia, December 2000.
10. Schreiber, G., et al.: Knowledge Engineering and Management. The CommonKADS Methodology. MIT Press, 1999.
11. 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.
12. Svátek, V., Labský, M., Vacura, M.: Knowledge Modelling for Deductive Web Mining. In: Proc. EKAW 2004, Springer Verlag, LNCS, 2004.
13. Svátek, V., ten Teije, A., Vacura, M.: Web Service Composition for Deductive Web Mining: A Knowledge Modelling Approach. In: Znalosti 2005, High Tatras, 2005.
14. ten Teije, A., van Harmelen, F., Wielinga, B.: Configuration of Web Services as Parametric Design. In: Proc. EKAW 2004, Springer Verlag, LNCS, 2004.
15. Vacura, M.: Recognition of pornographic WWW documents on the Internet (in Czech), PhD Thesis, University of Economics, Prague, 2003.

[4] `http://www.wsmo.org`