Configuration of Web Services as Parametric Design

Annette ten Teije [1] and **Frank van Harmelen**[2] and **Bob Wielinga**[3]

Abstract. The configuration of composite Web services is particularly hard given the heterogeneous, unreliable and open nature of the Web. Furthermore, such composite Web services are likely to be complex services, that will require adaptation for each specific use.

We propose a knowledge-intensive brokering approach to the creation of composite Web services. In our approach, we describe 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. Our approach exploits the knowledge engineering literature, and in particular the problem solving methods work of the last decade.

We illustrate our proposal by applying it to a specific family of Web services, namely "classification services". We have implemented a prototype of our knowledge-intensive broker and describe its execution in a concrete scenario.

1 INTRODUCTION

Web services have raised much interest in various areas of Computer Science. In AI, the notion of *Semantic Web Services* has attracted much attention. According to [3]: "Semantic Web services build on Web service infrastructure to enable automatic discovery and invocation of existing services as well as *creation of new composite services* [...]". In particular the configuration of Web services has gained attention from AI researchers [7, 5]. This problem is particularly hard given the heterogeneous, unreliable and open nature of the Web. Furthermore, such composite Web services will be complex services, that will require adaptation for each specific use.

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 [6]. This problem of creation of new composite web services is in principle equal to the old problem of generalised automatic programming. This problem is notoriously unsolved in general by any known techniques. There is no reason to belief that the Web service version of this problem will be any less resistant to a general solution.

In this paper, we propose instead a *knowledge intensive* approach to the creation of composite Web services. We describe 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 free": it is supposed to work on any set of components, given simply their descriptions. A *broker* on the other hand exploits specific knowledge about the objects he is dealing with.

In the remainder of this paper, we describe how such a broker can be equipped with configuration knowledge on how to combine these web services.

We illustrate our proposal by applying it to a specific family of Web services, namely "classification services", and we describe a specific implementation and execution of our approach.

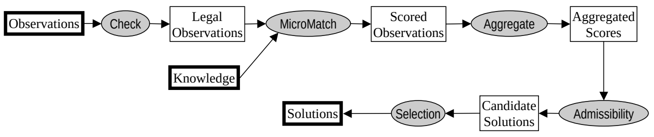

Figure 1. Structure of classification services. The boxes with thick lines are input and output. The ovals are the parameters of the template for the family of classification services.

2 PARAMETRIC DESIGN

Parametric Design is a simplification of general configuration. Parametric Design assumes that the objects-to-be-configured all have the same overall structure in the form of preconfigured templates. Variations on the configuration can only be obtained by choosing the values of given parameters within these templates.

Parametric Design requires that the object-to-be-designed (in our case: a Web service) is described in terms of a fixed structure containing parameters with adjustable values.

Question 1: can large classes of Web services be described in this way? This question will be tackled in section 3.

An existing reasoning method for parametric design is *Propose-Critique-Modify*, or PCM for short [2]. The PCM method consists of four steps:

The propose step generates an initial partial or complete 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. If the verification step fails, the critique step analyses the reasons for this failure: it indicates which parameters may have to be revised in order to repair these failures.

[1] Dept. of AI, Vrije Universiteit Amsterdam, annette@cs.vu.nl
[2] Dept. of AI, Vrije Universiteit Amsterdam
[3] Dept. of Social Science Informatics, SWI, University of Amsterdam

The modify step determines alternative values for the parameters identified by the critique step. After executing the modification step, the PCM method continues again with a verify step. This loop is repeated until all required properties of the service are satisfied.

The propose-critique-modify method for Parametric Design requires specific types of configuration knowledge to drive the different steps of the configuration process

Question 2: can this PCM-knowledge be identified for large classes of Web services? This question will be tackled in section 3.

3 EXAMPLE: CLASSIFICATION SERVICES

We illustrate our proposal by applying it to a *classification services*. The common definition of classification is [8]: "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 ⟨*feature,value*⟩-pairs) to classes.

We address question 1 above: can classification services be described in a single template? [4] does indeed present such a general template (see fig. 1):

First the observations have to be verified whether they are legal (Check). Each of these legal observations (⟨*feature,value*⟩-pairs) have to be scored on how they contribute to every possible solution in the solution space (MicroMatch). These individual scores are then aggregated (Aggregate). These aggregated scores are the basis for determining the candidate solutions (Admissibility). A final step (Selection) then selects among these candidate solutions the best final solutions.

This structure constitutes the overall template for classification services. Each box from fig. 1 is one parameter to configure in this fixed template. We have shown that such a template structure can also be easily captured in current Web service description languages, such as OWL-S [1].

We give some example values of one of the parameters (more examples can be found in [9]):

Example values of the Admissibility parameter:
(These are all taken from [8]).

• weak-coverage: Each ⟨*feature,value*⟩ pair in the observations has to be consistent with the feature specifications of the solution.

• strong-coverage: These are weak-coverage solutions with no unexplained features.

• explanative: These are weak-coverage solutions for which no feature specifications are missing.

Such parameter instances can also be described in current Web service description languages (e.g. OWL-S).

The question still remains if it is possible to identify the knowledge required for the propose-critique-modify method and each of its four steps (i.e. question 2) We show that this is indeed the case, by giving parts of the PCM knowledge required to configure classification services. (Again, more examples in [9]).

Example Propose knowledge for the Admissibility parameter:

• if many ⟨*feature,value*⟩ pairs are irrelevant, then do not use strong-coverage (because strong-coverage insists on an explanation for *all* observed features, including the irrelevant ones).

Example Critique knowledge for the Selection parameter:

• When the solution set is too small (e.g. empty) or too large (e.g. > 1), then adjust the Admissibility or the Selection parameter.

Example 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.

4 AN EXAMPLE SCENARIO

To test our brokering approach to Web service composition, we have configured the classification services needed to support Programme Chairs of major scientific conferences.

In our experiment, we have emulated the paper-classification process for the ECAI 2002 conference. There were 605 submissions to ECAI 2002, each characterised by a set of author-supplied keywords, i.e. each keyword is a ⟨*feature,value*⟩-pair with value either 0 (keyword absent) or 1 (present). In total, 1990 keywords were given by authors. These had to be mapped onto 88 classes: 15 broad classes which were further subdivided into 73 more specific classes. Of the 650 papers, 189 were classified by hand by the Programme Chair. These classifications can be considered as a golden standard.

Requirement 1: The classification service must classify each paper in at least one of the 15 major categories

Requirement 2: The service must reproduce the Chair's solution on the 189 handclassified papers.

Important characteristics of this domain are that:

Characteristic 1: the feature-values are often noisy (authors choose remarkably bad keywords to characterise their paper), and

Characteristic 2: it is hard to determine in advance what the required classification mechanism should be. Requiring all keywords of a paper to belong to a solution class might be too strict, resulting in many unclassified papers, and violating requirement 1. But requiring only a single keyword to appear might well be too liberal, causing violation of requirement 2. These characteristics ensure that this domain requires dynamic configuration of the classification process.

The iterative service-configuration process performed by our PCM broker is summarised in the table below:

Iteration	Answers	Golden Standard	Modification
1	**0 (0%)**	0 (0%)	Admissibility
2	**93 (15%)**	16 (8%)	Admissibility
3	595 (98%)	**81 (45%)**	Selection
4	595 (98%)	**103 (54%)**	Selection
5	595 (98%)	**145 (76%)**	Selection
6	595 (98%)	**169 (89%)**	

After a repeated series of six of cycles, our broker arrives at a configuration that sufficiently satisfies both requirements.

Acknowledgements: This research was partially supported by the European Commission IBROW project (IST-1999-19005). We like to thank Enrico Motta, Anjo Anjewierden, Guus Schreiber, Machiel Jansen, Marta Sabou.

REFERENCES

[1] A. Ankolekar, et al.: 'DAML-S: Semantic markup for web services', in *ISWC 2002, LNCS* 2342, pp. 348–363.

[2] D. Brown and B. Chandrasekaran, 'Design problem solving: knowledge structures and control strategies', *Research notes in AI*, (1989).

[3] M. Kiefer. Message to swsl-committee@daml.org, May 14, 2003.

[4] E. Motta and W. Lu, 'A library of components for classification problem solving', in *Pacific Rim Knowledge Acquisition Workshop*, (2000).

[5] S. Narayanan and S. Mcllraith, 'Simulation, verification and automated composition of web services.', in *WWW 2002*.

[6] M. Sheshagiri, M. desJardins, and T. Finin, 'A planner for composing service described in DAML-S', in *Workshop on Planning for Web Services, at ICAPS 2003*.

[7] E. Sirin, J. Hendler, and B. Parsia, 'Semi-automatic composition of web services using semantic descriptions', in *Web Services: Modeling, Architecture and Infrastructure workshop at ICEIS2003*.

[8] M. Stefik, *Introduction to knowledge systems*, Morgan Kaufmann, 1995.

[9] A. ten Teije and F. van Harmelen. IBROW deliverable wp4.1 & 4.2: Task & method adaptation, jan 2003.