Configuring Configuration Requests: An Application to the Semantic Web

Mathias Kleiner and **Laurent Henocque**
ILOG S.A, 9 rue de Verdun, 94253 Gentilly, France
Laboratoire LSIS, Université de Saint-Jérôme, Avenue Escadrille Normandie-Niemen, 13397 Marseille, France

Abstract

Beyond its usual industrial fields of application, a current body of research explores the use of constraint based configuration to address general AI problems, like for instance automatic composition of semantically enriched web services (SWS). A configuration request is naturally formulated as a fragment of the desired solution, that the configurator will attempt to complete according to constraints. We address here a case where the design of the configuration request may itself be the result of a configuration phase, that helps the user design the request by formulating it on more abstract grounds. Within this framework, the configurator is first used to complete an abstract request formulated in a specific formalism. Then a translation is performed from the goal model to the final model to yield the actual request sent to the second configuration phase. This research builds on previous experience showing the adequacy of using configuration to compose SWS, that raised further issues regarding the nature of queries.

Introduction

The present work builds upon a constraint based approach presented in (Albert *et al.* 2005a; 2005b) where it was proved feasible to compose semantic web services orchestrations using configuration.The original question is "can automatic composition be achieved?", based upon a description of the participant SWS interfaces. Even a non "SWS enabled" reader can convince himself that this is not possible. For instance, if two semantic web services produce a "price" as their output, no automatic system based solely on interface definitions can infer that these prices must be added, or that their mean must be taken. If such a point is relevant to composition, this information must be placed in some way in the composition request, so that the configurator (or any other composer) can only produce compatible solutions. In the target field of application, this makes the process of designing such requests too difficult in general, and in fact makes it an AI problem of its own. The main argument is that since a configuration request is a fragment of the solution, it must comply with the constraints, and can be very difficult to state. The main contribution of this paper is to describe a situation and framework where composition requests need themselves to be configured. Former experiments raised the requirement to automatically compose those requests from natural and adaptable user input at an abstract goal level. Among the "side" benefits that can be expected from such a framework is to reduce the global search space. The approach builds upon a framework experimentally validated in the DIP european project (DIP): first ideas specified in (Henocque and Kleiner a), and prototype presented in (Henocque and Kleiner b). The current Section briefly introduces the context (semantic web services, composition, configuration) and some related work. We then present the abstract syntax for the goal language, a concrete syntax and how it is translated to the workflow model. Finally we provide experiments through a real-life scenario as well as an analysis of this two-step procedure benefits, and we finally conclude and propose future work.

Copyright © 2007, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Brief introduction to Semantic Web Services

Semantic Web Services definition Semantic Web Services (SWS) are commonly defined using both *functional* and *behavioural* characteristics. There are currently two main formalisms for describing SWS: OWL-S and WSMO (Fensel). This paper concentrates on WSMO descriptions. They separate the behavioural description (called *interface*) in two different sections: the *choreography* describes the proper way for a client to consume its functionality, whereas the *orchestration* describes how the behaviour of a composed service is achieved using several clients (typically other SWSs). The functionality is described using a *capability*. We therefore consider SWSs being defined through:

- a capability: describes what the SWS can realize and what it requires in terms of *input* and *output messages*. Ontologies are the building blocks in this knowledge representation. Message types are defined using concepts taken from a specific ontology.
- a choreography: describes how to interact with the service in terms of message exchange patterns.
- an orchestration: in the case of a composed web service, describes how it uses and communicates with external SWSs to achieve its capability. Orchestration is the main output of a composition tool or activity.

Abstractions of SWS : atomic goals A user (or a composition tool) looking for a particular SWS will express

requirements allowing its *discovery*. Such requirements are called (atomic) *goals*. Goals generalize SWS, as they may match several registered SWSs and can hence be seen as SWS abstractions. Goals are therefore defined very much like SWS capabilities, specifying available inputs, expected outputs and constraints. In what follows, we call *input* and *output roles* the abstraction of input and output messages mentioned at the goal level.

Brief introduction to Composition

Composition is defined as the "act of combining and coordinating a set of Semantic Web Services". Under such a definition, "composition" naturally refers to the process involved in computing an orchestration.

Brief introduction to Configuration

A configuration task consists in building (a simulation of) a *complex product* from *components* picked from a catalog of *types*. Neither the number nor the actual types of the required components are known beforehand. Components are subject to *relations*, and their types are subject to *inheritance*. *Constraints* (also called well-formedness rules) generically define all the valid products. A configurator expects as input a fragment of a target object structure, and expands it to a solution of the configuration problem, if any, addind all necessary elements during search. This first-order logic problem is semi-decidable in the general case. A configuration program is well described using a *constrained object model* in the form of a standard class diagram, together with well-formedness rules or constraints. Technically solving the associated enumeration problem can be made using various formalisms or technical approaches: extensions of the CSP paradigm (Mittal and Falkenhainer 1990; Fleischanderl *et al.* 1998), knowledge based approaches (Stumptner 1997), terminological logics (Nebel 1990), logic programming (using forward or backward chaining, and non standard semantics)(Soininen *et al.* 2000), object-oriented approaches (Mailharro 1998; Stumptner 1997).

Related Work

Automated workflow or SWS composition is a field of intense activity, with applications to at least two wide areas: Business Process Modeling and (Semantic) Web Services. Approaches to these problems are experimented using many formalisms and techniques, among others Situation calculus (Sohrabi *et al.* 2006), Logic programming (Sirin *et al.* 2003), Type matching (Constantinescu *et al.* 2005), colored Petri nets (Dijkman and Dumas 2004), Linear logic: (Rao *et al.* 2004), Problem solving methods (Gómez-Pérez *et al.* 2004), AI Planning (Pistore *et al.* 2004), Markov decision processes (Doshi *et al.* 2005).
This work takes place within an emerging trend of research for semantic web composition, involving request languages (EaGLe (Pistore *et al.* 2004), its evolution in (Pistore *et al.* 2005) (where the term "composition goals" also occurs)). The composition language in (Kumar *et al.* 2005) introduces a concept of roles similar to our own definition. Some approaches involve constraint based composition (Hassine *et al.* 2006). In (Sohrabi *et al.* 2006) situation calculus is used with preferences. As far as we know, no existing language offers the expressiveness covered by our composition goals while staying at the abstract goal level.

Abstract Syntax

It is not possible to fully present here the formal specification of the goal language. We present its abstract syntax in Figures 1, 2 and 3, in the form of a constrained object model. We are using a subset of the Z language (Spivey 2001) to document the diagram restrictions that we have chosen to enforce. We also give unformal semantics to the constructs indicating their semantics for the workflow composer.

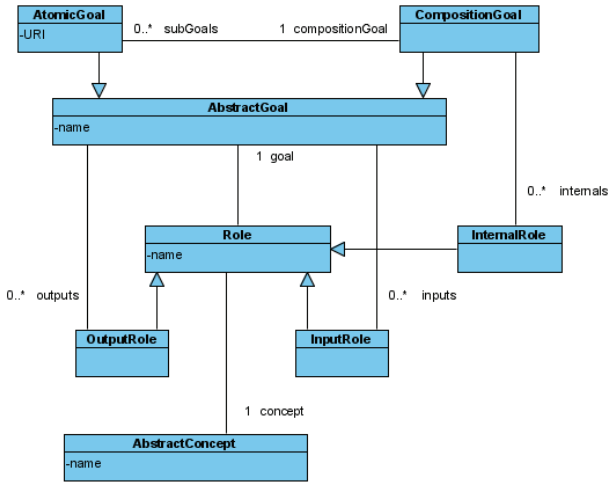

Figure 1: Goals, roles and concepts model

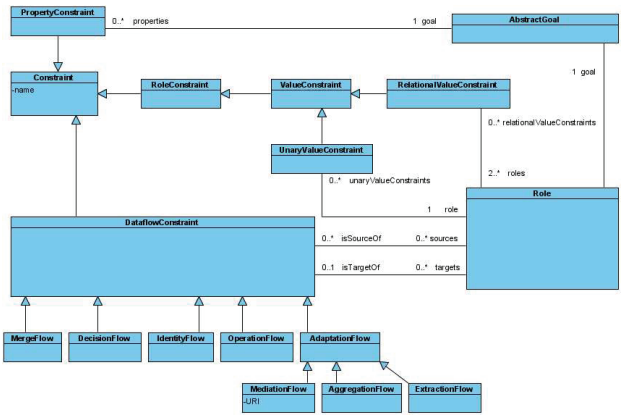

Figure 2: Constraints on Goals and Roles model

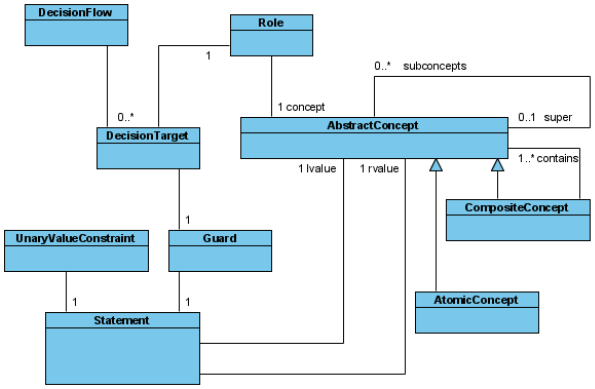

Figure 3: Concepts and statements model

Classes The Z formalization of the classes strictly matches the UML diagrams.

- Atomic goals: abstractions of SWS's.
- Roles: Inputs and outputs of matching SWS's. Internal roles can be used to denote intermediate objects in the orchestration. Each role has a type defined by a concept taken from an ontology. A concept can either be atomic or composite.
- Value Constraints: the solution workflow respects any given value constraints.
  - Unary Value Constraints: the solution workflow ensures that the specified object will respect given (ranged) values.
  - Relational Value Constraints: the solution workflow ensures that the specified objects will respect given statement between them.
- Dataflow Constraints: the solution workflow ensures the existence of a specific dataflow between sources and targets:
  - IdentityFlow: roles are semantically equivalent.
  - OperationFlow: the dataflow will perform an operation on sources tokens values to obtain targets.
  - MediationFlow: the dataflow between sources and targets needs to use a specific mediator.
  - AggregationFlow: the dataflow will aggregate sources into the composite concept target.
  - ExtractionFlow: the dataflow will extract parts of the composite concept source into the targets.
  - DecisionFlow: the dataflow will go from source to different targets depending on the guards statements.
  - MergeFlow: the dataflow takes any incoming source and delivers it to target.

Goal Language Constraints We cannot present here all the constraints that apply to the model. A few of them will give the flavor of it:

- Data flow's sources must be of class OutputRole or InternalRole:

$$\forall n : Role \mid \#(n.isSourceOf) > 1 \bullet \\ n \in OutputRole \lor n \in InternalRole$$

- Data flow's targets must be of class InputRole or InternalRole:

$$\forall n : Role \mid \#(n.isTargetOf) > 1 \bullet \\ n \in InputRole \lor n \in InternalRole$$

- ExtractionFlow has a single source which concept is composite, and targets are concepts included in the source:

$$\forall n : ExtractionFlow \bullet \\ \#(n.sources) = 1 \\ \land\ sources(n) \rightarrow concept \in CompositeConcept \\ \land\ targets(n) \rightarrow concept \\ \subset sources(n) \rightarrow concept \rightarrow contains$$

- Input and Internal roles are targets of dataflow constraints:

$$\forall r : InputRole \bullet \#(r.isTargetOf) > 0 \\ \forall r : InternalRole \bullet \#(r.isTargetOf) > 0$$

- Validity of Statements:

$$\forall a : Statement \bullet a.rvalue \in a.lvalue.subconcepts^{+}$$

Concrete Syntax

We briefly provide in Figure 4 a concrete syntax for the abstract goal language. We will use this notation for the examples in the following sections, for instance in Figure 8.

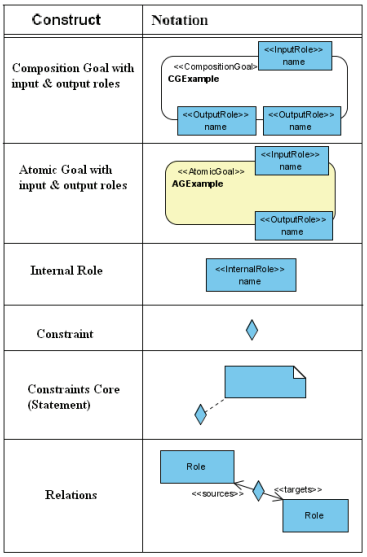

Figure 4: Graphical representation for composition goals

Translating Abstract Goals to Composition Requests

The language used for defining composition requests is not identical to the workflow metamodel used for the composition phase. Before the results for request generation can be used, they hence must undergo a specific translation step. In order to achieve this, we have extended the workflow model with a limited number (four) extra abstract classes, presented in Figure 5 with a fragment of the workflow model itself. Any abstract goal instance (notably the product of

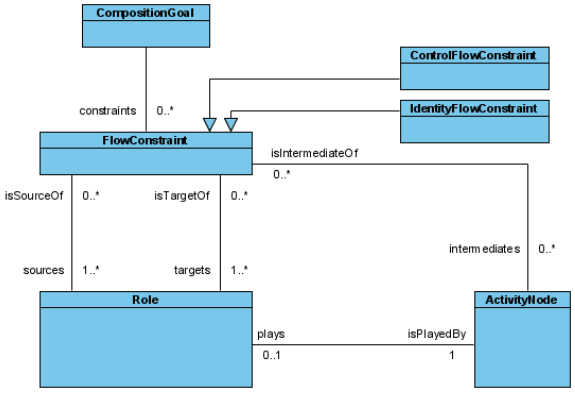

Figure 5: Extension of the UML2AD subset object model

goal configuration) can be translated using a straightforward linear algorithm to a proper composition request. Figure 6 shows the translation of a MergeFlow into the activity diagram model. The top diamond represents a MergeFlowConstraint in the request language, the bottom diamond is a true activity diagram merge node from the workflow model.

Experiments and analysis

We have conducted experiments on a real-life scenario, joint developped with a DIP use-case partner (British Telecom). The scenario has been integrated in a full SWS framework, from composition (out of a full composition goal request) to execution in the 3-layer descriptions defined in (Norton *et al.*). We used ILOG's JConfigurator for implementation. In this problem, the user wants to design a composed web service which offers to buy a bundle of a network connection, a modem and a PC. Figure 7 shows a possible request: the user specifies the message he expects as result of the orchestration as well as the information he can provide. Figure 8 shows a possible composition goal solution. The red diamonds represent ExtractionFlow and AggregationFlow constraints. The orange diamonds are DecisionFlow and MergeFlow constraints.

In this example no goal offers both AMD and Intel processors, which forces the composer to create alternative paths. Although the user gave little information, the composer is able to provide a very detailed composition request based on available elements.

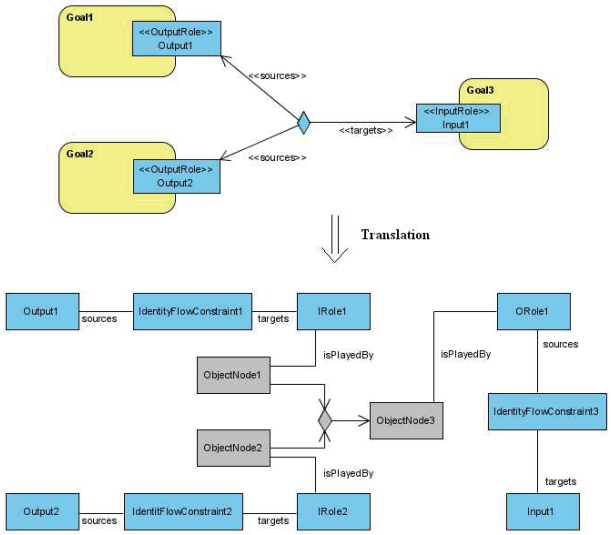

Figure 6: Example of the MergeFlow translation

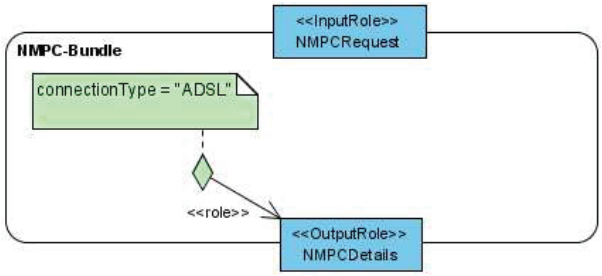

Figure 7: A possible input for the NMPC Bundle scenario

The propagation of UnaryValueConstraints such as the NetworkDetails' connectionType is key in reducing search space in the next step (workflow composition). Indeed, only SWS offering ADSL connections will need to be discovered, thus limiting the number of candidate choreographies. At design time, it is possible to check and/or refine the generated request by adding additional constraints (temporal, non-functional properties) before proceeding to the orchestration search.

We present in Figure 9 how this generated composition request is further exploited by a second configuration step to create the final orchestration.

Conclusion

This research illustrates on a working example how constraint based configuration use can be cascaded, to first generate in an abstract request language a request for the problem to solve. Since a configuration request is a fragment of the desired solution, and since such a fragment must comply with the model constraints, the need for user expertise

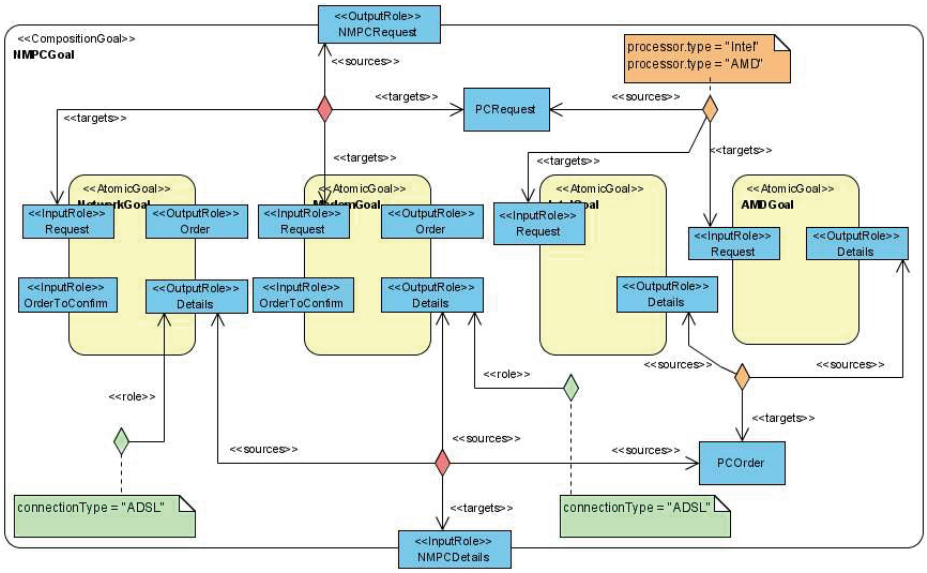

Figure 8: The computed composition goal for the NMPC Bundle scenario

when formulating configuration requests can be fairly well accepted in the general case. We presented these early results in the hope that they can foster thoughts about the nature of configuration requests and maybe give hints on the implementation of configuration tool interfaces.

References

Patrick Albert, Laurent Henocque, and Mathias Kleiner. Configuration based workflow composition. In *proceedings of International Conference on Web Services ICWS'05*, Orlando, Florida, USA, 2005.

Patrick Albert, Laurent Henocque, and Mathias Kleiner. A constrained object model for configuration based workflow composition. In *proceedings of the Third International Conference on Business Process Management*, pages 102–115, Nancy, France, september 2005.

Ion Constantinescu, Walter Binder, and Boi Faltings. Flexible and efficient matchmaking and ranking in service directories. In *IEEE International Conference on Web Services*, pages 5–12, 2005.

R. Dijkman and M. Dumas. Service-oriented design: A multi-viewpoint approach, ctit technical report series no. 04-09. Technical report, Centre for Telematics and Information Technology, University of Twente, The Netherlands, February 2004.

DIP. Data, information, and process integration with semantic web services http://dip.semanticweb.org. Technical report, DIP Project.

Prashant Doshi, Richard Goodwin, Rama Akkiraju, and Kunal Verma. Dynamic workflow composition using markov decision processes. *International Journal of Web Services Research*, 2(1):1–17, Jan-March 2005.

D. Fensel. The web services modeling ontology. http://www.wsmo.org. Technical report.

G. Fleischanderl, G. Friedrich, A. Haselböck, H. Schreiner, and M. Stumptner. Configuring large-scale systems with generative constraint satisfaction. *IEEE Intelligent Systems - Special issue on Configuration*, 13(7), 1998.

Asunción Gómez-Pérez, Rafael González-Cabero, and Manuel Lamaa. A framework for design and composition of semantic web services. In *Semantic Web Services, 2004 AAAI Spring Symposium Series*, 22nd-24th March 2004.

A. Ben Hassine, S. Matsubara, and T. Ishida. A constraint-based approach to horizontal web service composition. In *5th International Semantic Web Conference*, 2006.

Laurent Henocque and Mathias Kleiner. Dip deliverable 4.12: Goal-oriented sws composition module specification, 2005. http://dip.semanticweb.org/deliverables.html. Technical report, DIP Project.

Laurent Henocque and Mathias Kleiner. Dip deliverable 4.22: Goal-oriented sws composition prototype, 2006. http://dip.semanticweb.org/deliverables.html. Technical report, DIP Project.

A. Kumar, B. Srivastava, and S. Mittal. Information modeling for end to end composition of semantic web services. In *4th International Semantic Web Conference*, 2005.

D. Mailharro. A classification and constraint based framework for configuration. *AI-EDAM : Special issue on Configuration*, 12(4):383 – 397, 1998.

S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In *Proceedings of AAAI-90*, pages 25–32, 1990.

B. Nebel. Reasoning and revision in hybrid representation systems. *Lecture Notes in Artificial Intelligence*, 422, 1990.

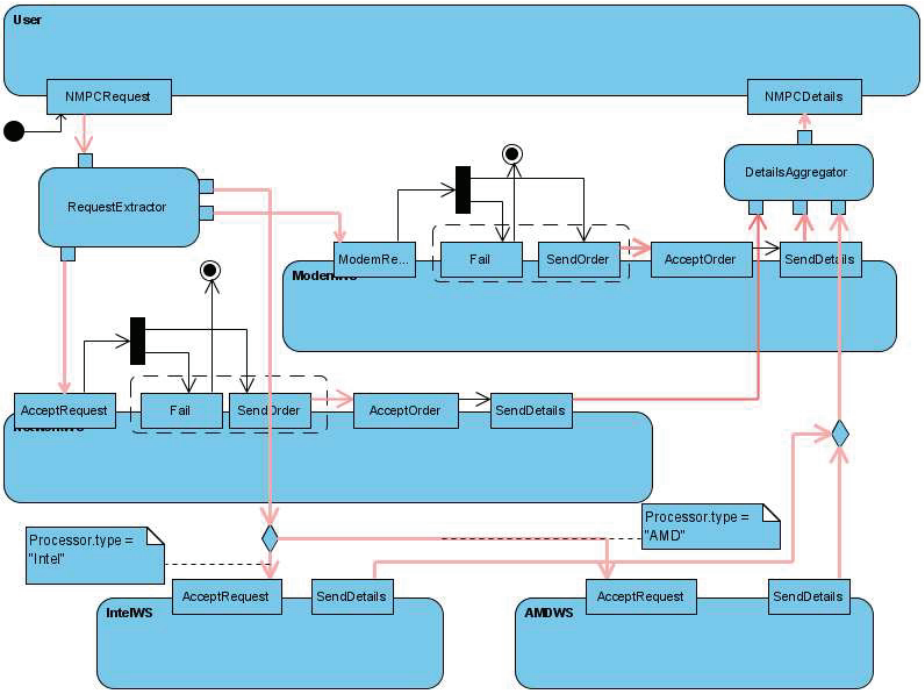

Figure 9: The computed orchestration for the presented request

B. Norton, J. Lemcke, C. Pedrinacci, Laurent Henocque, and Mathias Kleiner. Dip deliverable 3.8: An ontology for web services choreography and orchestration, 2006. http://dip.semanticweb.org/deliverables.html. Technical report, DIP Project.

Marco Pistore, F. Barbon, Piergiorgio Bertoli, D. Shaparau, and Paolo Traverso. Planning and monitoring web service composition. In *proceedings of the Workshop on Planning and Scheduling for Web and Grid Services held in conjunction with ICAPS 2004*, Whistler, British Columbia, Canada, June 3-7 2004.

M. Pistore, P. Roberti, and P. Traverso. Process-level composition of executable web services: on-the-fly versus once-for-all composition. In *European Semantic Web Conference*, 2005.

J. Rao, P. Kungas, and M. Matskin. Logic-based web service composition: from service description to process model. In *proceedings of the 2004 IEEE International Conference on Web Services, ICWS 2004*, San Diego, California, USA, July 6-9 2004.

E. Sirin, J. Hendler, and B. Parsia. Semi automatic composition of web services using semantic descriptions. In *proceedings of the ICEIS-2003 Workshop on Web Services: Modeling, Architecture and Infrastructure*, Angers, France, April 2003.

S. Sohrabi, N. Prokoshyna, and S. A. McIlraith. Web service composition via generic procedures and customizing user preferences. In *5th International Semantic Web Conference*, 2006.

T. Soininen, I. Niemelõ, J. Tiihonen, and R. Sulonen. Unified configuration knowledge representation using weight constraint rules. In *ECAI 2000 Configuration Workshop*, 2000.

J. M. Spivey. *The Z Notation: a reference manual*. Prentice Hall originally, now J.M. Spivey, 2001.

Markus Stumptner. An overview of knowledge-based configuration. *AI Communications*, 10(2):111–125, June 1997.