Requirements for Implementing Business Process Models through Composition of Semantic Web Services

Ingo Weber

SAP Research, Vincenz-Priessnitz-Strasse 1, 76131 Karlsruhe, Germany
ingo.weber@sap.com

Summary. Despite significant research efforts, automated service composition is neither taken up broadly nor can it be considered a solved problem. We believe that this is partly the case due to an unclear formulation of the actual business requirements and frame for this technology. In this paper, we formulate a list of 14 requirements for composition in the contexts of business process implementation, enterprise application integration, and interoperability, which can be considered quite challenging not only for current technology but also in terms computational complexity. However, if these requirements are met by an approach to composition – potentially by combining existing technology – the relevance of automated composition to enterprise application software could increase significantly, and also quicken its success in other usage scenarios. We give a list of requirements towards service composition along with a starting point for a composition approach and a short analysis of the computational complexity.[1]

1 Introduction

The basic building blocks of Service-Oriented Architectures (SOAs) are Web services (WSs): self-contained pieces of software that are remotely callable and described in a standardised way. Service composition, the task of automatically combining the functionality of multiple Web services to yield a composite service with higher utility, is a big challenge and opportunity for the SOA paradigm. In order to tackle this and other problems, methods for describing Web services' semantics in a formally and machine-accessibly were developed in the more recent past [21, 3] and are referred to as Semantic Web services (SWSs).

Composite services, workflows and executable business process models are considered to be quite similar [20, 4], thus, a composite service can be ex-

[1] This work has in part been funded through the European Union's 6th Framework Programme, within Information Society Technologies (IST) priority under the SUPER project (http://www.ip-super.org)

pressed as an orchestration in a language such as WS-BPEL [1]. An *orchestration* expresses control flow over calls to various Web services and guarantees a predefined execution order. In BPEL, the orchestration itself is exposed as a Web service.

In contrast to other approaches towards web service composition, we plan to use service composition for the implementation of business process models. Here we refer to business process models as graphical diagrams, which depict a business process (BP) in an enterprise on a higher modelling level, such as that of a business expert. Since functionality from various application systems in an organization can be provided through Web service interfaces and composed together in an end-to-end business process, this approach can serve as a corner stone for facilitating enterprise application integration (EAI).

The service composition problem in this setting becomes the following: given a high-level BP model and a set of services, some with and some without control flow semantics over their operations, the composition component should automatically find a way to combine the services in order to implement the process model. For each of the activities in a high-level process model, this means finding one or more services which directly or together, respectively, implement the desired behavior, and establish an execution order over them. Some of the requirements and challenges are: subprocess interfacesthat implement more than one function; consistency in the orchestration of the various high-level activities, e.g., services which are supposed to be executed in parallel must be compatible with each other; varying data formats between various services; and more. Composition in this setup is clearly not an easy task, and will be accompanied by many requirements in any real world-domain. A list of such requirements is given in this paper, which, to the best of our knowledge, addresses a number of aspects that were neither stated nor taken into account in their completeness in previous work. Besides, it is yet unclear if the requirements can all be satisfied jointly, preferably through an efficient algorithm, or whether the resulting computational complexity exceeds tractability.

The remainder of this work is thus structured as follows: in order to give some background, Section 2 discusses the problem of composition in business process modelling in more detail. Section 3 is the main contribution of this paper, as it lists the actual requirements. Subsequently, some first steps towards an approach meeting the requirements are given in Section 4.1, while Section 4.2 discusses in short the computational complexity in domain. Related work is presented in Section 5, and Section 6 concludes.

2 Business Process Modelling: Background and Current Challenges

Business Process Modelling (BPM) serves as an abstraction of the way enterprises do business, whereby threads of work are noted down as human-comprehensible business process models. While notations such as the Event-

driven Process Chains (EPCs) received much attention as notations in the past, the Business Process Modelling Notation (BPMN) is gaining momentum and support. Figure 1 shows an example of a BPMN process model.

When modelling a business process today, the modeller usually creates the process manually in a graphical tool. Three common approaches are: creating the process from scratch, starting from a reference model, or improving or creating a variant of an existing process model (cf. [10]). The outcome is a process model that reflects a business expert's view. Subsequently, this process model is implemented by IT experts – or, rather, its control and data flow are mapped to implemented artifacts in information systems. The relationship between the business-level model and its IT-level implementation is oftentimes weak. Consequently, the process implementation can deviate substantially from the model, and changes on one of the levels cannot be easily propagated to the respective other level.

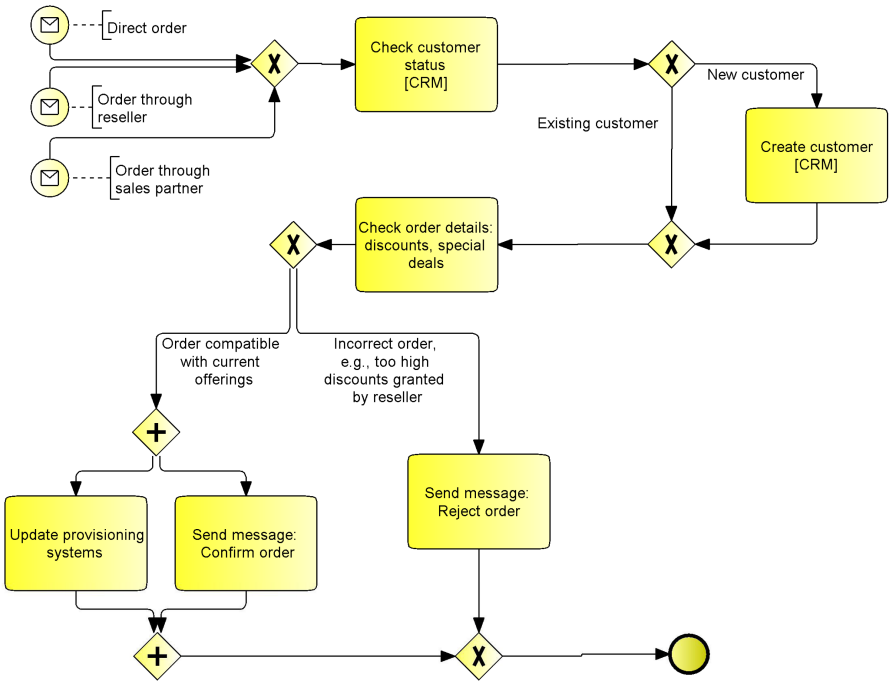

Fig. 1. Order-to-provisioning process from the Telecomunications domain.

The approach of automated composition which we pursue attempts to bridge that gap by finding program parts which can be used for the implementation of a process model and defining their usage in an executable process model. For this vision to become reality, several needs must be met: first, ap-

plication program fragments must be available for remote invocation and loose coupling, e.g., as Web services. Second, in order to allow the automation of tasks such as composition, these services are annotated with formal, machine-accessible semantics. If such an infrastructure is available and the requirements from the next section are met by a composition system, the automated implementation of business process models can in principle be performed. A very favorable side effect is the intrinsic relation between activities in the model and services in the implementation: it can simplify change management in both, graphical process model and executable process, significantly.

We look at composition from the viewpoint of an enterprise, not an end user. While business processes in business-to-consumer (B2C) scenarios can be a touching point with end users[2], service composition on the consumer's end is not the focus here. Rather, our work which can be regarded in the context of enterprise application integration and interoperability. In the scope of this work, composed processes must be correct and compliant with regulation such as Basel II or Sarbanes Oxley (SOX). Thus, the focus is on design-time composition in a known domain, which simplifies the problem in two ways: firstly, design-time composition can always be checked and approved manually before deploying and enacting it. Thus, unanticipated side effects[3] can be avoided through the manual control step, where decisions from the automated composition can be overruled. And secondly, in our known domain – the enterprise – we can assume to own the services and can thus enforce a uniform way and formalism of description.

If the approach is extended towards larger domains (e.g., cross-organisational, to service marketplaces or ecosystems, or even the internet), where presented information is more likely to be incomplete, knowledge gathering techniques as in [12, 13] can be re-visited. Also, adaptors for differing service descriptions need to be constructed, e.g. in form of mediators. Runtime service composition, on the other hand, is suggested as one of the base requirements of several existing approaches towards composition. While being an interesting problem with desirable features, the quality and correctness of current runtime composition forbids its application in our context of enterprise-level composition today and in the near-term future.

Other BPM-related interpretations of composition than ours are given in the following. (1) If a given process interface has to be implemented by orchestrating web services and (sub-)processes. (2) A collaborative, cross-organisational business process, also called choreOgraphy, requires the provision of a executable process from each one of involved parties. There is an

[2] Note, that e.g., the often-cited travel booking example (cf. among others [15, 2]), where a customer wants to book hotel and flight at the same dates, is a B2C scenario, where the service composition could be triggered by the customer. However, it is far more likely that the composite process is provided by a travel agency, which provides correct and tested composite services to the customers.

[3] Cf. [11] for such a side effect: a composite process satisfies the pre-condition of *having* a credit card by *applying* for one.

overlap between these situations and the problem that is addressed here, and potentially parts of the solution here can be used for a solution to those issues. However, the focus of this work remains on process model implementation.

3 Requirements for Service Composition Support to Business Process Modelling

Service composition in the context described above needs to meet a row of requirements, which we state in this section after describing the setup more concretely.

3.1 Problem Setup

Given a business process model and a set of Semantic Web services, each of the activities in the process must be implemented with one or more services. It is assumed that each activity is described in terms of preconditions and the anticipated goal (in a formal representation). The composition approach should then be able to come up with a BPEL-like orchestration fragment of calls to the provided subprocesses for each activity. The overall composite process, which is the join over the composite process fragments implementing the activities, can also be constrained in terms of the interface it should provide and must be consistent and constraint-preserving with respect to the fragments' content, i.e., the used services.

3.2 Requirements

A list of the most important requirements is provided below, which is divided into fundamental, common, and situational requirements. Fundamental requirements serve for defining the functionality of the basic composition technology in terms of service combination. Common requirements are expected to arise in most of the cases of the problem described here. Situational requirements point at specific situations in an enterprise application world that are more business-driven.

Fundamental Requirements

1. A task[4] can be implemented by one service operation[5]. The service must be found, included, and data formats must be adjusted accordingly (data mediation).

[4] Here we refer to tasks as they are used in the BPMN notation. In other business process modelling notations, the respective matching concepts are sometimes called functions or activities.

[5] This case comes down to service discovery, which can thus be seen as a special kind of composition.

2. A task can be implemented by a service with multiple operations. Here, the constraints may be overlapping with those of other called services (among others: process mediation between different subprocesses in the process model).
3. A task can be implemented by orchestrating multiple services. This implies that the control flow over the service calls must be inferred from their description. The need may arise from dependencies leading to the execution of services not directly related to the goal of a certain task.
4. Multiple tasks may be implemented by a service. This case may arise, when the granularity of the tasks is smaller than that of the service. Since BPEL processes can be exposed as Web services again, the case is considered to be relevant.
5. A task has a resource[6] attached to it, which must be taken into account in the composed process.
6. A task has multiple resources attached to it.
7. A set of tasks shares a resource assignment.

Note, that the combinations of these requirements are likely to occurr. E.g., the combination of multiple services (Requirement 3) with multiple operations each (Requirement 2) span and together implement multiple tasks from the high-level process (Requirement 4).

Common Requirements

8. Take into account transactional requirements as also done by [17], e.g., book hotel and flight, but if one of them fails, don't book anything. Scopes over which transactional behavior is specified could be expressed in the high-level business process.
9. Preparation of the runtime choice of one out of a class of services: e.g., based on non-functional properties like price, estimated execution time, location a runtime selection on a concrete service out of a class of functionally matching services can be made.
10. Protection scopes within subprocesses, i.e. parts of processes where certain constraints must hold (e.g. a customer complains about his invoice, and the respective department did not yet answer him – then the customer should not get a dunning letter before his complaint was processed). In terms of the workflow patterns [25], this requirement leads to the need for the pattern "interleaved parallel routing". The used language must be expressive enough to allow for modelling this kind of protection, and the composition must be able incorporate it.

[6] In BPM, such resource assignments are not uncommon. E.g., an approval task may be associated with a manager-role, meaning that during the execution of the process a specific manager needs to be assigned to the role. The term *resource* is more general than a role, and also used for machinery, physical goods, and the like.

11. Take into account business rules, in that the space of all allowed composite services may be constrained through such rules. E.g., if insurance claim approval must follow the four-eye principle, the composition approach must take this into account and only come up with composite processes that conform with the business rules.
12. Correctness and validity[7] of the dataflow: the composed process has a dataflow, be it implicit or explicit, which has to be correct by some means. E.g., data should be available by the point in the process where it is needed. This aspect can get tricky, e.g., when data is only provided by an optional branch in a process, and guarantees should be made regarding the availability of this data after the optional branch.

Situational Requirements

13. Goal-based configuration of subprocesses. If a subprocess is configurable and it is necessary or beneficial to the composite process to exploit this fact, then the composition approach should make use of it. E.g. if there are two mutually exclusive branches in a subprocess, the branching condition is configurable, and the only way to achieve a goal is to take one of the branches, then the composite process should call a respectively configured variant of the subprocess such that it always takes the desired branch.
14. Detection of missing services, as suggested in [9] and potentially identifying services that are related to a missing service. This issue requires some heuristics for finding the gaps, i.e., one must know that he is on the right street before he can say that a bridge is missing. He must thus know that the street goes in the desired direction, both, in front of and behind the obstacle, and that all that is missing is this bridge.

As services may be heterogeneous in terms of data structures in their messages or their behavioral interfaces, the composition approach should be able to overcome the heterogeneities through incorporating mediators at planning time: a mediator can, e.g., overcome differences in message formats, given the content is compatible on the semantic level. Thus, the question if two services can interact is no longer the question if their message formats are compatible out-of-the-box, it is rather the question if a mediator can be found that can bridge the gap.

The requirements given here arise from the differences between a business process modelling context and various Web service composition approaches: we see them as the first logical step in the attempt to bridge the perceived gap. In essence, they attempt to answer the question of why service composition is not widely used in today's EAI and BPM scenarios, and what is necessary to make a change to this situation.

[7] A thorough definition of correctness or validity regarding dataflow in process models is outside the scope of this work.

4 Analysis of the Feasibility of Technical Solutions

In the previous section, a number of requirements for BP implementation through service composition were listed. This section is concerned with first steps towards a solution and the issue that service composition is often challenging in terms of the computational complexity of used algorithms. At the end of this section, we shortly elaborate on existing results regarding the latter aspect.

4.1 Starting Points for a Composition Approach

In order to achieve the targeted composition, we suggest to combine Semantic Web services technology with AI planning [22][8], such as Hierarchical Task Network planning (HTN), Partial-Order Planning (POP), or Forward/Backward Search. The high-level to-be process model serves as a detailed goal description, including a refinement of the goal in the sense of HTN's task decompositions. E.g., the ordering process from Fig. 1 has as overall goal a successful new order, potentially a new customer in the CRM system, and service provisioning to the new customer. The process model itself then can be seen as a refinement of this goal towards an implementation, which orders the various tasks and shows other tasks, such as credit checks, consistency checks in the purchase order, or the distinction between existing and new customers.

Based on such a process model, the composition component then tries to find existing services for each of the tasks, where it is likely that more than one service is required for a task. The combination of all the service calls in the complete process then has to be consistent, such that concurrency of service execution does not violate constraints on service usage, a service call does not undo a previous achievement of another service call, and more. Preconditions and goals for single tasks are given through its semantic mark-up. In part, they may also be refined by inferrence from the bigger process model and environmental context of the process.

One issue with this approach is, that Semantic Web technology is usually based on the open-world assumption, while planning usually assumes a closed world.[9] We are aware of this issue and will take it into account in our approach to composition.

Missing rich semantic description identified as one of the strongest issues in existing composition approaches [11]. We believe in a number of advantages of combining AI planning and Semantic Web technology. One of them is that in former approaches to planning [22], the relationship between subgoals was unclear. By having taxonomies in ontologies, the relationships between

[8] Cf. Part IV, Planning

[9] The closed-world assumption means that any not-stated fact is assumed to be false, while the open-world assumption does not allow this inference: a not stated fact is simply not known.

subgoals should become available for reasoning. Also, non-functional properties can be beneficial, by allowing for service composition with insights into price/cost, duration, security requirements, quality of services, and more. The taxonomies in ontologies should also allow us to deal with differing modelling levels with respect to the objects in a process or the modelling level of a function, in that taxonomy helps identifying the relationship between modelling artifacts on differing levels. Especially subsumption reasoning could prove a valuable tool for this task.

Now, combining these ideas with real-world services and processes should serve us as a very promising starting point: not only do we have activities available for the composition that have a real-world meaning, also the heterogeneities between such activities can be overcome through mediation, and the level of goal-achievement can potentially be assessed during the planning process.

The assumption that implementations are available as services or executable subprocesses can be seen as realistic, e.g. the SAP Enterprise Services Repository contains all kinds of business functions exposed as services. Typically, if a new business process needs to be implemented, the service implementations are available already before its design.

4.2 Analysis of the Computational Feasibility

The composition problem is computationally quite challenging and many problems are undecidable or of exponential complexity [7]. One of them is that comparing behavioral interfaces, e.g., abstract BPEL in the unconstrained case is very similar to the Turing-equivalence problem, and suffers from exponential complexity [8]. This may apply to control flow-based check of discovery outcomes in our approach.

Besides these less encouraging results, there are a few concrete prototypes for automatic Web service composition which allow for a more optimistic outlook. E.g., [18][10] performs composition with 500 services and up to 50 parameters per service in reasonable time frames. Furthermore, in [23] composition based on OWL-S process models is compared to composition based on abstract WS-BPEL processes. There are two timing tables for this issue, showing that the former approach is several orders of magnitude faster than the latter - based on their approach. This approach transforms the domain to a state-transition system, where each possible value for a variable has to be represented in the states - which decreases its applicability in real world scenarios. However, their conclusion is that composition should take place on the appropriate level of abstraction. In general their evaluation shows - at least - that composition problems in reduced domains (limited number of available services and the like) are computable in time frames between fractions of seconds and few hours.

[10] Cf. Section 8.5

5 Related Work

The recent years brought afore a vast number of publications around the service composition topic, just to mention a few: [19, 17, 18, 2]. Amongst them is a collection of surveys on approaches, frameworks, and platforms in the service composition area [11, 5, 16, 20] and more. Some of these surveys formulate needs and motivations for service composition. But, in contrast to this paper, those needs and requirements are rather vague, the focus is on different ways of solving them.

Requirements on service composition are mentioned in a couple of publications. In [14] a list of requirements for a specific software component, the service composer, is given. Naturally, those requirements are much more specialized than the ones given here. [24] discusses among other aspects so-called non-functional requirements in service composition. While those are mostly related to modelling non-functional properties in our words, the point that those should be included in composition is emphasized. Web Service Composition Management, as described in [6], deals with security, fault, accounting, performance and configuration management aspects of service composition, and addresses these issues by giving requirements for this purpose. To the best of our knowledge, general requirements for service composition in the realm of business process enactment as given in this paper have not been published before. Related work that does not focus on requirements only is given in the following.

[11] lists a number of problems in current approaches, which partially overlap with the requirements collected here. Additionally, they describe the problem of degrading quality in long service chains, which we see as minor in the realm of a business process model: the process serves as the first refinement of the business goal, given certain requirements are met. Thus, our approach is going to be guided by the process model, and should therefore produce long service chains only in a user-desired way.

[5] defines seven criteria, along which they evaluate existing composition approaches/frameworks/platforms. Four of those criteria have the characteristics of requirements, and overlap with our requirements.

Business rules in composition play a role in [15] as well, only there they are not used as constraints to the search space, rather, the composition process is defined through composition rules. The composite service then is a result of evaluating the rules.

Business-Driven Development (BDD) [10, 8] is also related, but assumes many manual steps and is more directed to traditional software engineering, in contrast to the service-assumption made here. Probably it is not feasible to automate all the tasks in BDD, but the use of explicit, formal semantics could provide many desired features. If the requirements in this paper are met by a composition system, this system might advance automation in BDD significantly.

6 Conclusions and Future Work

In this paper, we presented the idea of using composition for the implementation of newly created or re-designed high-level business process models. Hereby, a BPEL-like process description is created based on the higher-level model, which orchestrates calls to Semantic Web services that encapsulate the actual business functions. This approach needs to take into account a number of requirements, such as the opportunity to deal with transactional behavior of parts of a process. 14 such requirements are listed in the body of this work. Fulfilling these requirements while solving the composition problem is a hard task, in technical and computational terms. However, a solution to this problem could facilitate tackling the interoperability issue between enterprise application systems. First approaches for the technical solution are given here, followed by a short analysis of the computational complexity.

The approach we take is hard and significantly exceeds today's state of the art, since in practice one can encounter business processes with over a hundred involved activities, running over several months or even years. With that in mind, the computational issue might prove the hardest issue for real world application of the approach. In future work, we plan to attempt solving the described problem step by step, starting with a general framework and extending it one after one for meeting the requirements from this paper. We plan to employ a careful combination of existing techniques together with required extensions in order to create an applicable and well-scaling solution.

References

1. T. Andrews, F. Curbera, and more. *Business Process Execution Language for Web Services*. Version 1.1, 5 2003.
2. B. Benatallah, M. Dumas, Q. Sheng, and A. Ngu. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In *ICDE '02: Proc. 18th Intl. Conf. on Data Engineering*, page 297, Washington, DC, USA, 2002. IEEE Computer Society.
3. M. Burstein, J. Hobbs, and more (eds.). *OWL-S: Semantic Markup for Web Services. OWL-S 1.1.* http://www.daml.org/services/owl-s/1.1/, Nov. 2004. Version 1.1.
4. F. Casati, M. Sayal, and M.-C. Shan. Developing E-Services for Composing E-Services. In *CAiSE '01: Proc. 13th Intl. Conf. on Advanced Information Systems Engineering*, pages 171–186, London, UK, 2001. Springer.
5. S. Dustdar and W. Schreiner. A Survey on Web services Composition. *Int. J. Web and Grid Services*, 1(1):1–30, 2005.
6. B. Esfandiari and V. Tosic. Requirements for Web Service Composition Management. In *Proceedings of the 11th HPOVUA Workshop*, June 2004.
7. R. Hull, M. Benedikt, V. Christophides, and J. Su. E-Services: a Look Behind the Curtain. In *Proc. 22nd Symposium on Principles Of Database Systems – PODS, San Diego, CA, USA*, pages 1–14, June 2003.

8. J. Koehler, R. Hauser, J. Küster, K. Ryndina, J. Vanhatalo, and M. Wahler. The Role of Visual Modeling and Model Transformations in Business-driven Development. In *5th Intl. Workshop on Graph Transformation and Visual Modeling Techniques*, April 2006.
9. P. Küngas and M. Matskin. Detection of missing web services: The partial deduction approach. *Special Issue on Recent Innovations in Web Services Practices, Intl. Journal of Web Services Practices*, 1(1-2):133–141, 2005.
10. J. Küster, J. Koehler, and K. Ryndina. Improving Business Process Models with Reference Models in Business-Driven Development. In *Proc. 2nd Workshop on Business Processes Design (BPD'06), LNCS, Springer*, 2006.
11. U. Küster, M. Stern, and B. König-Ries. A Classification of Issues and Approaches in Automatic Service Composition. In *Proc. 1st Intl. Workshop on Engineering Service Compositions (WESC'05)*, pages 25–33, Dec. 2005.
12. U. Kuter, E. Sirin, D. S. Nau, B. Parsia, and J. A. Hendler. Information Gathering During Planning for Web Service Composition. In *ISWC 2004: Proc. 3rd Intl. Sem. Web Conf., Hiroshima, Japan*, pages 335–349, 2004.
13. S. McIlraith and T. C. Son. Adapting Golog for composition of semantic Web services. In *KR-02: Proc. of the 8th Intl. Conf. on Principles and Knowledge Representation and Reasoning, Toulouse, France*, 2002.
14. H. Meyer and D. Kuropka. Requirements for Service Composition. Technical report, Universität Potsdam, Hasso-Plattner-Instituts für Softwaresystemtechnik, Technische Berichte Nr. 11, Univ.-Verlag Potsdam, 2005.
15. B. Orriëns, J. Yang, and M. P. Papazoglou. A Framework for Business Rule Driven Service Composition. In *TES-03: Proc. 4th Intl. Workshop on Technologies for E-Services, Berlin, Germany*, pages 14–27, Sept. 2003.
16. J. Peer. Web Service Composition as AI Planning - A Survey. Technical report, Univ. of St. Gallen, Switzerland, 2005.
17. M. Pistore, P. Traverso, and P. Bertoli. Automated Composition of Web Services by Planning in Asynchronous Domains. In *ICAPS-05: Proc. Intl. Conf. on Automated Planning and Scheduling*, 2005.
18. J. Rao. *Semantic Web Service Composition via Logic-based Program Synthesis*. PhD thesis, Norwegian University of Science and Technology, 2004.
19. J. Rao, D. Dimitrov, P. Hofmann, and N. Sadeh. A Mixed Initiative Approach to Semantic Web Service Discovery and Composition: SAP's Guided Procedures Framework. In *ICWS-06: Proc. IEEE Intl. Conf. on Web Services, Chicago, USA*, Sept. 2006.
20. J. Rao and X. Su. A Survey of Automated Web Service Composition Methods. In *SWSWPC*, pages 43–54, 2004.
21. D. Roman, H. Lausen, and U. Keller. *D2v1.2. Web Service Modeling Ontology (WSMO) WSMO Final Draft 13 April 2005*. wsmo.org, 2005.
22. S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.
23. P. Traverso and M. Pistore. Automated Composition of Semantic Web Services into Executable Processes. In *ISWC-04: Proc. 3rd Intl. Semantic Web Conf, Hiroshima, Japan*, pages 380–394, 2004.
24. M. T. Tut and D. Edmond. The Use of Patterns in Service Composition. In *AiSE '02/ WES '02: Intl. Workshop on Web Services, E-Business, and the Semantic Web*, pages 28–40, London, UK, 2002. Springer.
25. W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. *Distrib. Parallel Databases*, 14(1):5–51, 2003.