3-Level Service Composition and Cashew: A Model for Orchestration and Choreography in Semantic Web Services

Barry Norton and Carlos Pedrinaci

Knowledge Media Institute, Centre for Research in Computing,
Open University, Milton Keynes, UK
{ b.j.norton | c.pedrinaci } @open.ac.uk

Abstract. There are two types of behavioural model in the WSMO semantic description of services: an orchestration and a choreography, together called the interface. While an orchestration defines a service's behaviour as a composition of existing parts, a choreography is intended to document the conversation of messages exchanged with a single client. In this paper we present a three-level model for behavioural descriptions, and how the Cashew workflow model fits into this, building on existing work in, and establishing connections with, semantic web services, workflow, and software engineering design.

1 Introduction

Cashew is an ontological model for workflow-oriented descriptions of semantic web service interfaces, descended from an earlier generalised representation of the OWL-S process model [10], called CASheW-S [22]. The definition of Cashew has extended, from this previous work, to accomodate WSMO [8] in three ways:

– whereas previously only orchestrations were described in workflow terms, with primitive choreographies being implicitly described via automata, now choreographies are also given a high-level description;
– whereas previously the 'unit of composition' was an operation on a service, now orchestrations compose goals, and choreographies compose operations;
– whereas previously dataflow connections were simple, specifying only performance outputs as sources and performance inputs as targets, now these are specified as types of WSMO mediators.

As well as this alliance with WSMO, the redesign of Cashew has attempted to build stronger links with two other communities. Firstly, the workflow control forms used have been re-examined in the context of 'workflow patterns', where the aim is to standardise the vocabulary for workflow in business process management [26]. Secondly, for the visual representation of workflow models we have looked at UML, which aims at a standard language for software design [17]. We consider all of these as background work in the following section, propose our language in Section 2, consider its representation in UML in Section 3 and then conclude and consider future work in Section 4.

1.1 OWL-S

The process model in OWL-S is an algebra of workflows, called processes, where the atomic processes are grounded to operations on web services. Although this has been called ‘service composition’, we have previously pointed out [20] that this is really a model of ‘operation composition’, *i.e.* a composite process is used to define a single operation, not a general service with multiple operations, and within that definition, the attachment to services of operations is not considered.

The gap between services and operations widens further when we consider statefulness of interaction in the service interface. When we allow that there is a ‘protocol’ governing the order of use of operations of a service in any given session, it becomes important that we consider the notions of service and session, and also that we give a semantic model to this protocol.

It is for these reasons that we consider that the OWL-S process model is only a useful model for orchestrations formed over stateless services, but we shall see that it is deficient when we must deal with services with protocols.

1.2 WSMO

The fragment of the WSMO meta-model [8] we deal with is represented, as a UML class diagram, in Figure 1. The central concepts, duly shown centrally, are ‘web service’, ‘mediator’ and ‘goal’.

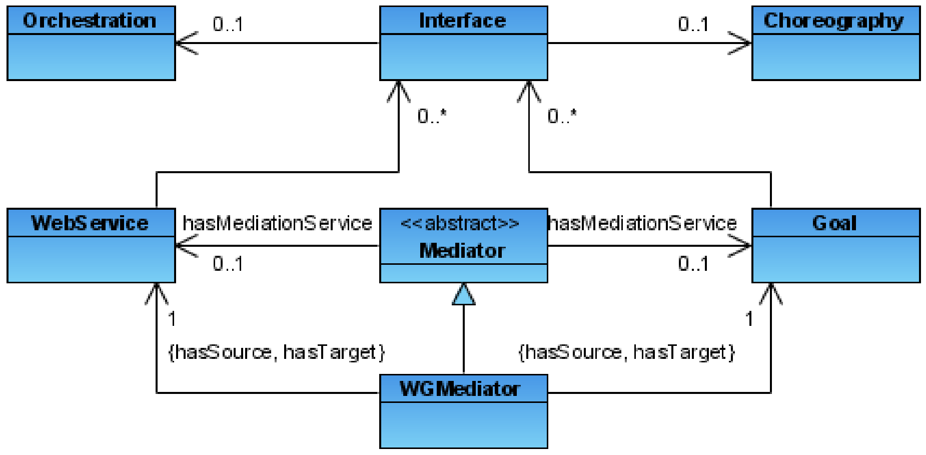

Fig. 1. (Partial) WSMO Meta-model

An instance of ‘web service’ is the basis for a semantic description of a web service, divided into two parts: the ‘capability’, not shown or considered in this paper, contains the *functional* description of the service; and the ‘interface’, which contains *behavioural* descriptions.

Following the connections from interface, we see that each instance here can contain an orchestration and/or a choreography. We also see that the description of a goal may contain interfaces, so that goals are more than simply first class representations of what have been called 'service templates', *i.e.* only functional descriptions of service requirements, following the initial publication of OWL-S.

So far in WSMO, the formal basis of orchestration and choreography in the form of abstract state machines (*ASM*s) [2] has been prescribed. Thereafter use of ASM choreographies to define the protocol that a client must use in order to consume a service has been defined [9] and exemplified [11]. The use of ASMs for orchestration is less developed [7].

Even the restricted two-party view on the definition of choreography that WSMO takes[1] is open to refinement. The DIP project[2] has proposed [15] that the distinction that the IRS [4], one of two DIP implementations of the WSMO model, makes between *service-* and *client-choreographies* [5] should be formalised in WSMO. The first difference between the two is the viewpoint: in a service choreography, the service documents which communications are offered to clients; in a client choreography, the client documents which communications they offer and accept in return. Although to some degree orthogonal, the second difference is that a client choreography will be finite and contain only those communications intended to achieve some goal, whereas the service choreography will document all allowed interactions with the service, which may be infinite.

The latter form, which DIP calls service choreography, is what is documented in the Amazon Case Study [11], and shows an infinite behavioural model where the client can interact with the Amazon service by searching at any point, and interleave this with logging in, logging out and making purchases. An example of a client choreography might connect a goal 'buy the most recent edition of book x by author y' to this service via the refined conversation consisting of search, followed by log-in, purchase and log-out. Due to this goal-orientation of client choreographies, the meta-model in this paper will suggest, as detailed in Section 2, that client choreographies are actually choreographies attached to wg-mediators, a specialisation of mediators that, as shown in Figure 1, are used to attach goals to services.

The meta-model proposed in this paper will also follow IRS, in turn following the literature in problem-solving methods, in insisting that the choreographies of goals are always 'one-shot', *i.e.* that in expressing the desire to achieve a goal, a user does not have to worry about 'control flow', or complex interaction, and merely present some inputs and waits for an output. We will also follow the IRS in viewing orchestrations as a composition of goals, *i.e.* the units of composition in an orchestration will be abstracted to this level, and goal-based invocation will be used to match these goals to service at run-time, which may involve the use of discovery. In this way, the use of several operations of the same service can be abstracted into a goal that may then be considered atomic for composition.

[1] Which can be contrasted with the multi-party viewpoint of W3C [13], but also seen as a projection on this.

[2] http://dip.semanticweb.org

1.3 Workflow Patterns

'Workflow Patterns' are the result of a long-running project to formalise the possible variations between workflow systems, and provide a common vocabulary to compare these [26]. It is telling that, when presented[3], one of the examples given of hidden differences between workflow systems in the early days was the distinction between, in workflow patterns terms, 'XOR' and 'Deferred Choice'. In general terms, when asked whether their systems supported a choice operator, vendors would answer 'yes'. On the other hand, given the vocabulary to ask whether systems supported choices resolved internally by the engine and choices resolved externally by the outcome of component tasks, the answer was too often only 'yes' one or the other, rather than both. In fact, this very example is another reason for the deficiency of OWL-S in the presence of choreography, since deferred choice is not supported by OWL-S. We shall show the extension to, and use of, this form of choice operator in the Cashew model.

1.4 UML

The UML is an ongoing effort by the Object Management Group [4], standardised by ISO [17], to provide a language for software engineering design via diagrams describing both the static and dynamic characteristics of software artifacts. Of particular relevance are Activity Diagrams which, it has been suggested, are expressive enough to represent visually many of the Workflow Patterns [6] [27].

2 The Cashew Model

An extended '3-level' WSMO meta-model, as proposed in the DIP project, is diagrammed in Figure 2. As sketched above, the concepts of both orchestration and choreography are abstracted, so that ASMs form just one way to represent these. The upper two levels represent a high-level view on these behavioural models, one oriented towards first-class workflow features and another towards diagramming for human inspection. In the DIP project these levels are filled by the Cashew workflow language and by UML Activity Diagrams. These are not the only possibilities, however, and in Section 4 we discuss others.

The lower half of the diagram documents a proposed extension to WSMO that has already been abstracted out from the DIP work and proposed to the working group [21]. The notion of 'Performance', due to OWL-S, allows us to hierarchically compose workflows from *performances* of workflows, so that each defined workflow can be reused in other contexts than where it was defined, and each instance is given a different identifier in context.

The details of the Activity Diagram meta-model — called 'ADO', the Activity Diagram Ontology — are not reproduced here; the reader is directed to the annex [12] of the relevant DIP deliverables.

[3] Wil van der Aalst's 'Life After BPEL?', presented as keynote at WS-FM'05

[4] http://www.omg.org/technology/documents/formal/uml.htm

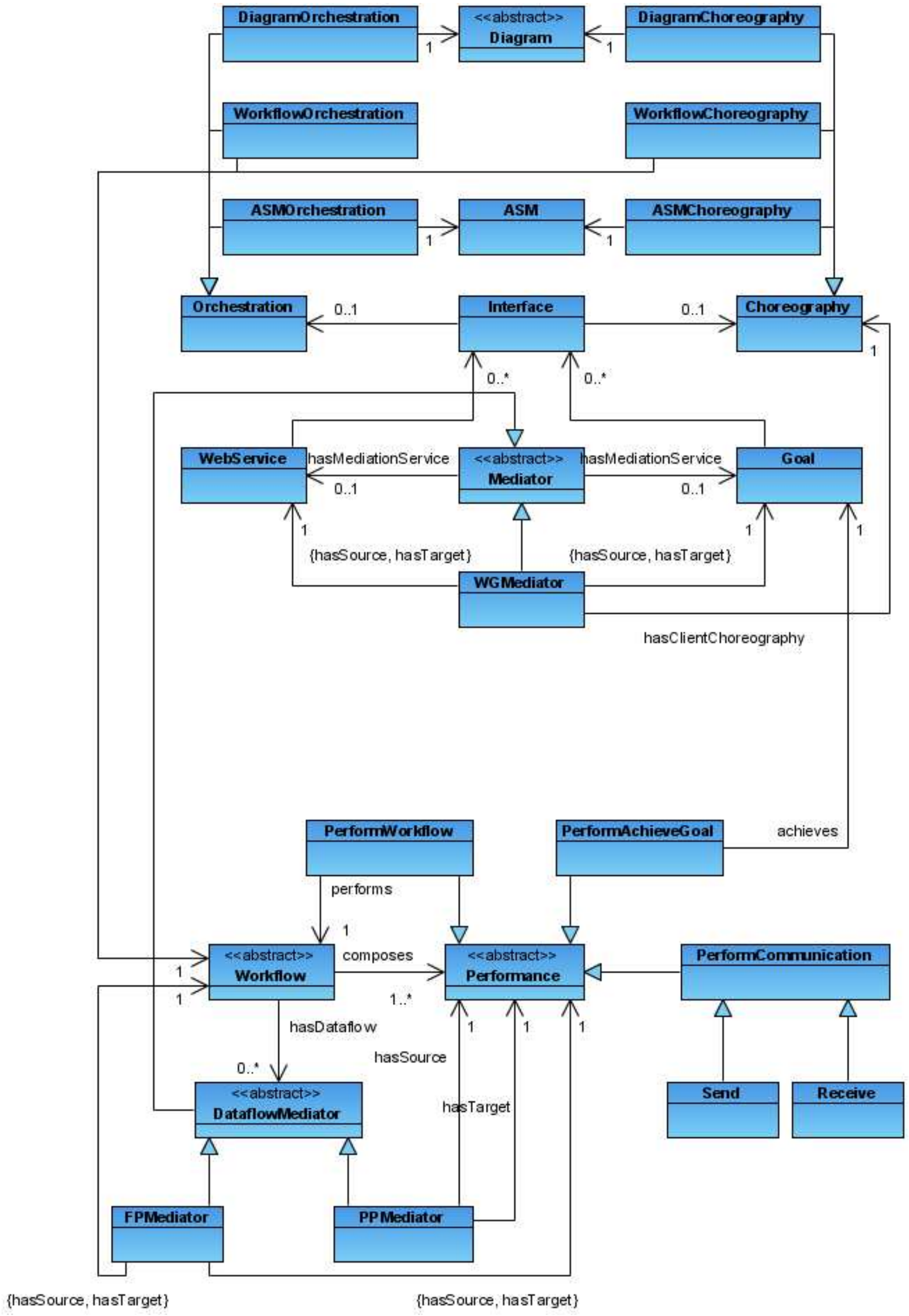

Fig. 2. Extended WSMO Meta-model

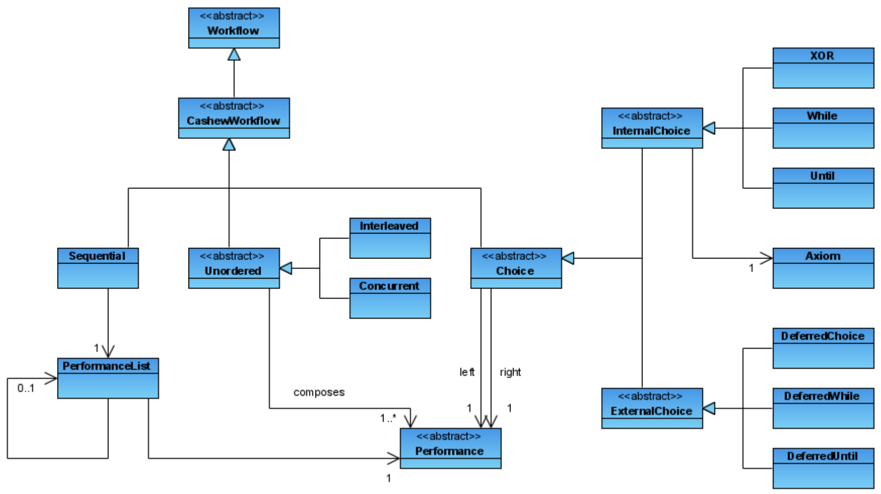

Fig. 3. Cashew Workflow Meta-model

The details of the Cashew workflow meta-model, extending the new concept *Workflow*, are diagrammed in Figure 3. The workflow operators are divided into three types: 'Sequential' depends on an ordered list of performances; 'Concurrent' — called 'Split-Join' in OWL-S — and Interleaved — called 'Any-Order' in OWL-S, and renamed as a shortened form of the Workflow Pattern 'Interleaved Parallel Routing' — both rely on an *unordered* set of performances; *choices* abstract over once-off choices and loops over exactly two performances.

The subset of operators to which 'Internal Choice' is a superconcept represent exactly those in OWL-S — with 'If-Then-Else' renamed after the Workflow Pattern 'XOR' — but substituting the WMSO concept 'Axiom' for the condition that the engine will evaluate to resolve the choice. In the case of 'XOR' the condition will only be evaluated once to chose between the left and right performance; in the case of 'While' and 'Until', after the left performance is evaluated, which will happen without evaluating the condition in the first instance with 'Until', the condition will be evaluated again.

In our previous semantics for OWL-S [22], we paid careful attention to the 'Any-Order' operator, elided in other semantics [1]. In the informal semantics published in the specification [10] it is stated that only one performance at a time will be executed, and that the performance to be executed at run-time will depend on availability of input data, since component performances may communicate to supply one another with data. This data-driven characteristic is in contrast to the control-driven 'flow' processes, due to WSFL [19], in BPEL [16].

In the spirit of this data-driven approach, since this happens to coincide with our own previous work [23], we offered an alternative semantics for 'Choose-One', where a non-deterministic choice would be made only between the 'ready' branches, *i.e.* those whose input has been provided. In the case that all branches are ready, this is an equivalent non-deterministic choice. In the case that different outputs can be produced, *e.g.* by the invocation of an operation — not considered in OWL-S, but expected in WSMO — this allows the choice to be resolved externally between subsequent performances depending on the different messages. Furthermore, given the extension to explicit message receipts from the client, having extended the types of performance, this becomes the 'classical' deferred choice workflow pattern, which we therefore claim to generalise on, and name our operator after. We extend this 'external choice' to be able to decide also loops, as shown in the remaining concepts shown in Figure 3.

3 Representing Cashew in UML

The alliance with workflow patterns allows a standard mapping from most parts of Cashew directly into UML Activity Diagrams [27]. Rather than detail the whole translation here, we concentrate on the distinction between XOR and Deferred Choice discussed in the previous section.

For a XOR-type workflow between performances **V** and **W**, with axiom a, the resulting diagram fragment is as shown in Figure 4. A performance of this workflow would connect in control flow at the 'decision node' (the upper diamond), and connect it out at the 'merge node' (the lower diamond). It is an important part of the translation that each stage defines these two points uniquely, in order to be composable.

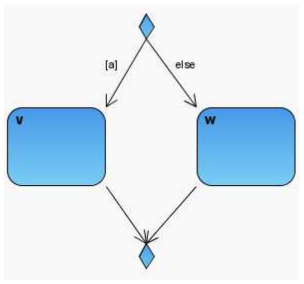

Fig. 4. XOR in UML

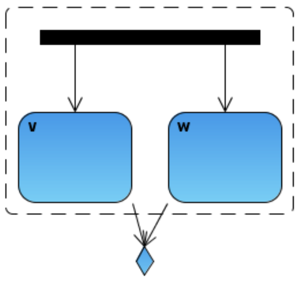

Fig. 5. Deferred Choice in UML

The Deferred Choice-type workflow between performances **V** and **W** is shown in Figure 5. Although this starts concurrently — the incoming control flow connects to the 'split node' (the horizontal bar) — there is within an 'interrupting region' (the dashed box) and each 'interupting edge' preempts the other.

This can be contrasted with the representation of a performance of a concurrent workflow over performances **W1 .. Wn**, shown in Figure 6, where there is no interruptable region and the outgoing control flow resynchronises on a 'join node', rather than the 'merge node' in Figure 5, so every thread must complete.

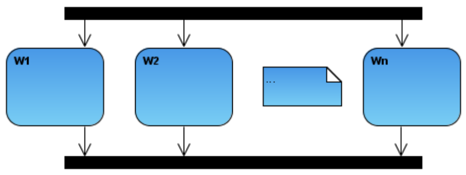

Fig. 6. Concurrent in UML

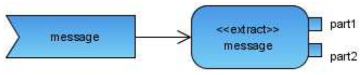

Fig. 7. Receive in UML

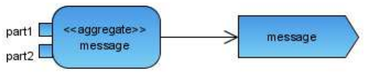

Fig. 8. Send in UML

In order to diagram the performance of sequential workflows, we simply create a control flow arrow between the representation of each successive performance, with the incoming control flow connecting to the first, and out-going to the last.

In order to define loops we allow the control flow to connect back to the 'left' performance via a 'merge node', after either a 'decision node' or an interuptable region in the form of Figures 4 and 5 respectively.

In order to easily define dataflow, each send and receive performance is associated with an 'action' with 'pins' to show the unpacked version of each message, as shown in Figures 8 and 7. In this way, dataflow can be represented by connecting pins together with edges that UML calls 'object flow edges'. In fact, in WSMO terms, the dataflow edges represent mediators, as proposed in [3], and it is up to the mediators to specify the relationship between the message and the part needed, in the capability of their associated mediation goal or service, but this is a diagrammatic convenience.

4 Conclusions and Further Work

In this paper we have shown a three-level representation of behavioural models and how this is compatible with WSMO. We have shown how Cashew and the UML can be fitted into this model, allowing a relationship to be established with many existing communities. Finally, we have sketched how translation from a Cashew workflow model to an Activity Diagram model can be carried out.

The key advantages of the three-level model are the ability to deal with services with complex choreographies within orchestrations, and also to abstract from the details of sessions with these services in a goal-oriented fashion. The key advantages of Cashew are the ability to express the two kinds of choreography necessary to do this, including the distinction between internal and external choices which our example has shown is core to the WSMO notion of choreography, and the ability to communicate with the practitioners and tools of other communities, via the alliance with workflow patterns and UML diagrams.

Existing implementation involves an interpreter for Cashew in the IRS [4], and an on-going implementation of translation from Cashew to UML and a (partial) reverse-translation. Future work involves implementation of translations from Cashew and Activity Diagrams to Abstract State Machines, and implementation of an orchestration engine based on Abstract State Machines in WSMX [18], the open-source reference implementation for WSMO.

Further on-going work involves the creation of an extended WSML grammar in which interfaces can be expressed in both Activity Diagrams and Cashew, as well as ASMs, and the proposal of further parts of the three-level model, with these concrete instances, to the WSMO/WSML Working Groups[5].

Within the SUPER project[6] the intention is to find a synergy between business process management and semantic web services. Early results in this project suggest that a three-level model, where an upper diagram-oriented representation is based on event-driven process chains (EPCs) [25], a middle level on a 'semanticised BPEL' and the bottom-layer on SWS-oriented representation may be useful. It is hoped that the three-level model here can be extended to express this in WSMO-compatible terms

5 Acknowledgements

This work is supported by the DIP project, an Integrated Project (no. FP6 - 507483) supported by the European Union's IST programme. We should like to particularly acknowledge the contribution of the DIP partner ILOG, specifically Laurent Henocque and Mathias Kleiner.

References

1. A. Ankolekar, F. Huch, and K. Sycara. Concurrent semantics for the web services specification language DAML-S. In *Proc. 5th Intl. Conf. on Coordination*, volume 2315 of *LNCS*, 2002.
2. E. Börger and R. Stärk. *Abstract State Machines*. Springer, 2003.
3. L. Cabral and J. Domingue. Mediation of semantic web services in IRS-III. In *Proc. Workshop on Mediation in Semantic Web Services (MEDIATE 2005), in conjunction with ICSOC 2005*, 2005.
4. J. Domingue, L. Cabral, F. Hakimpour, D. Sell, and E. Motta. IRS-III: A platform and infrastructure for creating WSMO-based semantic web services. In *Proc. of the Workshop on WSMO Implementations (WIW 2004)*, volume ISSN 1613-0073. CEUR Workshop Proceedings, 2004.
5. J. Domingue, S. Galizia, and L. Cabral. Choreography in IRS-III: Coping with heterogeneous interaction patterns. In *Proc. 4th Intl. Semantic Web Conference (ISWC 2005)*, number 3729 in LNCS, 2005.
6. M. Dumas and A. H. M. ter Hofstede. UML Activity Diagrams as a workflow specification language. In *Proc. 4th Intl. Conf. on the Unified Modeling Language (UML)*, number 2185 in LNCS, 2001.
7. D. Roman *et al.* Orchestration in WSMO (working version). http://www.wsmo.org/TR/d15/v0.1/, January 2005.

[5] http://www.wsmo.org/

[6] http://super.semanticweb.org/

8. D. Roman *et al.* Web service modeling ontology WSMO v1.2. http://www.wsmo.org/TR/d2/v1.2/, April 2005.
9. D. Roman *et al.* Ontology-based choreography of wsmo services v0.3. http://www.wsmo.org/TR/d14/v0.3/, May 2006.
10. David Martin *et al.* OWL-S: Semantic markup for web services. http://www.daml.org/services/owl-s/1.1/overview/, 2004.
11. J. Kopecky *et al.* WSMO use case: Amazon e-commerce service v0.1. http://www.wsmo.org/TR/d3.4/v0.1/, December 2005.
12. M. Stollberg *et al.* DIP interface description ontology. http://dip.semanticweb.org/documents/DIO-Annex-to-D3.4-and-D3.5.pdf, January 2005. Annex to DIP Deliverables D3.4 and D3.5.
13. N. Kavantzas *et al.* Web services choreography description language v1.0. http://www.w3.org/TR/ws-cdl-10/, November 2005.
14. S. Bhiri *et al.* An orchestration and business process ontology. http://dip.semanticweb.org/documents/D3.4.pdf, January 2005. DIP Deliverable D3.4.
15. S. Galizia *et al.* An ontology for web service choreography. http://dip.semanticweb.org/documents/D3-5.pdf, January 2005. DIP Deliverable D3.5.
16. S. Thatte *et al.* Business process execution language for web services version 1.1. ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf, 2003.
17. Object Management Group. UML 1.4.2 specification. Technical Report ISO/IEC 19501, ISO, 2005.
18. A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. WSMX - a semantic service-oriented architecture. In *Proc. 4th Intl. Semantic Web Conference (ISWC 2005)*, number 3729 in LNCS, 2005.
19. F. Leymann. Web services flow language (WSFL 1.0). http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, 2001.
20. B. Norton. Experiences with OWL-S, directions for service composition: The Cashew position. In *OWL: Experiences and Directions Workshop (co-located with ESWC 2005)*, 2005. http://www.mindswap.org/OWLWorkshop/sub23.pdf.
21. B. Norton. Dataflow for orchestration in WSMO. http://www.wsmo.org/TR/d15/d15.1, July 2006.
22. B. Norton, S. Foster, and A. Hughes. A compositional semantics for OWL-S. In *Proc. 2nd Intl. Workshop on Web Services and Formal Methods (WS-FM 05)*, number 3670 in LNCS, Sept 2005.
23. B. Norton, G. Lüttgen, and M. Mendler. A compositional semantic theory for synchronous component-based design. In *14th Intl. Conference on Concurreny Theory (CONCUR '03)*, number 2761 in LNCS. Springer-Verlag, 2003.
24. M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. 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.
25. W. M. P. van der Aalst. Formalization and verification of event-driven process chains. *Information & Software Technology*, 41(10):636–650, 1999.
26. W. M. P. van der Aalst, A. H. M ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(3):5–51, June 2003.
27. P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Ruseell. Pattern-based analysis of UML activity diagrams. BETA Working Paper Series WP 129, Eindhoven University of Technology, 2005.