Protocol Mediation for Adaptation in Semantic Web Services

Stuart K.Williams[1], Steven A. Battle[1], and Javier Esplugas Cuadrado[2]

[1] Hewlett-Packard Laboratories, Filton Road,
Stoke Gifford, Bristol, BS34 8QZ, UK
{skw, steve.battle}@hp.com
[2] Hewlett-Packard Espanola SL, Jose Echegaray n°8,
La Rozas, Spain. 28230
javier.esplugas.cuadrado@hp.com

Abstract. Protocol mediation enables interaction between communicating parties where there is a shared conceptual model of the intent and purpose of the communication, and where the mechanics of communication interaction vary. The communicating partners are using different protocols to achieve the same or similar ends. We present a description driven approach to protocol mediation which provides a more malleable approach to the integration of web services than the current rigid 'plug-and-socket' approach offered by description technologies such as WSDL. It enables the substitution of one service provider with another even though they use different interaction protocols. Our approach is centred on the identification of common domain specific protocol independent communicative acts; the description of abstract protocols which constrain the sequencing of communicative acts; and the description of concrete protocols that describe the mechanisms by which the client of a web service interface can utter and perceive communicative acts.[1]

1 Introduction

Web service technology places powerful tools in the hands of developers enabling independent invention and re-invention of web service interfaces. Businesses will develop and deploy web service interfaces to visible aspects of their business process. Many of these interfaces encapsulate similar if not identical concepts. However the factoring of otherwise similar interfaces will vary. The mechanics of interaction protocols will differ. Yet conceptually they encapsulate similar if not identical interaction metaphors. Consider the familiar catalogue, cart, checkout metaphor of a typical eCommerce web site. The human user is guided through the process by their recognition of the metaphor, their intuition about the process they are engaged in and the continuous guidance provided by the user interface decoration (labels on buttons, explanatory text etc.). The human user is unconstrained about which of many available on-line stores they trade with. Our aim is to provide a similar level of flexibility for automated web service clients in the selection and use of service providers.

[1] This work was conducted as part of the EU funded Semantic Web enable Web Services project (SWWS EU IST-2002-37134).

Y. Sure and J. Domingue (Eds.): ESWC 2006, LNCS 4011, pp. 635 – 649, 2006.

By analogy, WSDL [1] supports description of the syntactic operation of individual user interface controls; BPEL [2] describes the service provider processes which respond to control invocations; WS-CHOR [3] describes a global view of the sequencing constraints on externally visible messages exchanged between multiple parties in web service interactions. However, in the current web service stack there is no machine readable account of what a particular web service interaction or sequence of web service interactions actually accomplish.

In this paper we describe a framework for providing rich service descriptions that enable web service clients to adapt their interaction behaviour to the constraints of a particular provider's web service interface. This removes cost and time from the process of integrating new service providers and enables consumers of web services greater freedom and flexibility to dynamically choose service providers. For service providers it also means access to a broader customer base and results in a service oriented economy where service consumer/provider relationships are formed on the basis of business fundamentals without requiring an exact fit between the client and provider sides of a particular web service interface.

In section 2 we introduce the topic of protocol mediation more fully. In section 3 we introduce a case study scenario drawn from the IST EU Semantic Web enabled Web Services project (SWWS EU IST-2002-37134) which we use as a running example through the remainder of the paper. Section 4 gives a detailed presentation of the protocol mediation framework developed in the SWWS project. Section 5 describes our interface description language. Section 6 discusses related work. Finally section 7 presents our conclusions and ideas for further work.

2 Protocol Mediation

Bridging or gatewaying between compatible protocols has been studied since the 1980's [11, 13, 14] continuing through a period of considerable work in the field of Open Systems Interconnection (OSI) [8, 12] and the Internet. Our work on protocol mediation draws inspiration from that work. We make particular use of the concepts of abstract service definition [8, 12] and gateways/half-gateways. Much of the previous work was focussed on mediating between protocols established by standardisation processes. In contrast, our work on protocol mediation is focussed on the dynamic instantiation of description driven mediation behaviour. Our work is motivated by the existence of similarly intentioned, independently created, and evolving protocols which are an inevitable consequence with the successful adoption of Web Service technologies.

Figure 1 illustrates protocol mediation the form of a protocol gateway made up of two half gateways and a relaying function. Two processes, X and Y, wish to communicate with one another at a business level. Each process adopts some role with respect to the interaction. For example, process Y may act on behalf of the provider of some (business) service, while process X may act on behalf of a consumer of that service. Unfortunately, process X and process Y communicate using two different protocols, P1 and P2, each of which has capabilities C1 and C2 respectively, expressed in the abstract as the protocol layer services of P1 and P2.

Clearly if effective communication is to occur between processes X and Y, some mediation must occur.

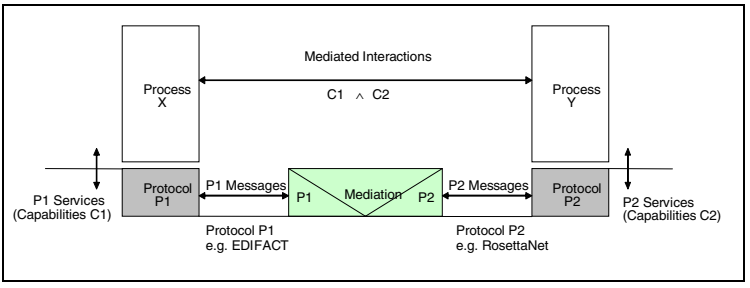

Fig. 1. Protocol Mediation – A conceptual model

It should be equally clear that it is only possible to mediate in the intersection of the capabilities of the two protocols. If the mediated channel becomes too impoverished to support the required interaction, some other approach becomes necessary and either process X or Y or both needs to have their behaviour changed to address the missing capability in some other way. This is known as "process mediation" and is not the subject of this paper.

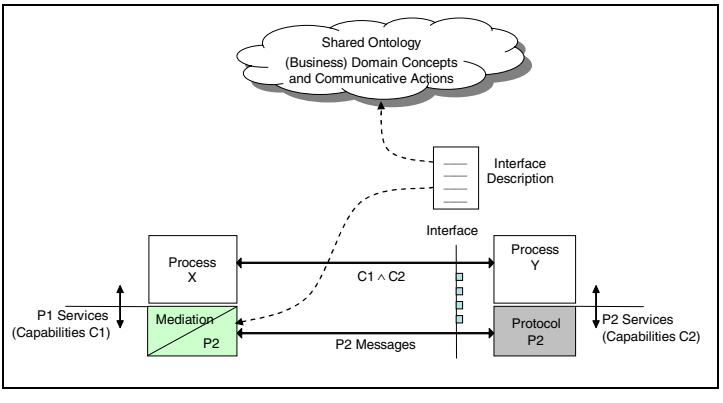

Fig. 2. Description Driven Adaptation

It is possible to associate the mediation element more strongly with one party or the other. One can slide the mediation element toward, say, process X. At some point, the presence of P1 in the system becomes somewhat vestigial and the mediation component becomes logically absorbed within the infrastructure supporting process X. Figure 2 illustrates this diagrammatic manipulation. Process X continues to make use of P1's protocol layer services and capabilities (restricted to those that lie at the intersection with protocol P2's capabilities). However, only protocol P2 messages are exchanged externally. The P2 protocol provider and process Y are unchanged. Figure 2 also introduces the notion of there being an exposed interface at process Y which is described with a rich behavioural description which is consumed by a mediation component.

Our behavioural descriptions rely on abstracting the communicative actions [5, 6] of a protocol from the underlying mechanisms of that communication. This echoes the practice of the OSI community [8, 12] of specifying the service abstraction separately from its message vocabulary, encoding and rules of procedure. However we make our descriptions machine readable and interpretable by a mediation component. Conceptually we regard an interaction protocol as animating domain concept instances and the communicative acts which result in changes in their state are themselves part of the ontological structure of the domain. Thus, the ontology of the interface description is what needs to be shared between partners rather than prior agreement on a specific interaction protocol.

The description driven adapter of Figure 2 may be thought of as a 'half-gateway' and it should be possible to use two such structures and descriptions of the interfaces that they each face to create a description driven 'full-gateway' or mediator structure as shown in Figure 1. It should be apparent that rather than relaying protocol messages between protocol P1 and protocol P2, such a mediator relays the common communicative actions of protocols P1 and P2.

3 Logistics Scenario

Figure 3 illustrates the supply chain logistics scenario used as a case study in SWWS to motivate our work [9, 10].

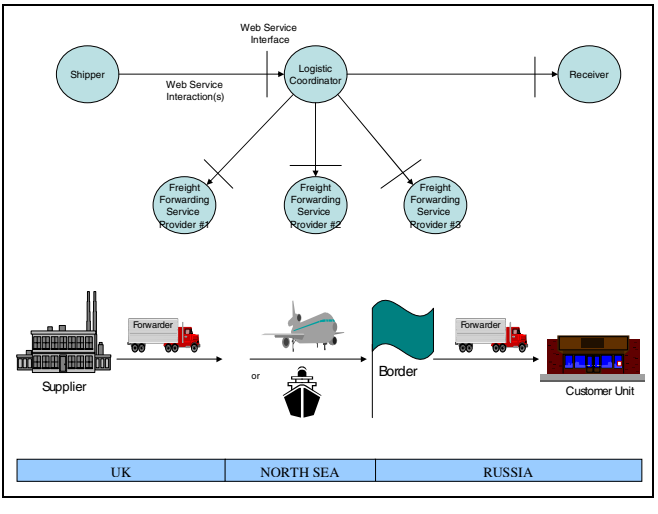

Fig. 3. Multi-leg Shipment Logistics Scenario

The diagram illustrates four different logical roles: Shipper, Logistic Coordinator, Freight Forwarding Services Provider and the Receiver. The scenario requires replacement of Freight Forwarder #2. The replacement provider uses RosettaNet [15] for interacting with the logistics coordinator whereas the replaced provider uses

EDIFACT [16]. This choice of message sets is compounded by local variations in the way that different businesses use the message formats.

Our goal is to provide a rich description of the interaction protocol use across a freight forwarding service provider's web service interface. Our intent is that the user of an interface has a rich enough description of the syntax and semantics of the interface to enable it to adapt its behaviour to the constraints of that interface.

4 The SWWS Protocol Mediation Framework

Under the assumption that we are not at liberty to redesign, alter or replace an existing interaction protocol, our approach is to provide a sufficiently rich machine readable description of the protocol. A mediation component within the client system can then adapt its interaction behaviour to meet the interface constraints of the service provider in much the same way as a human user of an eCommerce web site adapts their interaction behaviour on the basis of the controls and surrounding UI narrative presented to them.

Thus, classic web service clients can use classic integration techniques organized around programmers retrieving WSDL [1] descriptions from UDDI registries in order to write integration code whilst a semantic web service client containing a protocol mediation component retrieves a rich description of the interface and adapts its behaviour to suit.

The following sections introduce the components of our framework: communicative acts [5] and primitives which model the significant domain specific communications between interacting parties; abstract protocols which describe the conversational structure of the exchange of primitives used to model communicative acts and which are used operationally to restrict primitive sequencing; concrete protocols which elaborate the concrete interaction behaviours required to initiate and perceive communicative acts across a particular concrete interface; and message filters which are used to bind inbound messages or web service invocations either to concrete behaviours within existing active conversation instances or to factories that create new conversation instances. Interactions between a service provider or consumer agent and the communication infrastructure are modelled as primitive events accompanied by knowledge bases containing relevant domain instances.

4.1 Roles and Communicative Acts

Our first step is to identify the communicative acts [5, 6] associated with our domain and the roles involved in communication which have some correspondence with the concept of illocutionary particles and roles articulated in the ISLANDER framework [7]. We regard roles and communicative acts as part of the ontology which structures concepts within the domain. In our logistics scenario we identify the following 6 communicative acts that occur between a Logistics Coordinator (LC) and a freight forwarding service provider (FF) about a particular shipment journey leg:

Although only short names are used here, in practice, within a web ontology, the names of all concepts (and communicative acts) are made global through the use of URI [21].

Table 1. Communicative acts involved in a Logistics Journey Leg

Communicative Act	Direction	Communicative intent
informReadyForCollection	LC to FF	Inform the FF that the shipment is available for collection.
requestShipmentStatus	LC to FF	Request an update of the shipment status from the FF.
informShipmentStatus	FF to LC	Inform the LP of the shipment status
informReadyToDeliver	FF to LC	Inform the LP that the FF is ready to deliver the shipment.
informShipmentDelivered	FF to LC	Inform the LP (and provide proof) that the FF has infact delivered the shipment.
requestPayment	FF to LC	Request payment for delivering the shipment from the LP.

The utterance and perception of communicative acts by the logistics coordinator and freight forwarding services provider are significant events in the interaction between partners as the physical movement of the corresponding shipment progresses. We model these events as the service primitives of a communication protocol in the style adopted by the OSI Basic Reference Model [8].

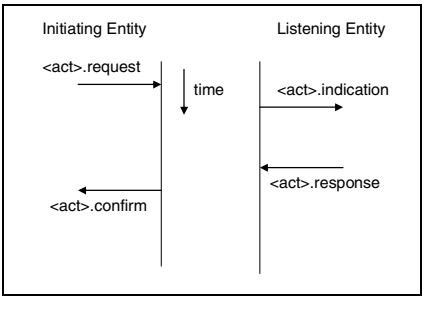

Fig. 4. Sequencing of Abstract Protocol Service Primitives

The occurrence of a communicative act is modelled as the occurrence of 4 primitives illustrated in Figure 4. Two primitive events are experienced at the initiating party which utters the communicative act and, in the absence of failure, two primitive events are experienced at a listening party which perceives the communicative act. The four primitives of the communicative act, `<act>`, model:

- the initiation of the act by the initiating agent, `<act>.request`;
- the perception of the communicative act by a listening/responding agent, `<act>.indication`;
- acknowledgement by the listening/responding agent that the act has been perceived, `<act>.response`;
- and reporting the outcome of the communicative act to the initiating agent, `<act>.confirm`.

The `.response` and `.confirm` primitives effectively model a technical acknowledgement that the communication has reached its intended recipient. Any substantive response motivated by the communicative act itself is modelled as a subsequent communication in the opposite direction. Communicative acts therefore achieve a single domain level communication, but may correspond to an exchange of one or more lower level messages or web service operations.

At the initiator, the outcome of a communicative act falls into one of three broad categories:

- Success: The communication is known (by the initiator) to have reached the intended recipient.
- Exception: The communication is known (by the initiator) to have failed to reach the intended recipient.
- Indeterminate: The outcome of the communicative act is unknown (to the initiator).

This provides the basic framework for modelling communication between agents. Each communicative act may carry information (a message) from initiating agent to responding agent and return status information about the outcome of the communication. An important facet of our model is that the occurrence of a `.request` pri-mitive at the initiator is always followed by an occurrence of a `.confirm` primitive, even if the latter reports that the outcome of the communication is indeterminate or failure.

4.2 Abstract Protocol

The next step in our process is to observe that the sequencing of communicative acts is constrained. In our example scenario, the dialog about a given shipment commences with the utterance of an `informReadyForCollection` and ends either with an `informShipmentDelivered` or a `requestPayment`. The structure of these conversational constraints can be captured in the form a monitoring process which (impractically) takes a global view of the system, the occurrence of a communicative act only being possible when it is admissible by the monitoring process. The behaviour of the monitoring process may be expressed in a number of formalisms, such as the ad-hoc notation in figure 5 or more formally using process algebra's such as CCS [18] or UML style Harel State Charts[17] as in figure 6.

```
seq( informReadyForCollection,
     par( repeat( seq( requestShipmentStatus,
                       informShipmentStatus ) ),
          seq( informReadyToDeliver,
               par( requestPayment,
                    informShipmentDelivered )
          )
     )
)
```

Fig. 5. Simplifed ad-hoc expression of the Abstract Protocol for Journey Leg monitoring and execution

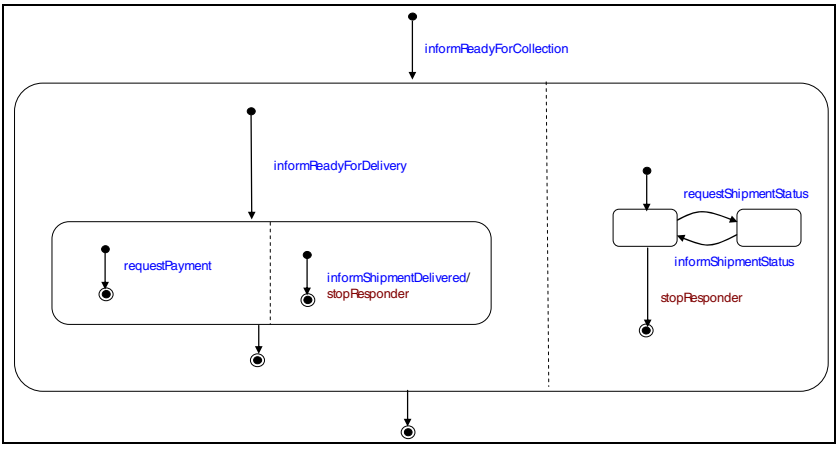

Fig. 6. UML/Harel State Chart expression the Abstract Protocol for Journey Leg monitoring and execution

Note that as specified here these behavioural expressions treat a communicative act as an atomic occurrence, however, as stated earlier we have modelled each as a sequence of four primitives, two of which are experienced by each party to the communication. The basic patterns above can be specialized to the consumer and provider roles with appropriate re-labelling of events. In addition, since the primary motivation for treating a communicative act as four discrete events is to enable explicit consideration of errors handling, different behaviours may be added to cater for the different kinds of outcome listed above: success, exception and indeterminate.

Our concept of an abstract protocol corresponds closely with the ISLANDER [7] concept of a scene and its accompanying dialogical framework.

4.3 Concrete Protocol

In the previous section we considered the role based sequencing constraints on the occurrence of abstract primitives crossing the boundary between a service provider agent or a service consumer agent and the underlying entities that realise concrete interaction behaviours, see figure 7. We now consider the interface specific concrete protocol description which binds the occurrence of these primitives to concrete protocol behaviours. Descriptions are divided into initiating and responding behaviours.

Initiating behaviours are associated with the occurrence of a `.request` primitive and ultimately giving rise to the corresponding `.confirm` primitive. Responding behaviours perceive the occurrence of a communicative act generally through the arrival of a message or an inbound invocation of a web service operation which gives rise to a `.indication` primitive. A responding behaviour may remain active beyond the occurrence of the corresponding `.response` primitive in order to absorb duplicated inbound messages or to repeat apparently lost outbound messages in accordance with the requirements of the concrete protocol.

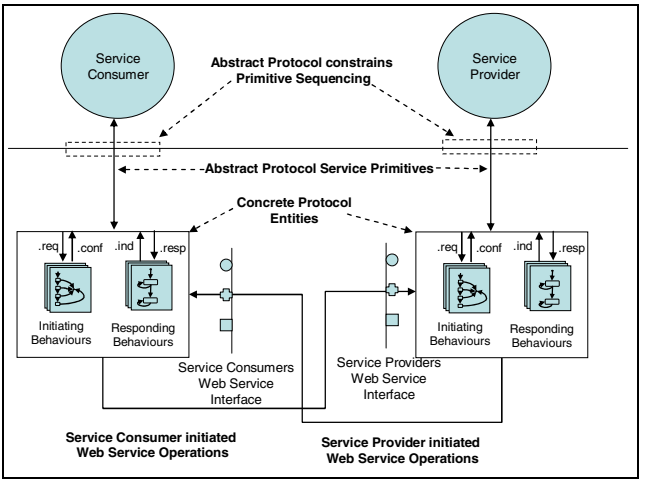

Fig. 7. Abstract and Concrete Protocols

Both initiating and responding behaviours may involve both the sending and receiving of one or more messages or the inbound and outbound invocation of one or more web service operations. For example, in RosettaNet, lost business action messages may be retransmitted a prescribed number of times at prescribed intervals, typically 3 times at 30 minute intervals. This behaviour is embedded in the concrete protocol and is not exposed to the service consumer/provider except in as much as it may give rise to failed or indeterminate outcomes.

The abstract protocol acts as a guard which ensures that abstract service primitives cannot occur except when they are admissible. The concrete protocol descriptions provide an expression of how to initiate and perceive the communicative acts initiated and perceived. These behaviours can also be described as processes using any of the formalisms noted earlier. However the actions associated with state transitions need to be capable of performing simple computations and manipulations on message content. We use a simple event, guard, and action model to described concrete behaviours as simple state machine processes. Figure 8, below, illustrates the concrete RosettaNet protocol behaviour required of a freight forwarding service initiating the `informReadyForCollection` communicative act. Similarly, figure 9 illustrates the corresponding behaviour required of the freight forwarding services provider in order to perceive the occurrence of the same communicative act. One of the important complex operations that we hide here is the extraction of domain instance information from inbound messages and the generation outbound message content from the instances of the domain ontology. This is the problem of data mediation, and our approach to this is described elsewhere [19]. The operation of these concrete behaviours coordinates the lifting and lowering of domain knowledge between message structure and ontology instances.

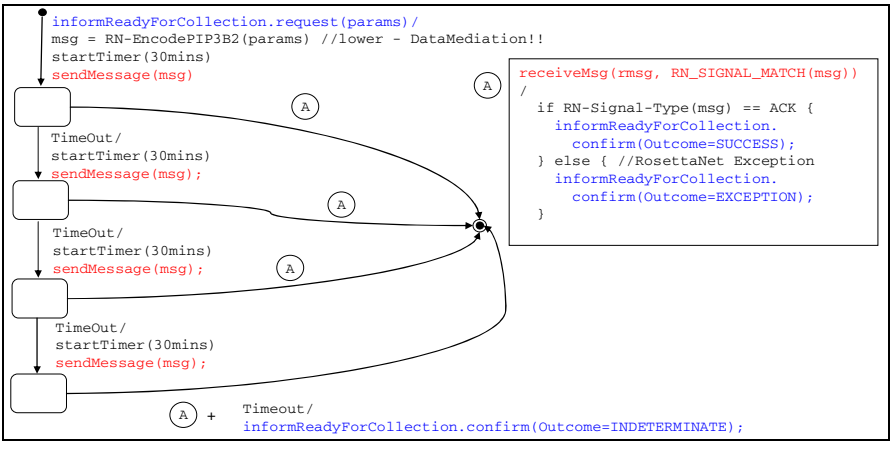

Fig. 8. Concrete RossettaNet protocol behaviour associated with a Logistics Coordinator initiating uttering an informReadyForCollection communicative act

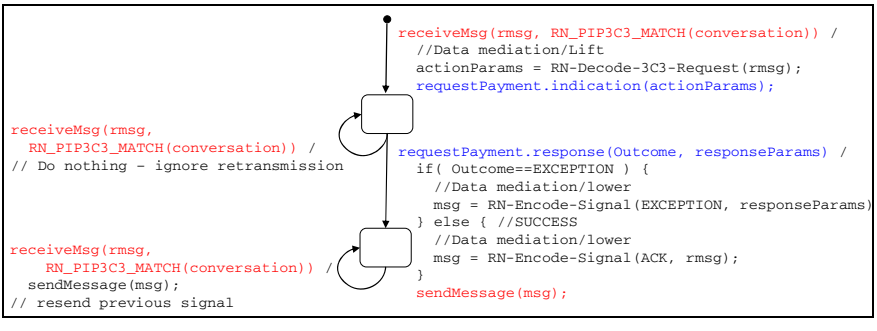

Fig. 9. Concrete RosettaNet protocol behaviour associated with a Freight Forwarder perceiving the occurrence of an informReadyForCollection communicative act

The message driven transitions shown in figures 8 and 9 involve the installation of message filters specified by the second parameter in the `receiveMsg` statements. When a message driven transition is followed the triggering message, which matches the corresponding filter is made available for computation via the variable nominated in the first parameter. A given state may have a number of message driven transitions to the same or to different successor states. Conceptually, on entry to a state with message driven transitions, the relevant filter expressions are installed to associate inbound messages with a given instance of a state transition. On transition, conceptually, all those filters are removed and on entry to a new state any filters relevant to that state are installed. In this way, messages are directed towards appropriate transitions. If multiple transitions are possible from a state, then the choice of which transition is actually taken is non-deterministic, however the message is only assigned to the nominated variable and thereby consumed if the particular message driven transition is actually taken.

5 Rich Service Description

The previous section introduced all the important elements of a rich service interface description.

- A domain ontology which structures the concepts associated with domain.
- A catalogue of roles adopted by the participants of domain interactions and the communicative acts which each role utters or perceives.
- On a per role basis, an expression of the abstract protocol which governs of the sequencing of the occurrence of the primitives modelling communicative acts.
- On a per provider per role basis, an expression of the concrete protocol associated with the utterance and perception of communicative.
- Associated with each concrete protocol description is an expression of the data mediation transformations that extract domain instance information from inbound messages and draw on domain instances in the formulation of outbound messages.

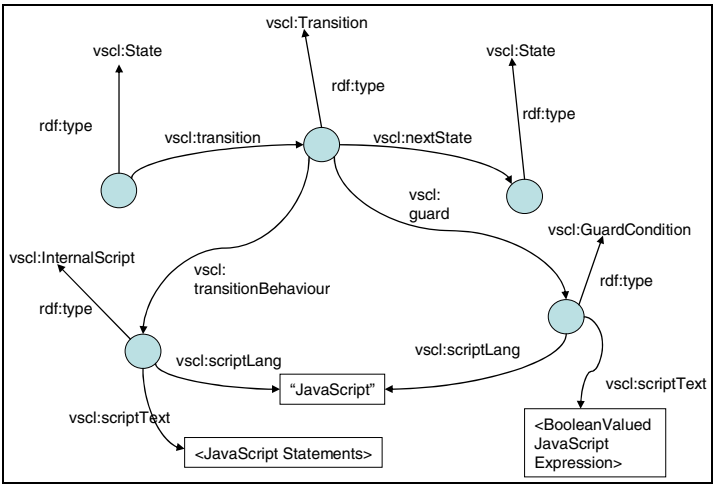

Fig. 10. A state transition in VSCL

During the course of the SWWS project we devised a "Very Simple Choreography Language" (VSCL) which embodies these elements. Abstract protocols are described as a collection of roles and each role is described in terms of the communicative acts which it initiates or perceives. The occurrence of primitives is constrained by a monitoring process. On per interface basis a concrete protocol is described in terms of the required concrete behaviour a peer role must adopt which is scoped by reference to the corresponding abstract protocol and role. Each primitive that a given role experiences is bound to the concrete behaviour required to either initiate or perceive the associated communicative act. Both the monitoring behaviours of abstract protocols and the concrete behaviours of concrete protocols are described as

processes which are expressed as finite state machines in the manner described previously.

The abstract syntax of VSCL is expressed as an OWL [4] ontology [22]. A common part abstract and concrete VSCL descriptions is the description of finite state machine processes. VSCL descriptions are written in RDF [23] using properties drawn from the VSCL ontology [22]. Figure 10 illustrates how a transition between two states is encoded in VSCL.

With respect to the example scenario presented in section 3 the VSCL description of the abstract journey leg protocol is available at [24] while the corresponding concrete protocol description provided by a freight forwarding services provider that uses the RosettaNet [15] protocol is available at [25].

The transition behaviours available in VSCL include: message driven, primitive driven, event driven and time driven transitions; sending receiving and replying to messages; raising events and primitives; forking concurrent processes (figures 5 and 6 illustrate the use of concurrency). In order to augment process behaviours with variables for storage and procedures which can perform computation over those variable we provide the ability to include scripted behaviours. In our prototype implementation we used the Mozilla open source embeddable Javascript engine, Rhino [26].

For protocol mediation, it is important that a description of a service provider's interface describes the roles and associated behaviours required of a user of that interface. The role and behaviour of the interface provider may be made explicit, but that is not strictly necessary. The assumption we make is that a service provider is economically motivated to ensure that potential service consumers are able to use the service provided. Hence, we place the onus is on the service provider to provide a rich description.

6 Related Work

OWL-S [4], WSMO [28] are two activities in the field of semantic web service description. We briefly consider the connection between these activities and the ideas discussed in this paper.

OWL-S is a natural vehicle for capturing the abstract protocols that describe the interfaces with each logistics provider. The protocol of figure 5 may be translated straightforwardly into an OWL-S composite process using sequential, iterative and concurrent process compositions. The leaves of this abstract process are described here as communicative acts, so can we identify OWL-S processes with such acts. Communicative acts certainly address the *actions* performed by agents, except that the communication is an intrinsic component of the action. This suggests that they really fit into a service-oriented, rather than a message-oriented, model. OWL-S processes are also designed to represent the actions of agents so we seem to have a good match. However, with the standard OWL-S to WSDL grounding, mapping each atomic process onto a WSDL operation can lead us astray. The problem is that there is nothing to stop a service provider mapping a pair of communicative acts onto a single operation, and hence a single atomic process. For example, it is reasonable to ground the `requestShipmentStatus` and `informShipmentStatus` in the

separate request and response messages of a single WSDL operation. The knock-on effect is that we have to model this with a single atomic-process. This decision bubbles up through the design of the interface forcing the designer to conflate two otherwise distinct acts all the way up the model. On the plus side, the current grounding is not mandated as the only possible grounding. Indeed, the concrete protocol described by the VSCL of section 5 may be thought of as a description-driven grounding that allows us to map these conceptually distinct acts to (different parts of) the same WSDL operation.

The work of the SWWS and WSMO projects are both motivated by the Web Services Modelling Framework (WSMF) [29] and there has been an on-going exchange of ideas between both projects. Our work is focussed in the mediation of interaction protocols and is most closely related to WSMO Orchestration and Choreography [30]. WSMO uses Abstract State Machines (ASM) as a formalism for describing both choreography and orchestrations. WSMO choreography is most closely aligned with our notion of an abstract protocol, whilst WSMO orchestration is most closely aligned with our notion of concrete protocols. Our work on SWWS has taken the 'easier' path abstracting communicative intent as communicative acts to which a semantic account could be given. WSMO takes the more challenging path of goal driven interaction intended to bring about desired change in the partial state of a world model.

7 Conclusions

Current practice in Web Service integration relies of a rigid plug and socket fit between the provider and consumer of a web service interface. We have demonstrated an approach that provides for description driven adaptation. Our approach relies on the provision of a rich description of the behaviour required of the user of a web service interface. Whilst this places a significant additional burden on the provider of the web service interface, it provides for massive leverage, since it vastly reduces the integration work required of a consumer of that interface. In effect we have provided a more malleable approach to the description of web service interfaces that enables interoperability and substitution were there is significant conceptual overlap between alternate interfaces.

Our approach relies on there being a shared understanding of the semantics of domain specific communicative acts and requires understanding of the semantics of individual web service operations on the part of the provider of the enriched interface description. This obviates the need for a machine readable semantic description of each web service operation, however, this results in concrete protocol descriptions that are somewhat imperative with respect to the behaviours associated with state transitions. Nevertheless, at both the abstract and concrete level, the structure of the concurrent state machines used to specify behavioural constraints is exposed and potentially available for more formal analysis with respect to the desired safety and liveliness properties of the combine abstract/concrete behaviour.

A prototype mediation component which implements the framework described in this paper has been was developed as part of the SWWS project and used as part of the logistics case study demonstrator described in [10].

Acknowledgements

The authors gratefully acknowledge the support of the EU who partially funded this work under SWWS consortium under agreement IST-2002-37134. In addition we extend particular thanks to Silvestre Losada, Oscar Corcho and Jorge Pérez Bolaño of Intelligent Software Components S.A. (ISOCO) [32] for their work implementing protocol mediation component discussed in this paper. Finally we would like to thank our colleague Chris Preist for his feedback on early drafts of this paper.

References

1. Christensen, E., Cubera, F., Meredith,G., and Weerawarana, S.: "Web Services Description Language (WSDL) 1.1", W3C Note (15 March 2001), <http://www.w3.org/ TR/ 2001/ NOTE-wsdl-20010315>
2. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., and Weerawarana, S.: "Business Process Execution Language for Web Services – Version 1.1" BEA Systems, IBM, Microsoft, SAP AG and Sibel Systems Whitepaper (5 May 2003), <ftp://www6.software.ibm.com/ software/developer/library/ws-bpel.pdf>
3. Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., and Lafon, Y.: "Web Services Choreography Description Language Version 1.0", W3C Working Draft (17 December 2004), <http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/>
4. Martin, D. et. al: "OWL-S: Semantic Markup for Web Services", W3C Member submission (November 2004) <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/>
5. Searle, J.R, "Speech Acts – An essay in the philosophy of language", Cambridge University Press, 1969
6. "FIPA Communicative Act Library Specification", FIPA, 2002. <http://www.fipa.org/ specs/fipa00037/SC00037J.pdf>
7. Esteva, M., Sierra, C.: "ISLANDER 1.0 language definition", Technicl Report of the Institut d'Investigació en Intel.ligència Artificial, IIIA-TR-02-02, 2002 <http://www.iiia.csic.es/~marc/islander-report.pdf>
8. ISO 7498/CCITT X.200, "Open Systems Interconnect Basic Reference Model", 1994 International Standards Organisation.
9. Esplugas-Cuadrado, J., Preist, C., Williams, S., "Integration of B2B Logistics Using Semantic Web Services", Lecture Notes in Computer Science, Volume 3192, Aug 2004
10. Chris Preist, Javier Esplugas-Cuadrado, Steven A. Battle, Stephan Grimm, Stuart K.Williams, Automated Business-to-Business Integration of a Logistics Supply Chain Using Semantic Web Services Technology, Lecture Notes in Computer Science, Volume 3729, Oct 2005, Page 987
11. Bochmann, G.V., "Higher-level protocols are not necessary end-to-end", ACM SIGCOMM Comput. Commun. Rev., Vol 13, No 2, April 1983.
12. Tomas, J.G., Pavon, J., and Pereda, O., "OSI service specification: SAP and CEP modelling", ACM SIGCOMM Comput. Commun. Rev., Vol 17, No 1-2, Jan-Apr 1987
13. Calvert ,L., and Lam, S. S., "Deriving a protocol converter: a top-down method", ACM SIGCOMM Comput. Commun. Rev., Vol 19, No 4, Sept. 1989.

14. Tao, Z. , Bochmann, G.V., Dssouli, R., "A formal method for synthesizing optimized protocol converters and its application to mobile data networks", Mobile Networks and Applications, Vol.2 No.3, p.259-269, Dec. 1997
15. "RosettaNet Implementation Framework: Core Specification Version 2.00.01", March 2002, <http://www.rosettanet.org>
16. ISO 9735, "Electronic data interchange for administration, commerce and transport (EDIFACT) -- Application level syntax rules", 2002, International Standards Organisation
17. Harel, D. "Statecharts: A Visual Formalism for Complex Systems", Science of Computer Programming, Vol , No 3 p. 231-274 June 1987
18. Milner, R., "Communications and Concurrency", Prentice-Hall, ISBN 0-13-115007-3, 1989.
19. Battle, S. "Round Tripping between XML and RDF", Poster ISWC 2004, <http:// iswc2004.semanticweb.org/posters/PID-BRRGVFRE-1090254811.pdf>
20. Berners-Lee, T., Fielding, R., Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax.", RFC 3986, IETF, January 2005. <http://www.ietf.org/rfc/rfc3986.txt>
21. Dean, M., Schreiber, G. (eds), "OWL Web Ontology Language Reference" W3C Recommendation, 10 Feb 2004. < http://www.w3.org/TR/2004/REC-owl-ref-20040210/>
22. VSCL Ontology <http://swws.semanticweb.org/ontologies/protocolMediation/vscl>
23. Beckett, D. (ed), "RDF/XML Syntax Specification (Revised)", W3C Recommendation,10 February 2004, < http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>
24. Sample VSCL Abstract Protocol Description <http://swws.semanticweb.org/ wp8/ logistics>
25. Sample VSCL Concrete Protocol Description<http://swws.semanticweb.org/ontologies/ choreo/rn.owl>
26. Mozilla Rhino: JavaScript for Java <http://www.mozilla.org/rhino>
27. Waldo, J., Wyant, G., Wollrath, A. and Kendal, S., "A Note on Distrubuted Computing", Sun Microsystems Laboratories, Inc TR-94-29, Nov. 1994, <http://research.sun.com/ techrep/1994/smli_tr-94-29.pdf>
28. Feier, C. (ed), "WSMO Primer", DERI Working Draft, Apr 2005, <http://www.wsmo.org/ TR/d3/d3.1/v0.2/>
29. Fensel, D., Bussler, C., "The Web Service Modeling Framework WSMF." In: Electronic Commerce Research and Applications, Vol. 1, Issue 2, Elsevier Science B.V., Summer 2002. <http://www.wsmo.org/papers/publications/wsmf.paper.pdf>
30. Roman, D., Scicluna, J., Feier, C. (eds), "Ontology Based Choreography and Orchestration of WSMO Services", DERI International, March 2005, <http:// www. wsmo.org/ TR/d14/v0.2/>
31. BEA, IBM, "BPELJ: BPEL for Java" Joint Whitepaper, March 2004, <ftp://www6. software.ibm.com/software/developer/library/ws-bpelj.pdf>
32. Intelligent Software Components <http://www.isoco.com>