Tools for Design of Composite Web Services

(Tutorial Abstract)

Richard Hull
Bell Labs
Lucent Technologies
hull@lucent.com

Jianwen Su*
Department of Computer Science
U C Santa Barbara
su@cs.ucsb.edu

1. INTRODUCTION

The web services paradigm promises to enable rich, flexible, and dynamic interoperation of highly distributed and heterogeneous web-hosted services. Substantial progress has already been made towards this goal (e.g., emerging standards such as SOAP, WSDL, BPEL) and industrial technology (e.g., IBM's WebSphere Toolkit, Sun's Open Net Environment and JiniTM Network technology, Microsoft's .Net and Novell's One Net initiatives, HP's e-speak). Several research efforts are already underway that build on or take advantage of the paradigm, including the DAML-S/OWL-S program [8, 25, 19], the Semantic eWallets project [18], ActiveXML [3], and automata-based models for web services [6, 21, 4]. But there is still a long way to go, especially given the ostensible long-term goal of enabling the automated discovery, composition, enactment, and monitoring of collections of web services working to achieve a specified objective. A fundamental question right now concerns the design and analysis of composite web services. Specifically, are existing tools for design and analysis of software systems sufficient for web services, or are new techniques needed to handle the novel aspects of the web services paradigm? This raises a variety of questions, several of which are relevant for the database research community. These include: What is the right way to model web services and their compositions? What is the right way to query them in order to support automated composition and analysis algorithms? And how can the data management aspects of composite web services be incorporated into current web services standards? This tutorial will provide the groundwork needed to address these questions, by describing emerging frameworks for studying composite services, and identifying emerging tools and techniques for both automated design and analysis of composite web services.

The tutorial will begin with an overview of the underlying goals and assumptions of the web services paradigm, from the perspectives of both emerging standards and the semantic web services community (Section 2). It then reviews key standards in the area, as these provide some of the basic building blocks to be used by the web services paradigm, and will influence the form that this paradigm eventually takes (Section 3).

*Supported in part by NSF grants IIS-0101134 and IIS-9817432.

SIGMOD 2004 June 13-18, 2004, Paris, France.

The tutorial will identify fundamental aspects for modeling of web services and their compositions. Research and standards are exploring a variety of approaches, which can be broken along several dimensions. A key dimension concerns whether the focus is on message passing (as found in WSDL and BPEL) or actions performed (as found in OWL-S and elsewhere) (Section 4). Another key dimension concerns how behaviors should be modeled, including both the behavior of individual web services and also the desired or actual behaviors of compositions of web services. Possibilities here include flowchart-based approaches (e.g., from workflow, BPEL), automata-based models, or goal-driven approaches (e.g., OWL-S). The tutorial discusses several variations that arise from the underlying operational model (Section 5). This includes issues such as whether messages passed are *synchronous* or *asynchronous*, whether *unbounded queues* are permitted, and the *topology* for compositions, e.g., peer-to-peer or hub-and-spoke.

The tutorial then examines approaches and technologies that are relevant to the problem of (manual or automated) *composition* of web services (Section 6). This includes examination of technologies that emerged before the web services paradigm came into being, such as automated synthesis as found in the verification community and automated workflow design from inter-task dependencies. It also includes in-depth discussion of recent research on composition for web services. Analysis of composite web services is then considered (Section 7). This includes again more classical results, e.g., from the workflow and verification communities, and several recent results based on various models of composite web services.

The final topic of the tutorial concerns research questions arising from the web services paradigm that are very interesting to the database community (Section 8). These focus on issues such as the development of abstractions to enable querying and manipulating web services and behavior models.

The work on web services described above is based on a variety of established fields including, e.g., automata theory [20], process algebras [26], temporal logics [12], and situation calculi [30]; review of selected results from these fields will be sprinkled through the tutorial.

The tutorial described here is focused primarily on issues of design and analysis of composite web services, primarily from the perspectives of models, languages, and algorithms. Many topics will be addressed only in passing or not at all; these include transactional properties and ontology-based reasoners.

2. GOALS OF WEB SERVICES

A good starting point for understanding the web services paradigm is to consider the stated goals, as found in the literature and the standards communities. The basic motivation of standards such

as SOAP and WSDL is to allow a high degree of flexibility in combining web services to create more complex ones, often in a dynamic fashion. The current dream behind UDDI is to enable both manual and automated discovery of web services, and to facilitate the construction of composite web services. Building on these, the BPEL standard provides the basis for manually specifying composite web services using a procedural language that coordinates the activities of other web services.

Much more ambitious goals are espoused by the OWL-S coalition [8] (formally known as DAML-S), and more broadly the semantic web services community (e.g., [9]). These goals are to provide machine-readable descriptions of web services, which will enable automated discovery, negotiation with, composition, enactment, and monitoring of web services.

A kind of middle ground is also emerging, which provides abstract "signatures" of web services that are richer than WSDL but retain a declarative flavor. Most popular here is the use of automata-based descriptions of permitted sequencing patterns of the web services, with a focus on either activities performed [4] or messages passed [21].

3. WEB SERVICE STANDARD STACK

The underlying structure for the web services paradigm will most likely be guided by already established standards and practices. Some of the current standards are illustrated by the layered structure shown in Figure 1. Briefly, web services interact by passing XML data, with types specified using XML Schema. SOAP can be used as the communication protocol, and the i/o signatures for web services are given by WSDL. All of these can be defined before binding web services to each other. Behavioral descriptions of web services can be defined using higher level standards such as BPEL, WSCI, BPML, DAML-S, etc.

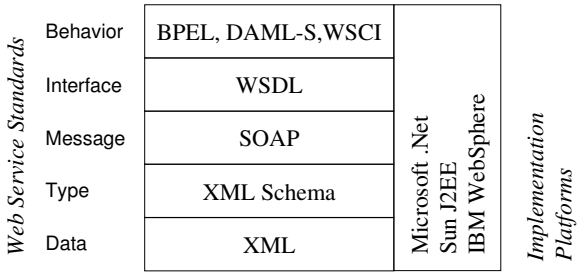

Figure 1: Web Service Standards Stack

Web service development based on these standards is supported by different (and competing) implementation platforms such as Microsoft's .Net, Sun's J2EE, IBM's WebSphere, etc. It should be noted that the choices of communication paradigms, e.g., including SOAP or JMS (Java Message Service), can have implications on how service compositions are formed. For example, SOAP primarily allows the sender to "pass" the message to the receiver, while JMS provides a message server that relays messages from sender to receiver on a first-come-first-serve basis.

4. BEHAVIOR MODELING

The model used to represent the behavior of individual and composite web services has fundamental implications on how they can be discovered, combined, and analyzed. It is not surprising that several paradigms for behavioral modeling are used in current standards and research explorations on web services composition.

In broad terms, the behavior of a service describes the changes of its "states". Depending on specific research topics, a *state* can be (a) the actual internal execution state, (b) only a part of the state of relevance to the parties connected with the service, or (c) the state of the "external world". Furthermore, different models rely on different kinds of "actions" to change state; these might be (i) messages, (ii) activities, and (iii) events.

The WSDL standard focuses heavily on passing messages between web services. This leads naturally to behavioral models that use messages as the "actions", and use (abstract) internal states for individual web services as the states that messages transition between [6]. This perspective is closely related to work on process algebras [26] and in the verification community [7], all of which study distributed automata with message passing of one form or another. It can also support investigations based on partial information of the internal states, as typical in the verification community.

The workflow community has traditionally focused on activity-based models. These represent a process by combining activities with essentially some forms of control flow. The typical formalisms in workflow community are flowcharts, Petri nets, and finite state machines or state charts.

The semantic web services community also favors an activity-based perspective. Much of that work assumes that atomic services perform activities, which have the effect of changing the state of an "external world". A situation calculus [30] is typically used to provide formal underpinnings. The emerging PSL standard [32] has also been advanced for this purpose [19]. These approaches permit the use of logic-based axiomatizations and reasoning about how composite web services affect their "external" world, and thereby permit the use of goals-based planning algorithms for automated construction of compositions.

Event-based formalisms have been used primarily in the context of workflows [31]. An event can be viewed as an abstract version of an activity. Event-based models allow declarative, logic based semantics and provide an alternative to analysis of workflow specifications [10].

In general, we can define a transition as a (single) change of states possibly with input and output. On every input (of the appropriate type), a transition produces an output of the defined type.

The correspondences between the three formalisms (message-based, activities-based, and event-based) have not been explored in depth. Activity-based models are simpler than message-based models and it is easy to group related behaviors into "modules". Message-based models allow processes to remain very much independent in their control, and interact only when it is necessary. Clearly, loose coupling is a better framework for web services and such features of message-based models are better suited for integration and composition of web services. Activity- and event-based models seem easier to associate semantics, while message-based models focus more on the mechanisms for composition.

5. VARIATIONS IN BEHAVIOR MODELS

Having identified the the major approaches to modeling web service behavior, it is informative now to drill more deeply into variations that arise within those approaches.

One fundamental parameter is the communication protocol. Under the synchronous communication protocol, the sender of a message waits for a reply after sending the message. The asynchronous protocols allow the sender to proceed with its computation; they can be further divided into two categories: the ones with message queues and the ones without. In queued asynchronous case, a message queue can be either at the receiver's side or at the sender's side (conceptually). Non-queued asynchronous protocol is useful in cases when messages are expected to be consumed quickly. Citations [6, 16] provide a formal framework for investigating the im-

plications of permitting queues with bounded or unbounded lengths.

The process of producing a web service composition can be generally viewed as starting with some global properties or dependencies that must be satisfied. Given those, design of composite web services is to orchestrate individual web services so that collectively they satisfy the global dependencies. Depending on whether and when the knowledge of the global dependencies is known, there are two different approaches to composing web services. In a mediated approach, all global dependencies are known to at least one service (called the mediator) prior to the execution, while in a peer-to-peer approach, every individual service knows only a subset of but not the full global dependencies.

Most workflow systems (e.g., Flowmark) are designed using the mediated approach and the synchronous communication protocol. The BPEL language together with WSDL and SOAP can be used to "program" mediators in the way similar to a high level programming language, although in theory BPEL also allows peer-to-peer type of composition over asynchronous messages (without queues). The Sun's J2EE package has an option of asynchronous communication with queues using JMS.

The notion of hierarchical composition, as used in some planning algorithms, will be important in the manual or automated creation of service compositions. Reference [24] explores the use of limited hierarchical composition in a framework based on "generic" compositions and user customizations.

6. AUTOMATED DESIGN

We now turn to approaches and techniques for creating composite web services. Pragmatic approaches, based on creating mediators (e.g., using BPEL) will be discussed, but the focus is on automated approaches.

One approach we will examine is the automated workflow design. In [33], the global dependencies are given as a tree, with "optional" and "choices" on some dependencies, resembling the event algebra [31]. An algorithm was given to map to a Petri net that generates the root of the tree without violating the dependencies. In a simpler model, [23] starts from a pair of pre- and post-conditions and assembles the workflow by selecting tasks from a library.

The synthesis problem for finite state specifications has been studied intensely within the automata theory and verification community [5, 1]. Consider synthesis of a collection of finite automata interacting via bounded queues. The synthesis problem has a variant for open and closed systems. In the closed case, a "folkloric" result is that synthesis from a temporal logic formula can be decided by linear reduction to the satisfiability test for the logic. Hence, it can be done in PSPACE for LTL and in PTIME for ω-regular sets represented explicitly by an automaton. The open case is undecidable [29] in general, but decidable when services are connected in a linear topology [22].

We will discuss results specific to the web services context. Recent work [4] has developed an approach to automated composition of web services, based on a model involving activity-focused finite state automata. One input to this approach is a set of descriptions of "atomic" web services, each given as an automaton. (Think of these as residing in a UDDI++ repository.) The second input is a desired global behavior, also specified as an automaton, that describes the possible sequences of activities. The output is a subset of the atomic web services, and a *delegator* (a specialized kind of mediator) that will coordinate the activities of those services, through a form of delegation. Finding a delegator, if it exists, can be done in EXPTIME.

Another approach to automated composition has been developed in the context of OWL-S [8]. The basic question in that work is whether a given collection of atomic services can be combined, using the OWL-S constructors, to form a composite service that accomplishes a stated goal. The approach taken is to encode the underlying situation calculus world view, the desired goal, the individual services (or more specifically, their pre-conditions and effects), and the OWL-S constructors into a Petri net model. This reduces the problem of composability to the problem of reachability in the Petri net.

In a third approach [6, 16], the desired global behavior is a *conversation* (a family of permitted message sequences) specified as a finite state automaton. Under certain conditions, the automaton can be "projected" to build (abstract) web services, which when combined will realize the desired conversation.

7. ANALYSIS AND VERIFICATION

The need for analysis is particularly acute for composite services, especially if they are to be created from pre-existing services using automatic algorithms. The ultimate goal is to be able to statically verify properties (e.g., in temporal logic) for composite services. We will discuss some of the known results on this topic.

Based on workflows represented in concurrent transaction logic, [10] studied the problem of whether a workflow specification satisfies some given constraints similar to the Event Algebra of [31]. Algorithms were given for such analysis. An alternative approach is developed in [33], which maps conventional workflows to Petri nets, and then applies standard results to analyze properties such as termination and reachability for workflows. Similar results have been obtained for OWL-S compositions [27].

There are two recent projects that use model checking techniques to verify BPEL composite web services. In [15], mediated composite services specified in BPEL were verified against the design specified using Message Sequence Chart and Finite State Process notations, with a focus on the control flow logic. In [17], finite automata were augmented with (i) XML messages and (ii) XPath expressions as the basis for verifying temporal properties of the conversations of composite web services. The extended model makes it possible to verify designs at a more detailed level and to check properties about message content. A framework is presented where BPEL specifications of web services are translated to an intermediate representation, followed by the translation of the intermediate representation to the verification language Promela, input language of the model checker SPIN. Since the SPIN model checker is a finite-state verification tool, the tool can only achieve partial verification by fixing the sizes of the input queues in the translation. Sufficient conditions for complete verification are also obtained.

8. QUESTIONS FOR THE DATABASE RESEARCH COMMUNITY

The web services paradigm raises a vast array of questions, including many that will be of interest to, and can benefit from, the database research community. These questions fall roughly into two categories. First is the question of finding appropriate models and abstractions for representing behavioral systems, which are suitable to the web services paradigm, and can support efficient querying and manipulation as needed by web service composition and analysis algorithms. For example, we expect that UDDI descriptions of services will expand to include automata-based specifications and/or OWL-S based specifications. In either case automatic discovery and composition algorithms will need to be able to query the UDDI++ repository. More broadly, to what extent can the problem of automated composition be re-cast to be a problem in writing and answering one or several queries against behavioral

descriptions of services? Another aspect concerns the application of XML constraint-checking techniques to perform compile-time or run-time checking of web service specifications (e.g., in WSDL and BPEL, or in emerging behavioral specification languages).

A second category of questions is how to bring data manipulation more clearly into the web services paradigm and their associated standards. The standards and most research at present are focused primarily on process model and i/o signatures, but not on the data flow and the manipulation of the data as it passes through this flow. One exception to this trend is recent work on XL [14], that blends XML query processing and web service composition constructs, and on XButler [28, 13], which essentially incorporates a WSDL stack into the Galax XQuery engine. Is there value in associating integrity constraints with web service i/o signatures? What is an appropriate way to model the data transformations occurring in a web service, which will enable reasoning about the behavior of data being passed or written to databases by a composite web service? Are there specialized models of web service composition that will be more suitable for applications that are targeted primarily at data processing? (A starting point here might be [2, 11].) Finally, it is useful to examine approaches such as ActiveXML [3], which use the web services paradigm to create richer data manipulation capabilities, such as distributed data access and query processing.

9. REFERENCES

[1] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *Proc. Int. Colloq. on Automata, Languages and Programming*, 1989.

[2] S. Abiteboul, V. Vianu, B. Fordham, and Y. Yesha. Relational transducers for electronic commerce. In *Proc. ACM Symp. on Principles of Database Systems*, 1998.

[3] S. Abitegoul, A. Bonifati, G. Cobena, I. Manolescu, and T. Milo. Dynamic XML documents with distribution and replication. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2003.

[4] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In *Proc. 1st Int. Conf. on Service Oriented Computing (ICSOC)*, *LNCS, Vol. 2910*, pages 43–58, 2003.

[5] J. Buchi and L. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[6] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: A new approach to design and analysis of e-service composition. In *Proc. Int. World Wide Web Conf.*, 2003.

[7] E.M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 2000.

[8] OWL Services Coalition. OWL-S: Semantic markup for web services, November 2003.

[9] SWSL Committee. Semantic web services language requirements (draft). `http://www.daml.org/services/swsl/requirements/swsl-requirements%.shtml`.

[10] H. Davulcu, M. Kifer, C. R. Ramakrishnan, and I. V. Ramakrishnan. Logic based modeling and analysis of workflows. In *Proc. ACM Symp. on Principles of Database Systems*, pages 25–33, 1998.

[11] A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web services. In *Proc. ACM Symp. on Principles of Database Systems*, 2004.

[12] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, volume B, Chapter 7, pages 995–1072. North Holland, 1990.

[13] M. Fernandez, R. Hull, N. Onose, and J. Simeon. Yoo-Hoo! Building a presence service with XQuery and WSDL. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2004.

[14] D. Florescu, A. Grünhagen, and D. Kossmann. XL: An XML programming language for web service specification and composition. In *Proc. Int. World Wide Web Conf.*, 2002.

[15] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based verification of web service compositions. In *Proc. the 18th IEEE Int. Conf. on Automated Software Engineering Conference (ASE 2003)*, 2003.

[16] X. Fu, T. Bultan, and J. Su. Conversation protocols: A formalism for specification and verification of reactive electronic services. In *Proc. Int. Conf. on Implementation and Application of Automata (CIAA)*, 2003.

[17] X. Fu, T. Bultan, and J. Su. Analysis of interacting BPEL web services. In *Proc. Int. World Wide Web Conf.*, 2004.

[18] F. Gandon and N. Sadeh. A semantic eWallet to reconcile privacy and context awareness. In *Proc. 2nd Int. Semantic Web Conf. (ISWC)*, Florida, October 2003.

[19] M. Grüninger. Applications of PSL to semantic web services. In *Proc. of SWDB'03, 1st Int. Workshop on Semantic Web and Databases*, 2003.

[20] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 1979.

[21] R. Hull, M. Benedikt, V. Christophides, and J. Su. E-services: A look behind the curtain. In *Proc. ACM Symp. on Principles of Database Systems*, 2003.

[22] O. Kupferman and M. Y. Vardi. Synthesizing distributed systems. In *Proc. IEEE Symp. on Logic In Computer Science*, 2001.

[23] S. Lu. *Semantic Correctness of Transactions and Workflows*. PhD thesis, SUNY at Stony Brook, 2002.

[24] S. McIlraith and T. Son. Adapting Golog for composition of semantic web services. In *Proc. the 8th Int. Conf. on Knowledge Representation and Reasoning (KR2002)*, 2002.

[25] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. In *IEEE Intelligent Systems*, March/April 2001.

[26] R. Milner. *Communicating and Mobile Systems: The π-calculus*. Cambridge University Press, 1999.

[27] S. Narayanan and S. McIlraith. Simulation, verification and automated composition of web services. In *Proc. Int. World Wide Web Conf.*, 2002.

[28] N. Onose and J. Simeon. XQuery at your web service. In *Proc. Int. World Wide Web Conf.*, 2004.

[29] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proc. IEEE Symp. on Foundations of Computer Science*, 1990.

[30] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge, MA, 2001.

[31] M. Singh. Semantical considerations on workflows: An algebra for intertask dependencies. In *Proc. Workshop on Database Programming Languages (DBPL)*, 1995.

[32] PSL Standards Group. `http://ats.nist.gov/psl/`.

[33] W. M. P. van der Aalst. On the automatic generation of workflow processes based on product structures. *Computer in Industry*, 39(2):97–111, 1999.