Planning with Workflows - An Emerging Paradigm for Web Service Composition

Biplav Srivastava
IBM India Research Laboratory
Block 1, IIT, New Delhi 110016, India
sbiplav@in.ibm.com

Jana Koehler
IBM Zurich Research Laboratory
CH-8803 Rüschlikon, Switzerland
koe@zurich.ibm.com

Abstract

In a previous work, we had analyzed the gaps in the prevalent approaches (i.e., Semantic Web Services and WSDL-described Web Services) for the problems of modeling, composing, executing, and verifying Web services, and derived challenges for the AI planning community. The challenges were in representation of complex actions, handling of richly typed messages, dynamic object creation and specification of multi-partner interactions. An important question that constantly arose was how the goals for automatic composition would be derived. In this paper, we revisit this issue in the light of new trends in software engineering towards Model Driven Architecture and early deployment of Web service composition solutions. We argue that Web services composition can not be seen as a one-shot plan synthesis problem defined with explicit goals but rather as a continual process of manipulating complex workflows, which requires to solve synthesis, execution, optimization, and maintenance problems as goals get incrementally refined. We then identify additional issues that become important in applying planning techniques.

Introduction

Web services have received much interest in academia and industry due to their enormous potential in facilitating seamless business-to-business or enterprise application integration. The business world has defined Web Services with the WSDL specification (Christensen *et al.* 2001), their composition into flows with the BPEL4WS specification (Curbera & others 2002), and their invocation with the SOAP protocol (Box *et al.* 2000). The Semantic Web community has propounded the DAML-S specification (Ankolenkar & others 2002) where semantic annotations describing web resources are explicitly declared with terms, which are precisely defined in ontologies including behavioral details of their preconditions and effects. For the composition of Web services, the Semantic Web community draws on the goal-oriented inferencing techniques from planning.

In a previous work (Srivastava & Koehler 2003), we had analyzed the prevalent approaches (i.e., Semantic Web Services and WSDL-described Web Services) for the problems of modeling, composing, executing, and verifying Web services. We found that the challenges for the AI planning community in applying their techniques are in the representation of complex actions, the handling of richly typed messages, the dynamic creation of objects, and the specification of multi-partner interactions. An important practical question that constantly arises is how the goals for automatic composition can be derived. So far, most approaches see composition as a one-shot plan synthesis problem defined with explicit goals. However, in many practical problems where Web service composition is needed (most of these are centered around business process and/or enterprise application integration), no explicit goals are given. We are observing that the starting point for Web service compositions are, quite often, abstract workflows, which are derived from business models. We will discuss example scenarios for this and derive what planning should deliver in the context of these scenarios. In particular, we dicuss that workflow expansion and feasibility testing, flow decomposition and composition as well as selecting the best web service instances for efficiency (optimization) are required. Composition itself must be a continual process that involves synthesizing, executing, optimizing and maintaining complex workflows as their structure gets incrementally refined.

The paper is organized as follows. We start with a brief summary of our previous observations for WSDL-described Web Services and Semantic Web Services. Next, we discuss *Model-Driven Architecture* (MDA) as the dominant paradigm for business process and enterprise application integration and what it means for composing web and grid services. We then describe the nature of planning for this application and identify new research topics in plan storage and retrieval, plan analysis, and continual optimization.

Background

The literature on composition of both WSDL-described (Staab & others 2003) and Semantic Web services (McIlraith, Son, & Zeng 2001) is quite comprehensive. [1] Much of it deals with resolving discrepancies in the description of Web services, the syntax and semantics of their composition and how they could be executed. Planning is being explored for automatic Web services composition [2] (McDermott 2002; Blythe & others 2003; Srivastava 2002) and

[1] See the survey at http://www.public.asu.edu/ jfan4/wssurvey.htm

[2] See papers of the ICAPS 2003 Planning for Web Services workshop at http://www.isi.edu/info-agents/workshops/icaps2003-p4ws/program.html.

many areas apart from classical planning could be relevant: distributed planning (DesJardins *et al.* 1999), planning as model checking (Giunchiglia & Traverso 1999), planning with extended goals and actions (Dal-Lago, Pistore, & Traverso 2002), and HTN planning (Erol, Hendler, & Nau 1994).

It is worth while to highlight an important characteristic about the Web service composition problem that is often overlooked. When one refers to Web services, one can refer to either the abstract web service type or to one (or more) of the several instances of a particular Web service. In the AI literature, the Web service composition problem is presented as the problem of synthesizing the complex Web service type and the scenarios are too small to have multiple Web service instances for a given type. The resulting composition is a plan - a simplified form of a workflow[3]. However, in practice, there can be a choice among many Web service instances (e.g., due to competing service providers) and the instantiated composite service will be published, deployed and made available to the clients. Since a deployed composite Web service is a long running process, it is crucial that the Web service instances are chosen carefully for performance. Industry has been primarily concerned with the instantiation/performance of the composite web service because Web services are synthesized and deployed less frequently than they are accessed. But if the benefit of running a deployed composite service reduces substantially compared to other available alternatives over time, new compositions will be tried.

We describe a realistic but simple Web service composition scenario and highlight the challenges of planning with WSDL-described and Semantic Web Services.

Example

Consider a scenario with three **types** of Web services: there is an *AddressBookService* which can return the address of a person given her name, a *DirectionService* which can return the driving directions between two input addresses, and a *GPSDirectionService* which can return the driving directions between the locations of two people given their names (see also Figure 1). The available Web service **instances** along with their invocation cost are shown in Table 1. The problem is to implement a composite Web service that can provide the driving directions between the location of any pair of persons. In practice, many metrics for a web service instance are collected, e.g., like the response time and throughput, see a detailed description in (Nanda & Karnik 2003). Any web services composition algorithm has to not only find a feasible plan (containing the relevant Web service types), but also decide how to optimize performance by invoking the best Web service instances for each type.

In Figure 2, two plans for composing the available Web service types are shown to get the driving directions. The plans can be instantiated with specific Web services listed in Table 1. In Plan1, the two *AddressBookService* services can be instantiated to the same or different Web service instances. If the runtime cost of the composite Web service has to be less than 100 units, only Plan1 with the two *AddressBookService* instantiated to AD1, AD2 or AD3 and *DirectionService* to DD1 can satisfy the constraint (total cost 75 units). Note that over time, the plans themselves may not change but the invocation cost estimate of the Web service instances can change and that can make another composition (different plan, instantiation or both) the better choice.

[3]See http://tmitwww.tm.tue.nl/research/patterns/.

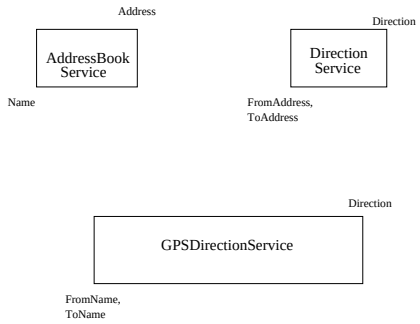

Figure 1: The input and output descriptions of the example Web services.

Service Type	Service Instances
AddressBookService	AD1 (25), AD2 (25), AD3 (25), AD4 (50)
DirectionService	DD1 (25), DD2 (80)
GPSDirectionService	GPS1 (200), GPS2 (200)

Table 1: A simple example listing the Web service types and available instances for each type. The cost of invoking each Web service instance is given in brackets.

WSDL-Described Web Service

The WSDL-described Web services specifications are primarily syntactical: the Web service interface resembles a remote procedure call and the interaction protocol is manually written. WSDL gives only the functional (input and output)

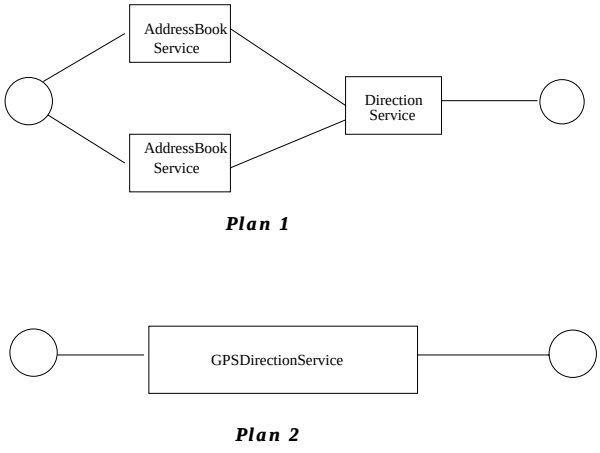

Figure 2: Two different plans for the composite Web service.

specification of Web services, not the process-centric specification of context (precondition and postconditions) to the overall computation being performed. As a very simple example, consider a Web service that manipulates two numerical expressions. The result type of the manipulation can be specified, but not the precedence rules, which determine the expression's correctness - an operator O_i is only appropriate if no operator O_j of higher precedence is pending evaluation in its immediate vicinity in any sub-expression involving O_i.

Due to the restrictiveness of the representation, most work on the automatic composition of WSDL-described Web services has focused on how to combine *instances* of Web services to serve a well specified request (output type). For example, a BPEL4WS specification for a composite service to provide driving directions given two names can be defined as shown below. Note that it identifies the exact web service instances (AD1, AD2, DD1) to use for the invocation.

```
...
<sequence>
   <receive name="receiveRequest" .../>
   <invoke name="invokeAD1"
     partnerLink="callTo"
     portType="AddrBook-PT"
     operation=".."
     inputVariable=".."
     outputVariable=".."/>
   ...
   <invoke name="invokeAD2" .../>
   <invoke name="invokeDD1" .../>
   <reply name="sendReply" .../>
</sequence>
...
```

As long as such BPEL4WS specifications are linear sequences of atomic service invocations, explicit goals are given, and each service is annotated with its predefined precondition and goals, an AI planner can be easily used to generate a plan. However, as we will show next, realistic scenarios may look quite differently from this ideal setting.

The normal situation today is that these BPEL4WS specifications are manually composed by IT experts using, for example, graphical editors to avoid writing the XML code by hand. Similarly, UML can be used to create BPEL4WS (Amsden *et al.* 2003). The BPEL4WS approach looks at composite services mainly from the runtime perspective of functions, data and control flow. The only information available for reasoning about a service are inputs, outputs and exception handlers. Schemas define and restrict the format of data and define their relationships. The only flexibility is in dynamically selecting a Web service binding details or to execute specific branches in the specified workflow.

Semantic Web Services

The Semantic Web (Berners-Lee, Hendler, & Lassila 2001) views the World Wide Web as a globally linked database where web pages are marked with semantic annotations. At the core, semantic annotations are assertions about web resources and their properties (example, "A is subclass of B") expressed in the Resource Description Format (RDF) (RDF 1999). The Semantic Web Services are dynamic web resources represented in the DAML-S representation (Ankolenkar & others 2002). The Web service here is described by *ServiceProfile* to advertise the service, *ServiceModel* to express its behavior and *ServiceGrounding* to find its implementations. The preconditions and postconditions of the Web service are explicitly declared in *ServiceModel* using terms from pre-agreed ontologies.

```
<rdf:RDF ...>
 <daml:Ontology>
 ...
 </daml:Ontology>
 <!-- Atomic Process : AddressBookService --
>
 <daml:Class rdf:ID="AddressBookService">
  <daml:subClassOf
   rdf:resource=".../Process.daml#AtomicProcess"/>
 </daml:Class>
 <!-- Inputs and Outputs -->
 <daml:Property rdf:ID="Name">
  <daml:subPropertyOf
   rdf:resource=".../Process.daml#input" />
  <daml:domain ... />
  <daml:range rdf:resource="...#Name" />
 </daml:Property>
 <daml:Property rdf:ID="Address">
  <daml:subPropertyOf
   rdf:resource=".../Process.daml#output" />
  <daml:domain ... />
  <daml:range rdf:resource="...#Address" />
 </daml:Property>
</rdf:Property>
```

In the above DAML-S specification, a Web service of type *AddressBookService* is described as an atomic process. The properties are defined by extending the DAML-S Process ontology (Process.daml) extensively. The Web service instances can be described by extending the properties of the corresponding Web service type.

The specification of the composite route finding service is given below which invokes concrete instances of the atomic process types.

```
<daml:Class rdf:ID="Route_Process">
 <daml:subClassOf rdf:resource=
     ".../Process.daml#CompositeProcess"/>
  <daml:subClassOf>
   <daml:Restriction>
   ...
     <process:listOfInstancesOf ...>
       <daml:Class rdf:about="#AD1Service"/>
       <daml:Class rdf:about="#AD2Service"/>
       <daml:Class rdf:about="#DD1Service"/>
     </process:listOfInstancesOf>
   ...
   </daml:Restriction>
  </daml:subClassOf>
</daml:Class>
```

The planning approaches in the Semantic Web are focused on the process-centric description of services as actions that are applicable in states. State transitions are defined based on the preconditions and postconditions of actions that mimic the intended behavior of the Web services.

Executing an action (a service) triggers a transition that leads to a new state where the effects of the action are valid. This approach relies on the representation of state, actions, goals, events and optionally, an ontology of standard terms. The plan can be adapted both offline and online. There is more flexibility in terms of considering different choices of services (plans) based on goals, but the goals are explicitly given.

Discussion

The key hurdles in applying AI planning technology to the service composition problem are the following:

- Complex actions. Action specifications traditionally consist of preconditions and effects. However, a Web service modeled as an action can have complex control structures in its specification involving loops and nondeterministic choices. Furthermore, the assumption of explicit preconditions and effects seems to be rather unrealistic.
- Rich data types. The "objects" manipulated by the Web services (actions) are typed messages which may contain identifiable parts that can be arbitrarily complex descriptions. Planning methods have traditionally worked with simple types.
- Dynamic objects. Processes using Web services can lead to new object being created at runtime. In planning, all objects are assumed to be known in the initial state.
- The execution of a workflow very often needs coordination among the partners, which are involved in an interaction. This is in contrast to the traditional on-off invocation by the executor assumed in AI planning. Real-time interleavings between a planning and an execution unit have not been studied very widely so far.

Trends Influencing Web Services

Web services, like other software building blocks, are affected by the changing practices in software engineering. In this section, we discuss two trends that will have an important influence on web services and their composition:

- Model-Driven Architecture and Model-Driven Development
- Continual Optimization of Processes and on-Demand Composition of Workflows

Model-Driven Architectures

Model-driven architectures (MDA) have been proposed by the OMG (The-Object-Management-Group 2003) to reinforce the use of an enterprise architecture strategy and to enhance the efficiency of software development. The key to MDA lies in three factors:

- the adoption of precise and complete semantic models to specify the structure of a system and its behavior,
- the usage of data representation interchange standards,
- the separation of business or application logic from the underlying platform technology.

Each of these key factors poses a challenge of its own, but the model representation seems to be the most critical among them. Models can be specified from different views, from the business analyst's or the IT architect's view, and they can be represented at different levels of abstraction. MDA distinguishes between platform-independent (PIM) and platform-specific (PSM) models, but we can expect that many different PIM and PSM models will be used to describe the different aspects of a system. Representing a model of a system (be it a PIM or a PSM) is only one side of the coin. In order to be useful, a model must be further analyzable by algorithmic methods. For example, the fundamental properties of the behavior of a system should be verifiable in a model, and different models of the same system should be linked in a way that relevant changes in one model can be propagated to the other models to keep the model set consistent.

The OMG has defined and is defining a set of standards that will help to implement model-driven architectures and model-driven development. At the core of these standards are the Unified Modeling language (UML), the Meta-Object Facility (MOF)—a unique object-oriented representation for models, the XML Meta-Data Interchange (XMI) to persist MOF models in an XML dialect, and the Common Warehouse Metamodel (CWM), which standardizes data formats.

Besides the OMG standards, a number of competing modeling approaches are under development. A particular emphasis is on modeling distributed enterprises and their processes. Graphical rendering tools such as Microsoft Visio or Powerpoint have been very popular so far. However, recently the focus moved from standalone models to business-process models, which are suitable to automate the transition from a business process model to the executable runtime of a system. To generate code means to start from a much more structured representation with a (hopefully) well-defined semantics. Two prominent representatives are ARIS (Scheer *et al.* 2002) and Holosofx (Deborin & others 2002),—just to name a few. The latter is forming the foundation of IBM's recent Websphere Business Modeler product, which is part of an integrated solution that supports modeling, code generation and monitoring of the deployed code. Interestingly, these modeling approaches come all from a Petri-net background. They focus on data-and control flow, but state information is never made explicit. Trying to apply AI planning to Web service composition therefore means to face the clash of two different worlds: Petri nets vs. State machines.

In the following, we briefly illustrate this problem by reviewing two major scenarios in which Web service composition occurs:

- Model-driven Business Integration, and
- Service Composition from Conversation Specifications.

Model-driven Business Integration

The automation of business processes and the integration of processes and applications is a current major driver for IT investments. Workflow systems on the basis of Message Oriented Middleware (MOM) products are often used to achieve this goal. A workflow defines a set of activities and their control- and data flow. Even in the simplest

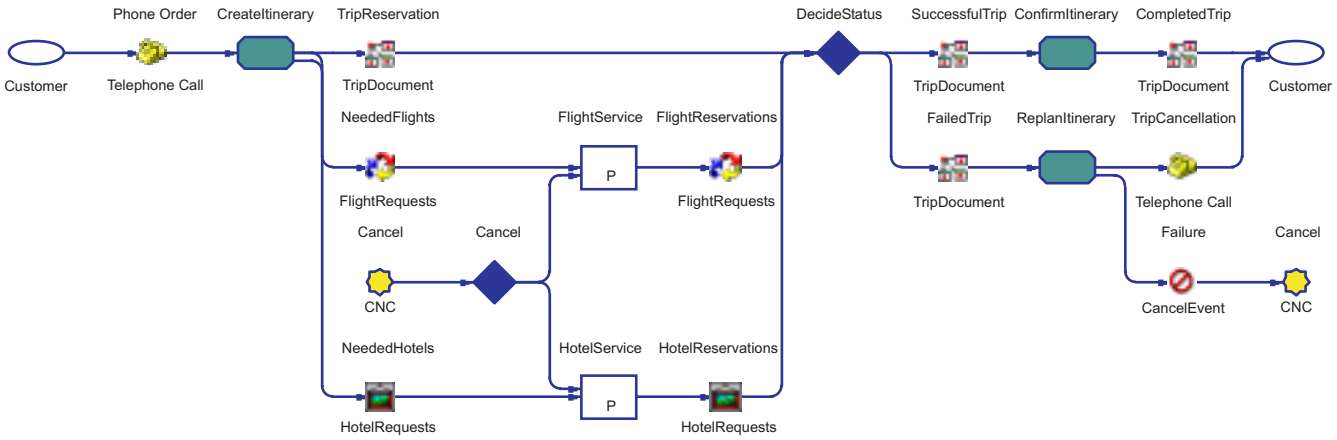

Figure 3: ADF representation of a trip reservation business process.

setting, a workflow usually contains branching and iterations in the control flow. In contrast to this, in AI planning, the notion of a plan has traditionally been simpler–a plan is a sequence of steps where each step can have a set of parallel action instances. The generation of plans involving more complex control flows has only been considered by a few approaches (Dal-Lago, Pistore, & Traverso 2002; Stephan & Biundo 1996). Interestingly, the standardization efforts in AI planning defined a unique domain representation format, but not a standardized format for plans. Adopting the upcoming BPEL4WS standard could fill this gap and would also nicely align plans with workflow standards.

In Model-Driven Business Integration, workflows ("plans") are assumed to be generated from business process models. Figure 3 shows the specification of a plan in the ADF formalism by Holosofx.

In this example representation, the customer is an external entity (depicted with a white oval) initiating the trip-handling process. Three main elements are used to capture the structure of the process: (1) Grey octagons depict the elementary process steps, which are named as *CreateItinerary*, *ConfirmItinerary*, and *ReplanItinerary*. (2) White boxes containing a "P" letter depict sub-processes and refer to the *FlightService* and *HotelService*. (3) Black diamonds show binary decision steps in the process. Typed information entities flow between the various process steps. In the example model, we distinguish between five different types of information entities named *TelephoneCall*, *TripDocument*, *FlightRequests*, *HotelRequests*, and *CancelEvent*. Each of them is depicted with a separate so-called Phi symbol. Below each Phi symbol, we find the type and above the symbol, we find the Phi name (the particular instance of this type). A star-shaped goto symbol allows to model cyclic processes. In our example, the Phi *Failure* of type *CancelEvent* is sent from the *ReplanItinerary* step back to the *FlightService* and *HotelService* sub-processes via the CNC goto symbol. Which of the services has to be notified is decided in the decision step receiving the Phi.

The task of the 'planning system' is to transform this representation into an executable BPEL4WS specification. The problem lies not in determining the set of actions and their execution order, but in deriving the correct structure of the BPEL4WS process, its required interactions with partners and the Web service interfaces based on the specified data flow. A particular challenge lies in transforming unstructured cyclic control flows into well-structured optimized workflow code. For simple cases, graph-based transformations can be used (Koehler *et al.* 2003), for more complicated cases, more sophisticated splitting and optimization techniques must be used, but so far there is not much need for search-based planning techniques. However, there is an upcoming need for verification due to the increasing complexity of the flows (in particular when the combination of concurrency and loops occurs).

Service Composition from Conversation Specifications

A second scenario, which seems to require more planning-like techniques is the problem of service composition from a given conversation specification. Business process models often only concentrate on specifying the process itself, but do not pay much attention to specify the process context. For a Web service-based process, it is of particular importance to specify the interactions between a process and its partners providing additional Web services. This specification can be formulated using the Web Service Conversation Language (WSCL) (Banerji & others 2002), a proposed standard submitted to the W3C, which is based on activity diagrams that describe the allowed interactions. The WSCL diagram for the trip-handling example considered above, may look like in Figure 4.

The abstract BPEL4WS specifications (also sketched as WSCL diagrams) of the three partners of this process, the customer, the flight agency, and the hotel booking service, may look like those shown in Figures 5, 6, and 7.

The task of a planner is to generate a BPEL4WS process based on this information that when executed will correctly interact with the given partner processes and implement the conversations specified in the WSCL diagram. This problems resembles much more a planning problem, because it

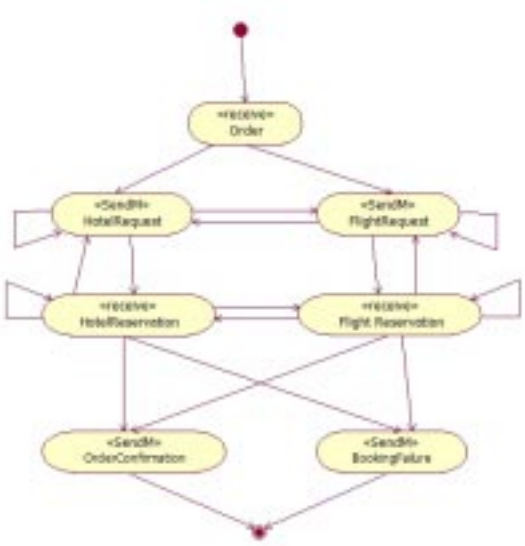

Figure 4: A WSCL Diagram for the Trip-Handling Process

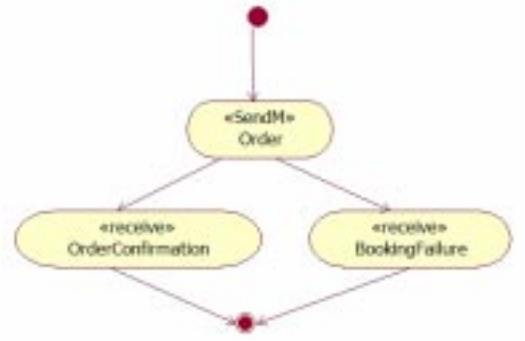

Figure 5: Conversational Behavior of the Customer

requires to select the necessary activities and their ordering. The difference is that again no explicit pre- and postconditions are given, but the conversational behavior of the various BPEL4WS activity types is known. It is also obvious that the plan must contain control structures to enable the cyclic conversations to be implemented.

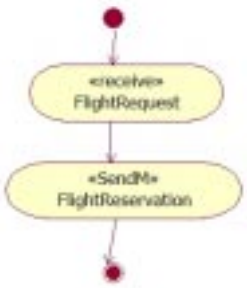

Figure 6: Conversational Behavior of the Flight Service

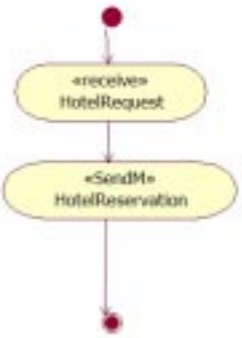

Figure 7: Conversational Behavior of the Hotel Service

Continual Optimization of Processes and on-Demand Composition on Workflows

There are two key concerns while composing Web services. Since applications are built for specific business requirements (which may not necessarily be explicitly stated), the composition of services has to be semantically compatible with the business domain in order to be useful. Another consideration is that the composed Web service should be executed efficiently by leveraging the inherent optimization opportunities, e.g., concurrency, in the continually running web service flow, cf.(Nanda, Chandra, & Sarkar 2003). These concerns were illustrated with a very simple example in Table 1.

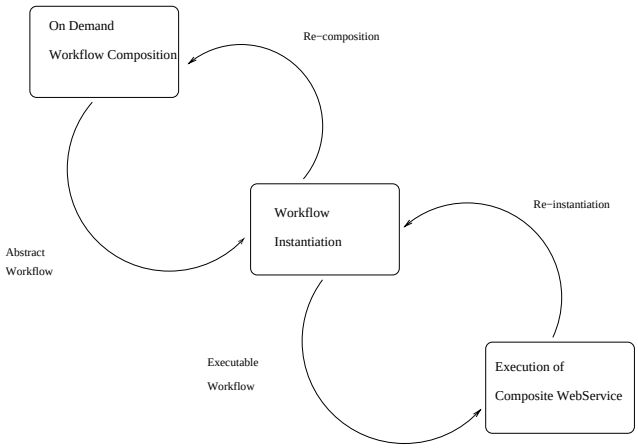

Figure 8: 2-stage View of Web Service Composition

Web service composition can be viewed as a 2-step process (see Figure 8) that addresses these concerns separately. In the first stage, the workflow representing the composition of the Web service composition may be produced on demand using planning techniques. The planning process can be implemented as a synthesis from scratch, as plan reuse involving a revision of previous cases, or as a semi-automated refinement of abstract workflows through the mixed-initative of planner and user. In the second stage, a composite process representing an instantiation of the workflow has to be pub-

lished, deployed and made available to the clients, so that it can be used just like any other atomic Web service.

However, the stages are not independent. While the composite web service is running, various runtime metric will be monitored and alternative instantiations or workflows, or both may offer better execution performance. The user may also change the requirements for the composition, which can be an additional driver for the generation of an alternative and workflow.

In planning, decoupling of causal reasoning and resource reasoning has been investigated earlier (Srivastava, Kambhampati, & Do 2001) where constraints were used to provide feedback between independent action selection and sequencing phases that iterated alternatively. This may also serve as a framework for the continual optimization of workflow compositions.

New Areas

In applying planning for Web service composition, some new areas have to be tackled.

Storage and retrieval of plans: As has been found in other applications of planning, users would want to use plans produced from previous planning episodes in guiding new compositions. The storage and retrieval of plans is related to the storage of workflows, but additional meta-data is available in planning about the objective and functionality of the plan. For example, information about how the plan was obtained (from scratch, with a user's participation or by reuse) is an important annotation for retrieving plans. In data integration, this is called the problem of *data provenance* and for plan storage, it may be termed *plan provenance*. Previous work on case retrieval for plans is relevant but there, the retrieval was performed in the context of a specific planner.

Plan Analysis and Optimization: Since a plan may be the best selection in one context and need not remain the same over time, techniques are required to analyze plans over an uncertain future (Garland & Lesh 2002). Optimization is of prime importance in web services because there can be many available Web service instances with drastically different invocation costs. Optimization has been given limited focus in planning: it was introduced with PDDL 2.1, but the metrics are rather simple. In this paper, we discussed with the help of a simple example that the continual optimization is the practical challenge even in the presence of simple objectives.

Plan Execution Monitoring: Previously, work has been done in monitoring of actions and overall plan execution. For Web services, the change is that the same plan or Web service composition will be executed continuously and the environment can evolve over time.

Conclusion

Building on our analysis of the gap between the needs of Web services composition for business applications like enterprise application integration and the current status of AI planning techniques, we argue that planning cannot be seen as a one-shot plan synthesis problem defined with explicit goals. Rather, it is a continual process of synthesizing, executing, optimizing, and maintaining complex workflows as goals get incrementally refined. We discuss example scenarios that illustrate new challenges for planning and identify additional areas that may become important for applying planning techniques.

References

Amsden, J.; Gardner, T.; Griffin, C.; Iyengar, S.; and Knapman, J. 2003. UML profile for automated business processes with a mapping to BPEL 1.0. IBM Alphaworks http://dwdemos.alphaworks.ibm.com/ wstk/ common/ wstkdoc/ services/ demos/ uml2bpel/ docs/ UMLProfileForBusinessProcesses1.0.pdf.

Ankolenkar, A., et al. 2002. DAML services. http://www.daml.org/services/.

Banerji, A., et al. 2002. WSCL: The web services conversation language. http://www.w3.org/TR/wscl10/.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The semantic web. Scientific American, May issue.

Blythe, J., et al. 2003. The role of planning in grid computing. Proc. ICAPS.

Box, D.; Ehnebuske, D.; Kakivaya, G.; Layman, A.; Mendelsohn, N.; Nielsen, H.; Thatte, S.; and Winer., D. 2000. Simple object access protocol (soap) 1.1. http://www.w3.org/TR/SOAP/.

Christensen, E.; Curbera, F.; Meredith, G.; and Weerawarana, S. 2001. The web services description language WSDL. http://www-4.ibm.com/ software/ solutions/ webservices/ resources.html.

Curbera, F., et al. 2002. Business process execution language for web services. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.

Dal-Lago, U.; Pistore, M.; and Traverso, P. 2002. Planning with a language for extended goals. In Dechter, R.; Kearns, M.; and Sutton, R., eds., *Proceedings of the 20th National Conference of the American Association for Artificial Intelligence*, 447–454. AAAI Press.

Deborin, E., et al. 2002. *Continuous Business Process Management with HOLOSOFX BPM Suite and IBM MQSeries Workflow*. IBM Redbooks.

DesJardins, M.; Durfee, E.; Ortiz, C.; and Wolverton, M. 1999. A survey of research in distributed, continual planning. *AI Magazine* 20(4):13–22.

Erol, K.; Hendler, J.; and Nau, D. 1994. UMCP: A sound and complete procedure for hierarchical task-network planning. In Hammond, K., ed., *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems*, 249–254. AAAI Press, Menlo Park.

Garland, A., and Lesh, N. 2002. Plan evaluation with incomplete action descriptions. In *Eighteenth national conference on Artificial intelligence*, 461–467. American Association for Artificial Intelligence.

Giunchiglia, F., and Traverso, P. 1999. Planning as model checking. In Biundo, S., ed., *Proceedings of the 5th European Conference on Planning*, LNAI. Springer.

Koehler, J.; Hauser, R.; Kapoor, S.; Wu, F.; and Kumaran, S. 2003. A model-driven transformation method. In *Proceedings of the 7th International IEEE Conference on Enterprise Distributed Object Computing (EDOC)*. IEEE Press.

McDermott, D. 2002. Estimated-regression planning for interactions with web services. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling*. AAAI Press, Menlo Park.

McIlraith, S.; Son, T.; and Zeng, H. 2001. Semantic web services. In IEEE Intelligent Systems (Special Issue on the Semantic Web), March/April 2001.

Nanda, M. G., and Karnik, N. 2003. Coordinating components in decentralized composite web services. In *Proceedings of the Association of Computing Machinery International Symposium on Applied Computing, Melbourne, Florida, USA*.

Nanda, M. G.; Chandra, S.; and Sarkar, V. 2003. Decentralizing composite web services. In *Proceedings of the Tenth International Workshop on Compilers for Parallel Computers (CPC), January 8-10, 2003, Amsterdam, The Netherlands.*

RDF. 1999. RDF: Resource description framework. http://www.w3.org/RDF/.

Scheer, A. W.; Abolhassan, F.; Jost, W.; and Kirchner, M. 2002. *Business Process Excellence - ARIS in Practice*. Springer.

Srivastava, B., and Koehler, J. 2003. Web service composition - current solutions and open problems. ICAPS 2003 Workshop on Planning for Web Services, Trento, Italy.

Srivastava, B.; Kambhampati, S.; and Do, M. B. 2001. Planning the project management way: Efficient planning by effective integration of causal and resource reasoning in realplan. *Artif. Intell.* 131(1-2):73–134.

Srivastava, B. 2002. Automatic web services composition using planning. In *Proceedings of 3rd Internation Conference on Knowledge-Based Computer Systems*, 467–477.

Staab, S., et al. 2003. Web services: Been there, done that? *IEEE Intelligent Systems, Jan-Feb issue.* 72–85.

Stephan, W., and Biundo, S. 1996. Deduction-based refinement planning. Proc. of AIPS-96, pages 213–220. AAAI Press, 1996.

The-Object-Management-Group. 2003. OMG model driven architecture. http://www.omg.org/mda.