Towards Interactive Composition of Semantic Web Services

Jihie Kim and Yolanda Gil
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, U.S.A.
jihie@isi.edu, gil@isi.edu

Abstract

We are developing a framework for interactive composition of services that assists users in sketching their requirements by analyzing the semantic description of the services. We are applying this framework to compose end-to-end simulations for earthquake scientists. We describe the requirements that an interactive framework poses to the representation of the services, and how the representations are exploited to support the interaction. We also describe an analysis tool that helps users create complete and correct compositions of web services.

Introduction

Composing executable workflows out of smaller components is essential in many areas, including large-scale scientific research and business related applications. A new kind of science is emerging from the integration of models developed by individual scientists and groups, to result in end-to-end scientific applications that result from the composition of those individual models. Another example is the composition of web services to create new applications out from existing software components (such as software modules or web services) given a customer's needs.

Web services are an emerging technology to describe and discover these components. Research in semantic web services provides more expressive representations that result in more powerful techniques for services composition.

This paper argues that:

- complex applications require interactions with users, who will need to formulate their goals at high levels of abstraction while the system will work out the details.
- partial workflows containing high-level descriptions of component services are needed to help users navigate the space of possible combinations of services.
- constraints shared by abstract types of components need to be checked at every step, since they result in commitments made by users as they narrow down the space of possible choices.

Most approaches to web service composition address automatically composing the services(Mcdermott 2002; Burstein *et al.* 2000; Paolucci *et al.* 2002; Sycara, Lu, & Klusch 1999). However, in many contexts users will want to drive the process, influencing the selection of components and their configuration. In addition, users may only have high-level or partial/incomplete description of the desired outcome or the initial state, so it may be hard to directly apply automatic approaches that require explicit goal representations. Business agreement and past experiences of how the components were used may also affect the development of the composition.

The goal of our work is to develop interactive tools for composing web services where users sketch a composition of services and system assists the users by providing intelligent suggestions.

Interactive approaches need to address additional challenges in composing services. First of all, users may make various types of mistakes and the system needs to help fix them. The user may forget to specify links between the steps or specify wrong links. There may be some missing steps or unnecessary steps. Also, user's input is often incomplete (such as incomplete or abstract goal descriptions) and may even be inconsistent with existing descriptions of the components.

We found such issues arising in constructing *computation pathways* in earthquake science where engineers interactively compose existing or web services in order to answer their queries. Even with a small number of services that need to be put together to generate answers, users would benefit from the assistance of intelligent tools that help them specify complete and correct pathways.

In order to help users in this context, we have developed a framework for guiding users in sketching a composition of services by exploiting a semantic description of the services. The framework is inspired by our earlier work in helping users construct process models from pre-defined components of objects and events (Kim & Gil 2001). In our previous work, we have built a tool that performs verification and validation of user entered process models by exploiting domain ontologies and event ontologies. In this work, we first take existing service descriptions and extend them with domain ontologies and task ontologies that address various task types in the domain. Our analysis tool then uses these ontologies in examining a user's solution (i.e., composition of services) and generating suggestions about how to pro-

Copyright © 2004, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

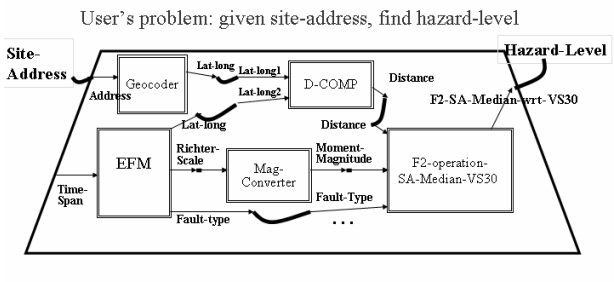

Figure 1: An example computational pathway in earthquake science domain.

ceed.

The tool we built is called CAT (Composition Analysis Tool). CAT's analysis is driven by a set of desirable properties of composed services including (1) all the expected results are achieved, (2) all the links are consistent, (3) all the input data needed are provided, and (4) all the operations are grounded (there are actual operations that can be executed). While performing these checks, CAT generates specific suggestions on how to fix the errors based on the type of the errors and the situation at hand. We show how these checks can effectively help engineers build computational pathways in earthquake science. We also show how the approach can be used in other domains when appropriate domain ontologies and task ontologies are provided. As ontologies become richer, the tool can provide more direct and focused suggestions.

This paper begins by introducing a problem that motivated us to build our framework. Then we describe how existing service definitions can be extended with domain and task ontologies. Next we present the current implementation of CAT including the kinds of checks made and the suggestions provided, and then show how the system works in the context of constructing computational pathway in earthquake science and travel planning. Finally we discuss remaining issues and future plans.

Motivation: Computational Pathway Elicitation for Earthquake Science

One of the key problems often addressed in earthquake science is to analyze the potential level of hazard at a given site. For example, engineers may want to determine the probability that some measure of earthquake shaking will be exceeded during a specified time period. Depending on the kind of structure, the engineer will be interested in looking at a particular Intensity Measure Type (IMT): PGA (Peak Ground Acceleration), PGV (Peak Ground Velocity), or SA (Spectral Acceleration). The engineer is concerned with the IMT exceeding an intensity measure level. There are simulation models available that provide an estimate of hazard at that site for that structure as a probability that the intensity measure level will be exceeded in a certain time period. They are called Intensity Measure Relationships (IMRs). An IMR can analyze the impact on that site for a given earthquake forecast, so the IMRs should be run considering Earthquake Forecast Models (EFMs) that suggest entire sequences of earthquake forecasts around the area where the site is located. Users can choose different IMRs depending on the situation at hand because each IMR is designed to take into account specific types of earth shaking phenomena. In addition, different constraints that are associated with the IMRs have to be taken into account when they are used.

To determine the hazard level given a site, we may need to put together various components, as shown in Figure 1, considering the overall task given and the constraints associated with each component. We call it a *computational pathway*. A computational pathway consists of a set of operations and a set of links that connect the operations based on their input and output parameter constraints.

In constructing a computational pathway, engineers may use a variety of strategies, including 1) top-down selection of components, starting from abstract types of models and then selecting specific ones; 2) result-based selection of components working from desired data to select models that can generate those results; 3) situation-based selection of components, working from the initial data available to select components whose constraints are consistent with those data.

Formulating complete and consistent pathways in this process is very hard for end users who don't have computer science background (Kim & Gil 2001; 2000). For example, user terms may be different from the description language that define the services, users may not know how to describe their problems, and they may make many different mistakes.

In order to build the pathway shown in Figure 1, users need a proactive help from a system that understands how the pathway is being built and generates appropriate suggestions.

Approach

Our approach complements simple WSDL (Web Service Description Language) models (Christensen *et al.* 2003) with both tasks and domain ontologies as follows:

- We cast each simulation model as a web service and describe its input and output in WSDL. A simulation model can be invoked in several ways, and each of them is mapped to different operations for its service.
- We use web services for message transport, but not for reasoning about the service. That is, the WSDL descriptions have "string" in the types assigned to message parameters.
- We use off-the-shelf domain ontologies to specify data types in the WSDL descriptions. That is, parameters in WSDL messages are mapped to terms in the domain ontology.
- We use a task ontology to describe abstract types of operations and services. We follow the approach in (Gil & Blythe 2000) to represent task types and their arguments. This representation is based on case frames (Fillmore 1968), where verbs are qualified by cases that reflect their linguistic usage.

We first show how existing service descriptions can be extended with domain ontologies. Then we present CAT's

analysis functions that are built based on this extended representation.

Representing services using domain ontologies

In order to support the kinds of interactions described above, CAT needs a semantic description of the services and their constraints on input and output parameters.

```
<!-- WSDL description of the Field-2000 Web APIs.-->
<definitions name="urn:Field_2000Query"
      targetNamespace="urn:Field_2000Query"
      xmlns:typens="urn:Field_2000Query"
       ...
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns="http://schemas.xmlsoap.org/wsdl/">
 <!-- Messages for Field-2000 Web APIs -->
 <message name="F2-operation-SA-Median-VS30-Request">
   <!-- Generic inputs -->
   <part name="model"   type="xsd:string"/>
   <part name="user"   type="xsd:string"/>
   <part name="logwsdl"  type="xsd:string"/>
   <!-- Inputs specific to the model -->
   <part name="MOMENT-MAGNITUDE" type="xsd:string"/>
   <part name="DISTANCE-JB" type="xsd:string"/>
   <part name="FAULT-TYPE-PARAMETER" type="xsd:string"/>
   <part name="VS30" type="xsd:string"/>
   ...
 </message>
 <message name="F2-operation-SA-Median-VS30-Response">
   <part name="return"  type="xsd:string"/>
 </message>
  ...
 <!-- Port for Field-2000 Web APIs -->
 <portType name="Field_2000Port">
   <operation name="F2-operation-SA-Median-VS30">
     <input message=
      "typens:F2-operation-SA-Median-VS30-Request"/>
     <output message=
      "typens:F2-operation-SA-Median-VS30-Response"/>
   </operation>
   <operation name="F2-operation-SA-Median-MAGNITUDE">
     <input message=
      "typens:F2-operation-SA-Median-Magnitude-Request"/>
     <output message=
      "typens:F2-operation-SA-Median-Magnitude-Response"/>
   </operation>
  ...
 </portType>
 <!-- Binding for Field-2000 Web APIs -->
 <binding name="Field_2000Binding"
   type="typens:Field_2000Port"> ...
 </binding>
 <!-- Endpoint for Field-2000 Web APIs -->
 <service name="Field_2000Service">
   ...
 </service>
</definitions>
```

The above shows a part of WSDL representation of an earthquake simulation service. In our work with earthquake scientists, web services have been built for existing software components. The service, called "Field_2000Service", is represented as a set of *operations* that take input data needed to run a simulation and generate output results as numbers. For example, an operation called F2-operation-SA-Median-VS30 produces median values of spectral acceleration with respect to valid VS30 value ranges. There are various other Field-2000 operations that can be chosen by users depending on the kinds of queries they have in mind, including the IMT type (SA or PGA or PGV), the probability function (Median or Probability of Exceedance or Standard-Deviation), the independent variable used to render the results (VS30 or Moment Magnitude or ...), etc. In this representation of a service, all the input and output data types are simply described as strings, so we consider it 'syntactic' in the sense that the description itself does not provide enough semantic information on what each operation needs, what it does and what it produces, which we believe are essential in supporting interactive service composition. Although some of the existing services use more complex data types, most of the service descriptions cannot directly support the kinds interactions needed for constructing computational pathways.

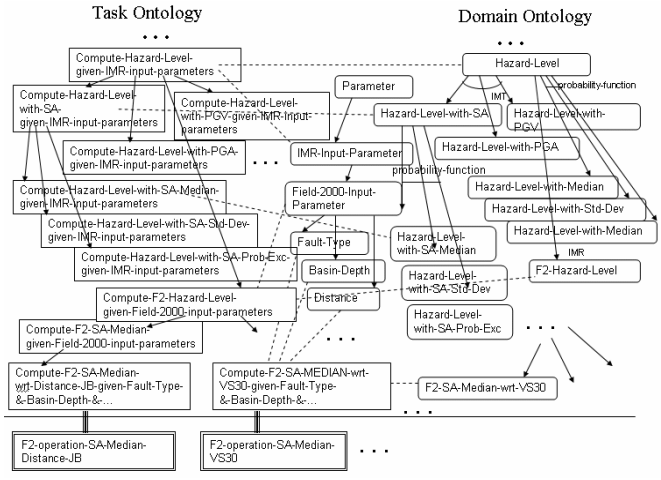

Figure 2: Task Ontology and Domain Ontology.

Our approach is to take these existing syntax level service descriptions and extend them with two types of knowledge: domain term ontology and task ontology. As shown in Figure 2, service operations can be defined using the domain terms defined in a domain ontology. That is, their input and output data types can be represented using domain objects, and their task types can be defined as mappings between input data types and output data types. For example, a task type Compute-Hazard-Level-given-IMR-Input-Parameters has IMR-Input-Parameters as the input and Hazard-Level as the output.

The current implementation uses description logic to represent and reason about these ontologies. For example, the above description of Field 2000 can be represented as a set of instances, where Field-2000-Service represents a service and F2-operation-SA-Median-VS30 is an instance representing one of the operations. Each operation is represented with a task type in the task ontology

and its input and output parameters are described using the data types defined in the domain ontology. Here F2-operation-SA-Median-VS30 is an operation that produces F2-SA-Median-wrt-VS30 given Fault-Type, Basin-Depth, etc. (Compute-F2-SA-Median-wrt-VS30-given-Fault-Type-&-Basin-Depth-&-...). As shown in Figure 2, this operation is a kind of earthquake simulation task (Compute-Hazard-Level-given-IMR-Input-Parameters).

Checking Computational Pathways with CAT

Given a computational pathway and a user task description (i.e., a set of initial input and expected results), CAT checks if (1) all the expected results are produced, (2) all the links are consistent, (3) all the input data needed are provided, and (4) all the operations are grounded (there are actual operations that can be executed). In addition, it generates warnings on (5) unused data and (6) unused operations that don't participate in producing expected results. Given any errors detected, CAT generates a set of fixes that can be potentially used by the user. The following shows the general algorithms that are used in checking errors and generating suggestions. The italicized parts are supported by queries to the knowledge base.

- **Checking Unachieved Expected Results**:
  - Detect problem: for each expected result, check if it is linked to an output of an operation or directly linked to any of the initial input (i.e., the result is given initially).
  - Help user fix problem:
    1. find any available data (initial input or output from introduced operations) that is *subsumed by the data type of the desired result*, and suggest to add a link
    2. *find operation types in the task ontology where an output is subsumed by the desired data type* and all the input are provided (i.e., *subsumed by either the initial input or some output from introduced operations*), and suggest to add the operation types.
    3. *find operation types where an output is subsumed by the data type of the desired result*, and suggest to add the operation types.
- **Checking Missing Data**:
  - Detect problem: for each operation introduced, for each input parameter of the operation, find if it is linked to any (either to the initial input or to some output from introduced operations).
  - Help user fix problem:
    1. find any initial input data or output of operations that is *subsumed by the desired data type*, and suggest to add a link.
    2. *find operation types in the task ontology where an output is subsumed by the desired data type* and all the input are provided (i.e., *subsumed by either the initial input or some output from introduced operations*), and suggest to add the operation types.
    3. *find operation types where an output is subsumed by the desired data type*, and suggest to add the operation types.
- **Checking Inconsistent Links**:
  - Detect problem: for each link between data types, *find if the former one is subsumed by the latter one*.
  - Help user fix problem:
    1. *find operation types where an output is subsumed by the latter one and an input subsumes the former one*, and suggest to add the operation types.
- **Checking Ungrounded Operation**:
  - Detect problem: for each operation type introduced in the pathway, check *if there is a mapping to an actual operation* that can be performed.
  - Help user fix problem:
    1. *find a set of qualifiers that can be used to specialize it* and suggest to replace the operation type with a more special one base on the qualifiers.
    2. *find the subconcepts of the task type in the task ontology* and suggest to choose one of them.
- **Checking Unused Data**:
  - Detect problem: for each initial input data type and the output from the introduced operations, check if it is linked to an operation or an expected result.
  - Help user fix problem:
    1. find any missing data or unachieved results that *subsumes the unused data type*, and suggest to add a link.
    2. *find operation types where an input subsumes the unused data and some output are subsumed by any of the missing data or unachieved results*, and suggest to add the operation types.
    3. *find operation types where an input subsumes the unused data*, and suggest to add the operation types.
- **Checking Unused Operation**:
  - Detect problem: for each operation introduced, check if its output or any output from its following operations is linked to an expected result.
  - Help user fix problem:
    1. suggest to add a link to connect the operation

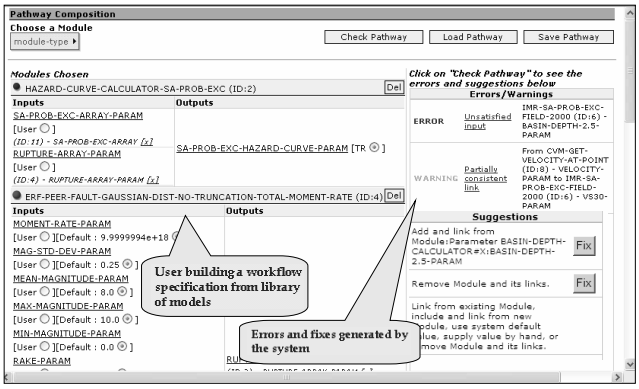

Figure 3: CAT Interface.

Whenever CAT detects an error, it sends an error message and a set of suggestions that can be used to fix the error. When there are more than one way of computing suggestions, CAT tries them according to the orders given in the algorithm (e.g., fix 1 then fix 2, ...). CAT also incorporate some heuristics for ordering errors and providing suggestions as they appear in the interface. More details of the algorithm and its formalism are available in (Kim, Spraragen, & Gil 2004; Spraragen, Kim, & Gil 2003).

Note that because the system has an ontology of operation types that describes high-level task types as well as specific

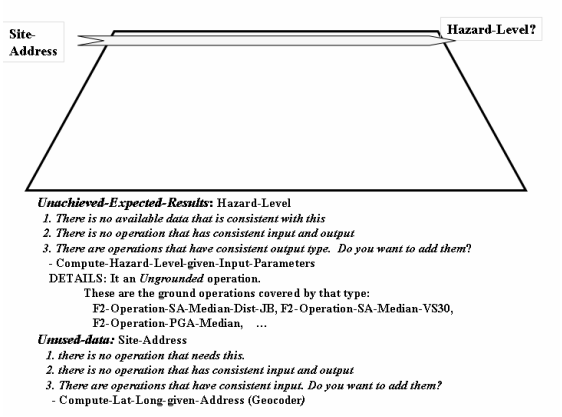

Figure 4: User's problem: Given site-address, find hazard-level.

operations that are mapped to actual operations, users can start from a high-level description of what they want without knowing the details of what operations are available. We often find that users have only partial description of what they want initially, and CAT can help users find appropriate service operations by starting with a high-level operation type and then specializing it while the pathway is being built. A general operation type can be specialized by itself or from the constraints introduced by other operations in the pathway.

Interactive Construction of a Computational Pathway

This section shows how the above checks and suggestions are used in helping users construct a computational pathway, using the problems described in Section 2. Our current implementation of CAT supports a textual editor where users can select services from a menu and make links by clicking their input and output parameters. Users can also apply the suggestions produced from CAT, which will update the pathways automatically, as shown in 3. Here we use conceptual diagrams in order to highlight CAT's report.

As described in Section 2, one of the key problems users often have in earthquake science is to analyze the hazard level of a given site. Most user may not know the details of the existing services, i.e., what are available and how to use them, and they may start with a high-level description the problem: Given a site address, compute the hazard level. The user may start with this high-level description of the task as shown in Figure 4.

Given this description of the task, CAT finds that the expected result (Hazard-Level) is not achieved (i.e. not initially given and not linked to any operations), so it generates a warning and a set of suggestions based on the strategies described in the previous section: first find any available data that is subsumed by the desired data type and then find operations that can produce the desired output from available data. Since none of that types are found, it then computes the operations that can just produce the output. A candidate found (Compute-Hazard-Level-given-Input-Parameters) is a high-level "ungrounded" operation (i.e., there is no actual operation of that type). However, there are operations that can produce more specific type of objects (e.g., F2-Operation-SA-Median-Magnitude, F2-Operation-SA-Median-VS30, etc.). CAT also notes that the given input (Site-Address) is not used yet. Since there is no operation that produces the unachieved result using the unused input, CAT suggests to add an operation that just uses the input (Compute-Lat-Long-given-Address). The user decides to add Compute-Hazard-Level-given-Input-Parameters in this case.

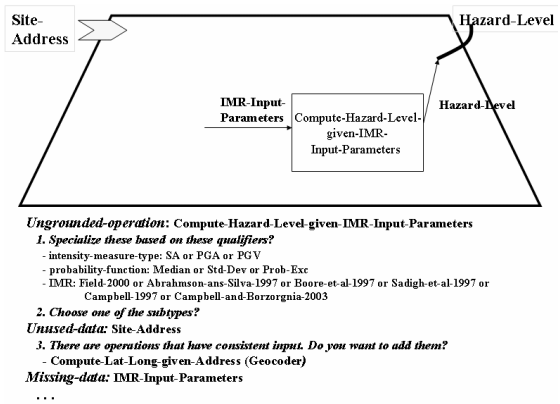

Figure 5: User adds Compute-Hazard-Level-give-IMR-Input-Parameters as suggested by CAT.

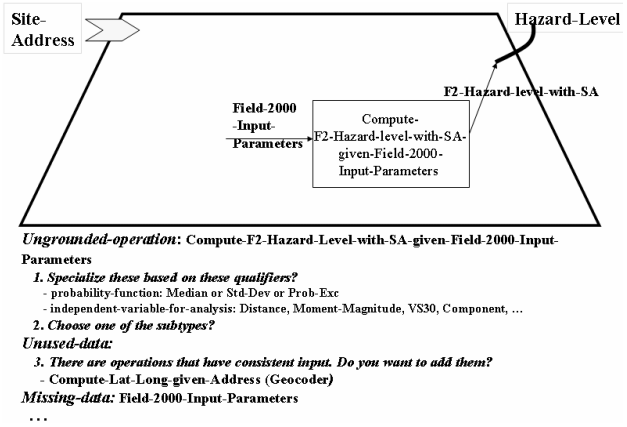

Figure 6: User specializes operation for SA and Field-2000.

Figure 5 shows that Compute-Hazard-Level-given-IMR-Input-Parameters instead of Compute-Hazard-Level-given-Input-Parameters is added to the computational pathway. Here the ontology of task types are used to find the most specific subsumer of all the grounded operations covered by the selected operation type. Since all the input parameters that the operations take are IMR-Input-Parameters, the operation is specialized into Compute-Hazard-Level-given-IMR-Input-Parameters.

CAT notes several problems as shown in Figure 5, including the operation being ungrounded yet (some of the CAT reports are not shown for brevity). For this type of problem, if the domain ontology has definitions of the qualifiers

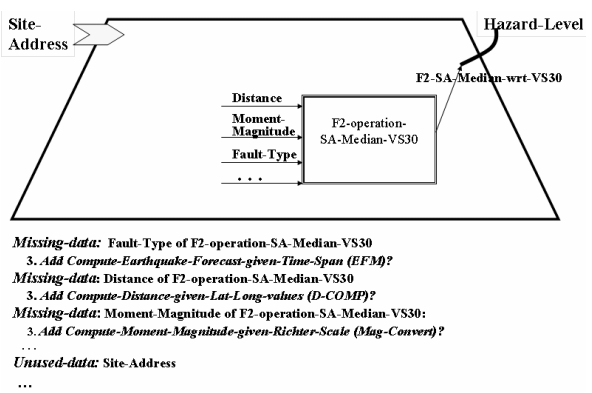

Figure 7: User selects Median and VS30 for further specialization.

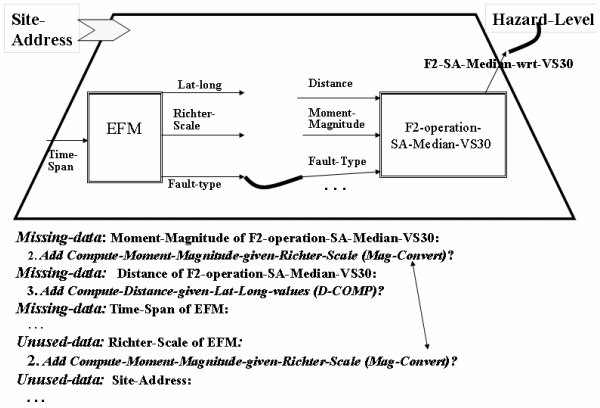

Figure 8: User adds EFM.

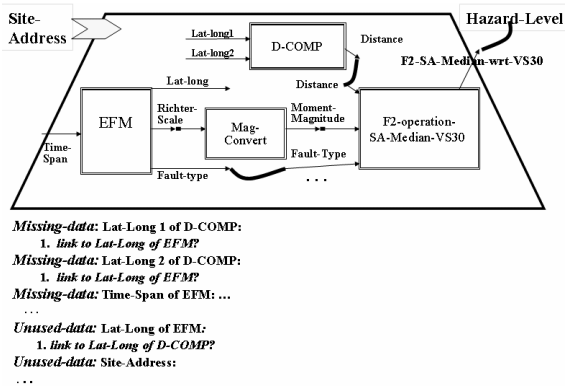

Figure 9: User adds Mag-Convert and D-COMP operations.

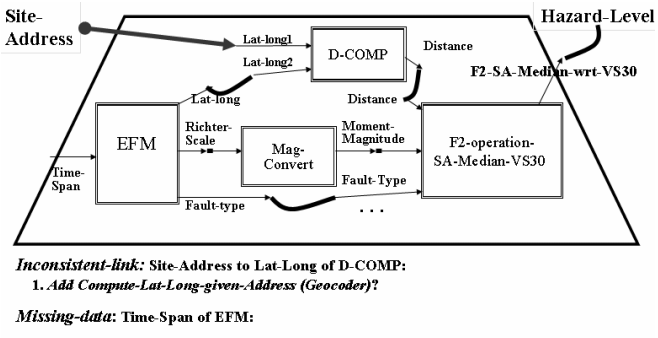

Figure 10: User adds a link between Site-Address and Lat-Long of D-COMP.

that distinguish different specializations of operation types, the system can exploit them in generating suggestions. For example, in our domain ontology, Hazard-Level can be specialized with respect to the intensity measure type (IMT), the probability function used in the analysis, the simulation module employed, etc. When there are a set of qualifiers, each of them may provide a different way of specializing abstract data types. That is, different combinations provide different paths to reach the grounded objects. For example, F2-SA-Median-wrt-VS30 can be reached by selecting SA as the intensity measure type, Median as the probability function, Field-2000 as the IMR used, and VS30 as the independent variable, in any order.

When the user picks SA as the intensity measure type, its input and output data types are recomputed according to the actual operations that are covered, as described above. That is, the output type changes from Earthquake-Hazard-Level to Earthquake-Hazard-Level-with-SA. Likewise, when the user choose FIELD-2000 as the IMR to use, its input type changes from IMR-Input-Parameters to FIELD-2000-input-Parameters (Figure 6).

This process of specializing ungrounded task can be continued until a ground operation is reached (F2-operation-SA-Median-VS30) as shown in Figure 7. Since its input parameters are not connected yet, CAT reports those as *missing-data*. The existing input (Site-Address) is not compatible with any of the desired (missing) data types, CAT uses fix type 3 and suggests to add additional steps that have consistent output types. For example, Compute-Earthquake-Forecast-given-Time-Span (a ground operation called EFM) can produce a Fault-Type.

Figure 8 shows the pathway after EFM is added. CAT finds that there a set of unused data and missing data. Note that one of the fixes can address two different issues: Compute-Moment-Magnitude-given-Richter-Scale can resolve missing-data (Moment-Magnitude of F2-operation-SA-Median-VS30) and unused-data (Richter-Scale of EFM).

When the user adds Mag-Convert, and then D-COMP, as shown in Figure 9, CAT notes a mapping between Lat-Long output from EFM and Lat-Long input of D-COMP from the checks on missing-data and unused-data.

Figure 10 shows an example of inconsistent link where Site-Address is directly linked to an incompatible data type, Lat-Long. In this case, CAT suggests to add an operation (Geocoder).

Figure 11 shows the result after adding Geocoder. The user can continue this process until all the expected results are achieved, all the necessary input data are provided, there are no inconsistent links, and all the operations be-

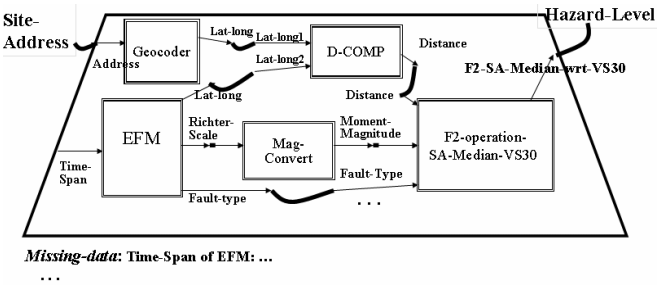

Figure 11: User adds Geocoder service operation.

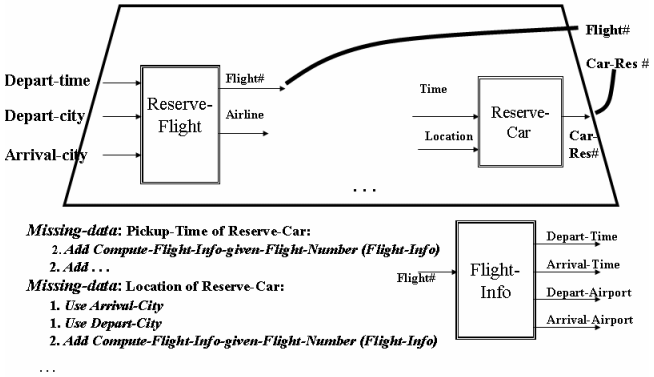

Figure 12: Interactive Service Composition for Travel Planning.

come ground. As shown above, checks on missing-data and unachieved results can give hints on unused-data, and vice versa. If an action can address more than one issues, it might be a better choice than than others. Results on inconsistent-link and ungrounded-operation are used for checking links entered by users and the groundedness of the operations.

Composing Services for Travel Planning

This section shows that how the same approach can be used in other domains. Figure 12 shows a process of composing services for a travel planning. The user wants to reserve a flight first and then reserve a car based on the reserved flight. Currently two input parameters of the Reserve-Car operation, (pickup) time and location, are not linked yet. Both of them can be potentially linked if the Flight-Info operation is added in between, since it produces data on Time (Depart-Time and Arrival-Time) and Location (Depart-Airport and Arrival-Airport) given a flight number.

As domain ontologies become richer in content, the system will be able to make more specific suggestions. For example, currently, the system cannot provide a direct mapping between input and output parameters of Flight-Info and Reserve-Car since each of them can be mapped to more than one sources. However, if a richer ontology of trips are given so that the pickup time and location should be consistent with the time that the airplane arrives at the Arrival-Airport, then the suggestions will become more specific. That is, the system will suggest to link Arrival-Time and Arrival-Airport to the Reserve-Car operation. The system will be able to filter out the option of using Depart-City as the pick-up location in the same way (in the suggestions for Missing-data in Figure 12).

Related Work

There are various related efforts concerned with composition of web services, but they concentrate on automatic composition and do not address the user interaction issues raised in this proposal. Existing approaches for composition of web services (Mcdermott 2002; Narayanan & McIlraith 2002; Sheshagiri, desJardins, & Finin 2003; Thakkar *et al.* 2000) use expressive languages and sophisticated reasoning and planning techniques to generate valid compositions of services. They complement our work in that they do not address user interaction issues. The Web Services TookKit (WSTK) (Srivastava 2000) includes a composition engine, but it has very limited models of the data used by the services that limits the support that underlying reasoners can provide. Little attention has been paid to the interactive composition of services. SWORD (Ponnekanti & A.Fox 2002) is a toolkit that addresses interactive service composition. However, it is designed for developers who have programming skills, not for end users that are the target users of our work.

Graphical tools to lay out a workflow and draw connections among steps abound (Chin *et al.* 2002; BizTalk 2003; KHOROS 2003; ProcedureCharter 2003; SmartDraw 2003) but the tools are limited to simple checks on the process models because there is no semantics associated to the individual steps and links. In contrast, we assume a knowledge-rich environment where the system can check whether the workflow makes sense within the background knowledge that it has.

Web service composition has many parallels with software composition (Jennings 2001; Heineman & Councill 2001), though there are many significant differences including the distributed nature of web services and the encapsulation techniques that they provide. Web services can be seen as a higher layer of abstraction than software provides, and can be extended with expressive languages to support composition in more powerful ways. Similar formal techniques to those in this paper have been used successfully to integrate software components (Stickel *et al.* 1993; Chien & Mortensen 1996; Lansky *et al.* 1995), but only in fully automated settings.

DAML-S is a semantic markup language for services that enables the expression of a complex service as well as the composition of services (Ankolekar 2002). Although it supports semantic description of web services by means of ontologies, the descriptions haven't been fully exploited in supporting the kinds of interactions described above.

Some languages to support composition of services are based on expressive formalisms to represent complex combinations of services (McIlraith & Fadel 2002; McIlraith & Son 2002). These languages include, for example, conditional expressions. This work is complementary in that it investigates the formal underpinnings of such languages, while our focus is on usability.

Discussion and Future Work

This paper presents a framework for interactive service composition where the system assists users in constructing a computational pathway by exploiting semantic description of services. We have built a tool that analyzes a sketch of a pathway based on the definitions of task types and their input and output data types, and generates error messages and specific suggestions to users.

We believe that our framework can be applied various problems if appropriate domain ontologies can be accessed and services are represented according to the domain terms defined in the ontologies. For example we may exploit the ontologies now available on-line including the ones available in the DAML ontology library(DAML-Ontology 2003) that are reusable across different applications. Task ontologies are relatively rare but may become more commonplace if they have clear value added in supporting more flexible composition of services. Currently our task ontologies are manually built, but we are planning to investigate a way of generating a hierarchy of general task types according to the kinds of input and output of given operations.

We are also investigating uses of automatic composition approaches in our interactive framework. For example, when users want to see possible completion of the pathways given their initial sketches, the system may send a request to an AI planning module.

Acknowledgments

We would like to thank Marc Spraragen for his contribution to our discussions. This research was funded by National Science Foundation (NSF), award number EAR-0122464.

References

Ankolekar, A. 2002. Daml-s: Web service description for the semantic web. In *Proc. 1st Int'l Semantic Web Conf. (ISWC 02)*.

BizTalk. 2003. http://www.microsoft.com/biztalk/.

Burstein, M.; McDermott, D.; Smith, D. R.; and Westfold, S. J. 2000. Derivation of glue code for agent interoperation. In Sierra, C.; Gini, M.; and Rosenschein, J. S., eds., *Proceedings of the Fourth International Conference on Autonomous Agents*, 277–284. Barcelona, Catalonia, Spain: ACM Press.

Chien, S., and Mortensen, H. 1996. Image processing for scientific data analysis of a large image database. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(8):854–859.

Chin, G.; Leung, R.; Schuchardt, K.; and Gracio, D. 2002. New paradigms in problem solving environments for scientific computing. In *Proceedings of the Intelligent User Interfaces*.

Christensen, E.; Curbera, F.; Meredith, G.; and Weerawarana, S. 2003. WSDL: Web service description language. http://www.w3.org/TR/wsdl.

DAML-Ontology. 2003. http://www.daml.org/ontologies.

Fillmore, C. 1968. The case for case. In *Universals in Linguistic Theory*.

Gil, Y., and Blythe, J. 2000. How can a structured representation of capabilities help in planning? In *Proceedings of the AAAI workshop on Representational Issues for Real-World Planning Systems*.

Heineman, G., and Councill, W. 2001. *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley.

Jennings, N. 2001. Building complex software systems: The case for an agent-based approach.

KHOROS. 2003. http://www.khoral.com/.

Kim, J., and Gil, Y. 2000. Acquiring problem-solving knowledge from end users: Putting interdependency models to the test. In *Proceedings of AAAI-2000*, 223–229.

Kim, J., and Gil, Y. 2001. Knowledge analysis on process models. In *Proceedings of IJCAI-2001*, 935–942.

Kim, J.; Spraragen, M.; and Gil, Y. 2004. An intelligent assistant for interactive workflow composition. In *Proceedings of the Intelligent User Interfaces Conference*.

Lansky, A.; Friedman, M.; Getoor, L.; Schmidler, S.; and Short, N. 1995. The collage/khoros link: Planning for image processing tasks. In *AAAI Spring Symposium on Integrated Planning Applications*.

Mcdermott, D. 2002. Estimated-regression planning for interactions with web services. In *AI planning systems Conference*.

McIlraith, S., and Fadel, R. 2002. Planning with complex actions.

McIlraith, S., and Son, T. 2002. Adapting golog for composition of semantic web services.

Narayanan, S., and McIlraith, S. 2002. Simulation, verification and automated composition of web services.

Paolucci, M.; Kawmura, T.; Payne, T.; and Sycara, K. 2002. Semantic matching of web services capabilities. In *First Int. Semantic Web Conf.*

Ponnekanti, S., and A.Fox. 2002. A. sword: A developer toolkit for web service composition. In *Proc. of the Eleventh International World Wide Web Conference*.

ProcedureCharter. 2003. http://www.imagespro.com/programs/2118/.

Sheshagiri, M.; desJardins, M.; and Finin, T. 2003. A planner for composing services described in daml-s. In *ICAPS 2003 Workshop on Planning for Web Services Program*.

SmartDraw. 2003. http://www.smartdraw.com/.

Spraragen, M.; Kim, J.; and Gil, Y. 2003. A formal model for intelligently assisted interactive workflow composition. ISI internal report.

Srivastava, B. 2000. Web services toolkit (WSTK).

Stickel, M.; Waldinger, R.; Lowry, M.; Pressburger, T.; and Unerwood, I. 1993. Deductive composition from astronomical software libraries. Technical Report, SRI International and NASA Ames Research Center.

Sycara, K.; Lu, J.; and Klusch, M. 1999. Dynamic service matchmaking among agents in open information environments. In *Journal ACM SIGMOD Record, Special Issue on Semantic Interoperability in Global Information Systems*.

Thakkar, S.; Knoblock, C.; Ambite, J.; and Shahabi, C. 2000. Dynamically composing web services from on-line sources. In *AAAI Workshop on Intelligent Service Integration*.