Towards Interactive Composition of Semantic Web Services (DRAFT)

Jihie Kim and Yolanda Gil

Information Sciences Institute, University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292, U.S.A.
{jihie, 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 computational pathways to put together 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 takes a sketch of a pathway and generates error messages and suggestions to users.

1 Introduction

Web services are a promising framework for reasoning about intelligent components on the web [3, 7, 15]. Web service composition is an important challenge for this vision. There has been a lot of interest in exploiting and integrating increasing number of services available on the web in order to reduce time and effort to build new applications.

Most approaches to web service composition address automatically composing the services[12, 4, 14, 17]. However, in many contexts users will want to drive the process, influencing the selection of components and their configuration. For example, there has been increased interest in approaches to draw users into automated processes [13, 16]. In this framework, users express their choices in constructing a solution and the system may assist the users by providing intelligent suggestions. Interactive service composition poses additional challenges to composing services. First, the system needs to combine user advice with automatic composition, and second, users may make mistakes and the system needs to help fix them. Another problem is that user's input is often incomplete and may even be incompatible with existing service descriptions.

We found such issues arising in constructing *computation pathways* in earthquake science domain where engineers interactively compose existing on-line 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 be benefit a lot from the assistance of intelligent tools that help them specify pathways correctly.

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 [10]. 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 use these ontologies in examining user's solutions (i.e., composition of services) and generating error messages and suggestions.

The tools 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 domain. 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 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 domain and travel planning. Finally we discuss remaining issues and future plans.

2 Background: 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: 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 services 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 as 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

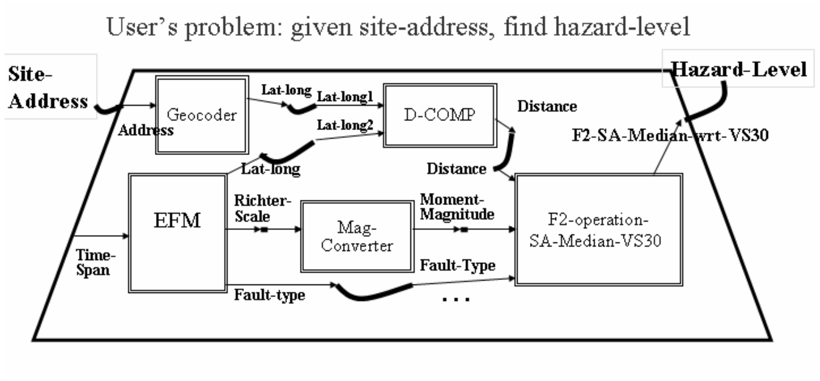


Fig. 1. An example computational pathway in earthquake science domain.

Earthquake Forecast Models (EFMs) that suggest entire sequences of earthquake forecasts around the area where the site is located. Users can choose different IMRs to use depending on the situation at hand because each IMR is designed to take into account of 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.

In fact, to be able to answer the question of determining the hazard level given a site, we may need to put together various service components, as shown in Figure 1, considering the overall task given and the constraints associated with each service 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 construction a computational pathway, users have to find the component they need, specify links between the components using their input and output, and check whether the expected outcome is generated from the composition. This type of problem in general is very difficult for end users who don't have computer science background [10, 9]. 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.

3 Approach

This section describes our approach to address the problems described in the previous section. 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.

3.1 Representing services using domain ontologies

```
<!-- WSDL description of the Field-2000 Web APIs.-->
<definitions name="urn:Field_2000Query"
             targetNamespace="urn:Field_2000Query"
             xmlns:typens="urn:Field_2000Query"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             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"/>
    <part name="BASIN-DEPTH-2.5"          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-Distance-JB">
      <input message="typens:F2-operation-SA-Median-Distance-JB-Request"/>
      <output message="typens:F2-operation-SA-Median-Distance-JB-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 - RPC, SOAP over HTTP -->
  <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 an existing WSDL (Web Service Description Language) representation [18] of an earthquake simulation service. This description has been built to support other existing applications that are developed before we start this effort. 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 call 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.

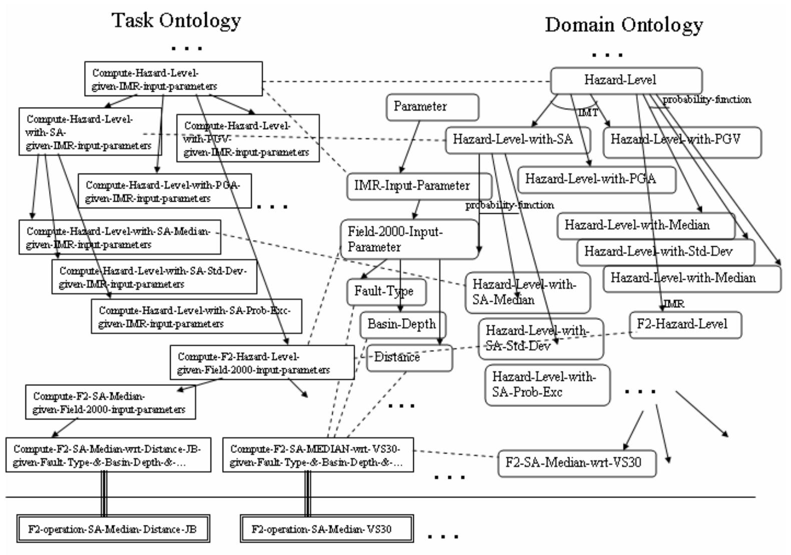

Task Ontology
Domain Ontology
Compute-Hazard-Level-given-IMR-input-parameters
Compute-Hazard-Level-with-SA-given-IMR-input-parameters
Compute-Hazard-Level-with-PGV-given-IMR-input-parameters
Compute-Hazard-Level-with-PGA-given-IMR-input-parameters
Compute-Hazard-Level-with-SA-Median-given-IMR-input-parameters
Compute-Hazard-Level-with-SA-Std-Dev-given-IMR-input-parameters
Compute-Hazard-Level-with-SA-Prob-Exc-given-IMR-input-parameters
Compute-F2-Hazard-Level-given-Field-2000-input-parameters
Compute-F2-SA-Median-given-Field-2000-input-parameters
Compute-F2-SA-Median-wrt-Distance-JB-given-Fault-Type-&-Basin-Depth-&-...
Compute-F2-SA-MEDIAN-wrt-VS30-given-Fault-Type-&-Basin-Depth-&-...
F2-operation-SA-Median-Distance-JB
F2-operation-SA-Median-VS30
Hazard-Level
IMT
probability-function
Parameter
Hazard-Level-with-SA
Hazard-Level-with-PGV
Hazard-Level-with-PGA
IMR-Input-Parameter
probability-function
Field-2000-Input-Parameter
Hazard-Level-with-Median
Hazard-Level-with-Std-Dev
Hazard-Level-with-Median
Fault-Type
Basin-Depth
Distance
Hazard-Level-with-SA-Median
Hazard-Level-with-SA-Std-Dev
Hazard-Level-with-SA-Prob-Exc
IMR
F2-Hazard-Level
F2-SA-Median-wrt-VS30

Fig. 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. There are now many domain ontologies available on-line including the ones in DAML ontology library[5]. We hope that our system will be able to exploit these existing ontologies in extending the descriptions of existing services.

The current implementation uses Loom for representing these ontologies. (Loom is a knowledge representation system of the KL-one family [11].) For example, the above description of Field 2000 can be represented as a set of Loom 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-Earthquake-Hazard-Level-given-IMR-Input-Parameters). Each parameter points to its format in the original description (r-format) as well as the mapping to a data type in the domain ontology (r-value-type).

```
(tell
 (:about FIELD-2000-Service IMR Service
      (:filled-by r-operation F2-operation-SA-Median-VS30
                  F2-operation-SA-Median-Distance-JB
                  F2-operation-SA-Median-Basin-Depth
                  F2-operation-SA-Median-Magnitude
                  F2-operation-SA-Median-SA-Period
                  F2-operation-SA-Std-Dev-VS30
                  ... ))
 (:about F2-operation-SA-Median-VS30
         service-operation
         Compute-F2-Hazard-level-with-SA-Median-wrt-VS30-given-Fault-Type-&-Basin-Depth-&-...
         (r-name "F2-operation-SA-Median-VS30")
         (r-input-message F2-SA-Median-VS30-request)
         (r-output-message F2-SA-Median-VS30-response))

 (:about F2-SA-Median-VS30-request
         input-message
         (r-name "F2-operation-SA-Median-VS30-Request")
         (:filled-by r-parameter F2-Magnitude F2-VS30 F2-Basin-depth-2.5
                     F2-Fault-Type F2-Component F2-Distance-JB
                     F2-SA-damping F2-SA-Period))
 (:about F2-Magnitude (r-value-type Moment-Magnitude)
         (r-name "MOMENT-MAGNITUDE") (r-format String))
 (:about F2-Distance-JB (r-value-type Distance)
         (r-name "DISTANCE-JB") (r-format String))
 (:about F2-Fault-type (r-value-type Fault-type)
         (r-name "FAULT-TYPE-PARAMETER") (r-format String))
 (:about F2-VS30 (r-value-type VS30) (r-name "VS30") (r-format String))
 (:about F2-basin-depth-2.5 (r-value-type Basin-Depth)
         (r-name "BASIN-DEPTH-2.5") (r-format String))
  ...
 (:about F2-SA-Median-VS30-response
         output-message
         (r-name "F2-operation-SA-Median-VS30-Response")
         (:filled-by r-parameter F2-SA-Median-VS30-result))
 (:about F2-SA-Median-VS30-result
         (r-value-type F2-SA-Median-wrt-VS30)
         (r-name "return") (r-format String))
 ...)
```

3.2 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.

- **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 most general operation types in the task ontology where an output is subsumed by the desired data type and all the input are providable (i.e., subsumed by either the initial input or some output from introduced operations), and suggest to add the operation types.
    3. find most general operation types where an output is subsumed by the data type of the desired result, and suggest to add the operation types.
- **Checking Unprovided 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 most general operation types in the task ontology where an output is subsumed by the desired data type and all the input are providable (i.e., subsumed by either the initial input or some output from introduced operations), and suggest to add the operation types.
    3. find most general 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 most general 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 unprovided data or unachieved results that subsumes the unused data type, and suggest to add a link.
    2. find most general operation types where an input subsumes the unused data and some output are subsumed by any of the unprovided data or unachieved results, and suggest to add the operation types.
    3. find most general 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

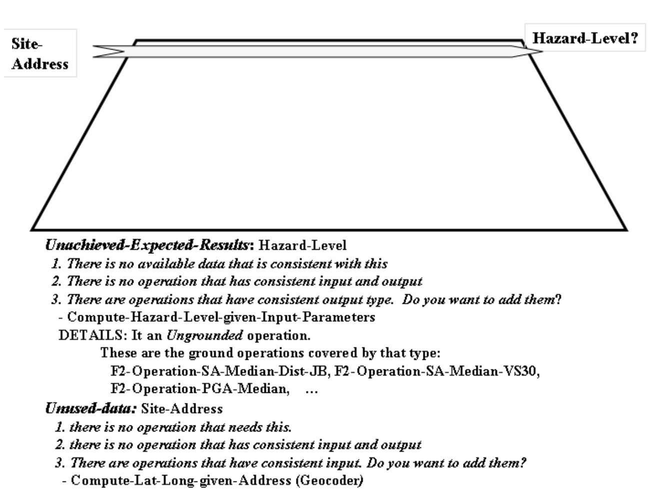
Whenever CAT detects an error, it sends an error message and a set of suggestions that can be used to fix the error. When there are more than one way of computing suggestions, CAT tries them according to the orders given in the algorithm (e.g., fix 1 then fix 2, ...).

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

4 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 functions are not linked to a graphical user interface yet. Here we use conceptual diagrams to make CAT's report more readable.

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 3.



***Unachieved-Expected-Results*:** Hazard-Level
1. *There is no available data that is consistent with this*
2. *There is no operation that has consistent input and output*
3. *There are operations that have consistent output type. Do you want to add them*?
   - Compute-Hazard-Level-given-Input-Parameters

   DETAILS: It an *Ungrounded* operation.
   These are the ground operations covered by that type:
   F2-Operation-SA-Median-Dist-JB, F2-Operation-SA-Median-VS30,
   F2-Operation-PGA-Median, ...

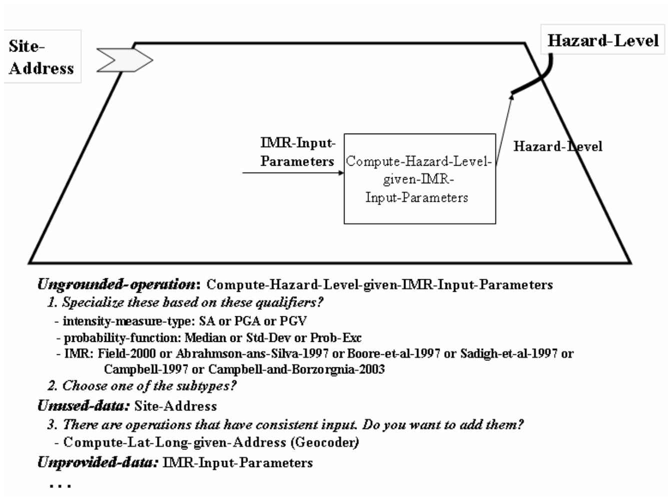
Unused-data: Site-Address
1. *there is no operation that needs this.*
2. *there is no operation that has consistent input and output*
3. *There are operations that have consistent input. Do you want to add them?*
   - Compute-Lat-Long-given-Address (Geocoder)

Fig. 3. User's problem: Given site-address, find hazard-level.

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-Dist-JB, 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 4 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 4, 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 that distinguish different specializations of operation types, the system can exploit them in generating suggestions. For example, in our domain ontology, Hazard-



***Ungrounded-operation*:** Compute-Hazard-Level-given-IMR-Input-Parameters

1. Specialize these based on these qualifiers?

- intensity-measure-type: SA or PGA or PGV
- probability-function: Median or Std-Dev or Prob-Exc
- IMR: Field-2000 or Abrahmson-ans-Silva-1997 or Boore-et-al-1997 or Sadigh-et-al-1997 or Campbell-1997 or Campbell-and-Borzorgnia-2003

2. Choose one of the subtypes?

Unused-data: Site-Address

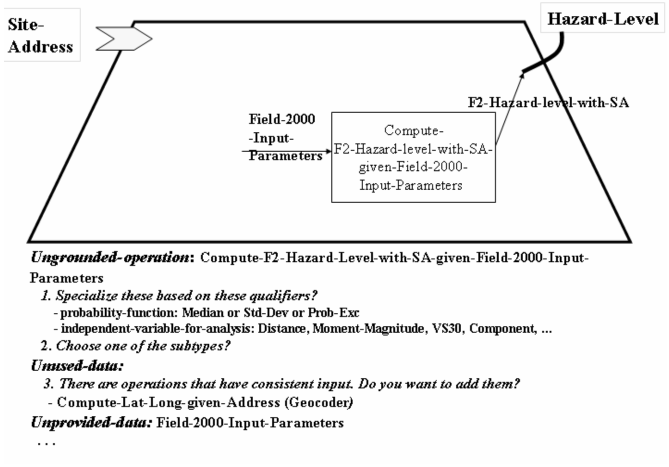
3. There are operations that have consistent input. Do you want to add them?

- Compute-Lat-Long-given-Address (Geocoder)

Unprovided-data: IMR-Input-Parameters

...

Fig. 4. User indicates use of suggested operation.


Site-Address
Hazard-Level
F2-Hazard-level-with-SA
Field-2000-Input-Parameters
Compute-F2-Hazard-level-with-SA-given-Field-2000-Input-Parameters

***Ungrounded-operation*:** Compute-F2-Hazard-Level-with-SA-given-Field-2000-Input-Parameters

1. Specialize these based on these qualifiers?

- probability-function: Median or Std-Dev or Prob-Exc
- independent-variable-for-analysis: Distance, Moment-Magnitude, VS30, Component, ...

2. *Choose one of the subtypes?*

Unused-data:

3. There are operations that have consistent input. Do you want to add them?

- Compute-Lat-Long-given-Address (Geocoder)

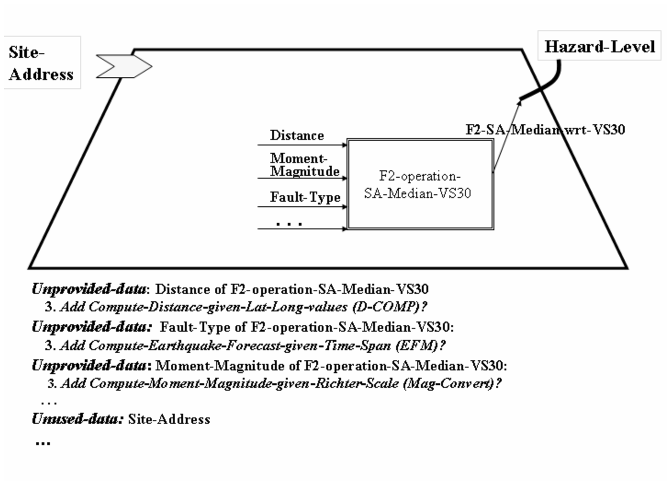
Unprovided-data: Field-2000-Input-Parameters

...

Fig. 5. User selects SA and Field-2000.

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 5).

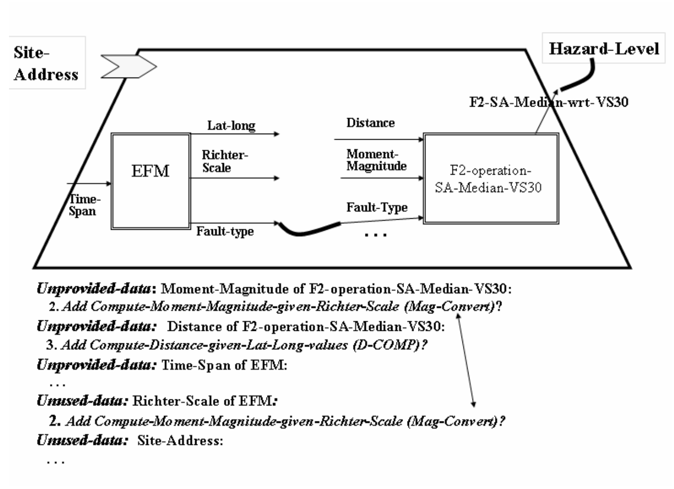


Unprovided-data: Distance of F2-operation-SA-Median-VS30
3. *Add Compute-Distance-given-Lat-Long-values (D-COMP)?*
Unprovided-data: Fault-Type of F2-operation-SA-Median-VS30:
3. *Add Compute-Earthquake-Forecast-given-Time-Span (EFM)?*
Unprovided-data**:** Moment-Magnitude of F2-operation-SA-Median-VS30:
3. *Add Compute-Moment-Magnitude-given-Richter-Scale (Mag-Convert)?*
...
Unused-data: Site-Address
...

Fig. 6. User selects Median and VS30 for further specialization.

This process of specializing ungrounded task can be continued until a ground operation is reached (F2-operation-SA-Median-VS30) as shown in Figure 6. Since its input parameters are not connected yet, CAT reports those as *unprovided-data*. The existing input (Site-Address) is not compatible with any of the desired (unprovided) 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.

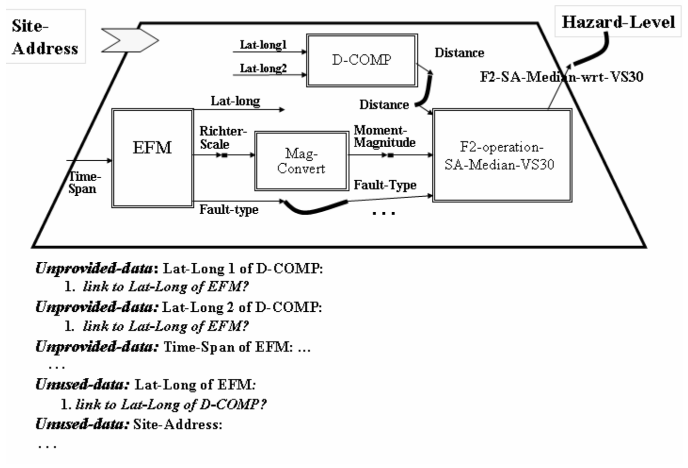
Figure 7 shows the pathway after EFM is added. CAT finds that there a set of unused data and unprovided data. Note that one of the fixes can address two different issues: Compute-Moment-Magnitude-given-Richter-Scale can re-



***Unprovided-data*:** Moment-Magnitude of F2-operation-SA-Median-VS30:
2. *Add Compute-Moment-Magnitude-given-Richter-Scale (Mag-Convert)?*
Unprovided-data: Distance of F2-operation-SA-Median-VS30:
3. *Add Compute-Distance-given-Lat-Long-values (D-COMP)?*
Unprovided-data: Time-Span of EFM:
. . .
Unused-data: Richter-Scale of EFM:
2. *Add Compute-Moment-Magnitude-given-Richter-Scale (Mag-Convert)?*
Unused-data: Site-Address:
. . .

Fig. 7. User adds EFM.

solve unprovided-data (Moment-Magnitude of F2-operation-SA-Median-VS30) and unused-data (Richter-Scale of EFM).

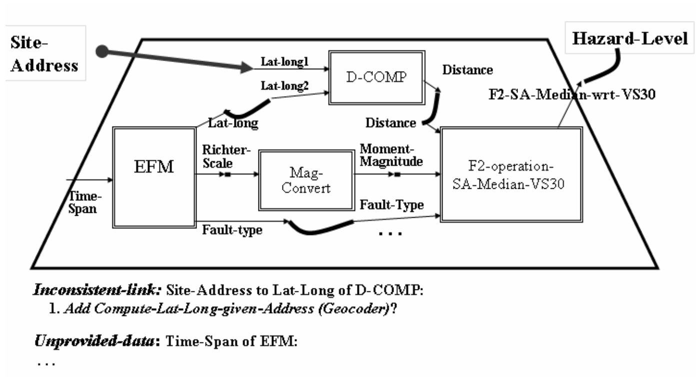


***Unprovided-data*:** Lat-Long 1 of D-COMP:
1. *link to Lat-Long of EFM?*
Unprovided-data: Lat-Long 2 of D-COMP:
1. *link to Lat-Long of EFM?*
Unprovided-data: Time-Span of EFM: ...
. . .
Unused-data: Lat-Long of EFM:
1. *link to Lat-Long of D-COMP?*
Unused-data: Site-Address:
. . .

Fig. 8. User adds Mag-Convert and D-COMP.

When the user adds Mag-Convert, and then D-COMP, as shown in Figure 8, CAT notes a mapping between Lat-Long output from EFM and Lat-Long input of D-COMP from the checks on unprovided-data and unused-data.

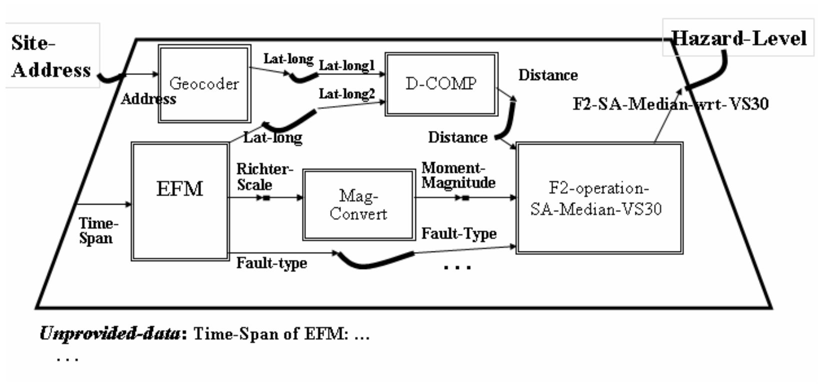
Figure 9 shows an example of inconsistent link where Site-Address is directly linked to an incompatible data type, Lat-Long. In this case, CAT suggests to add an operation (Geocoder).



Inconsistent-link: Site-Address to Lat-Long of D-COMP:
1. *Add Compute-Lat-Long-given-Address (Geocoder)*?

***Unprovided-data*:** Time-Span of EFM:
...

Fig. 9. User adds a link between Site-Address and Lat-Long of D-COMP.

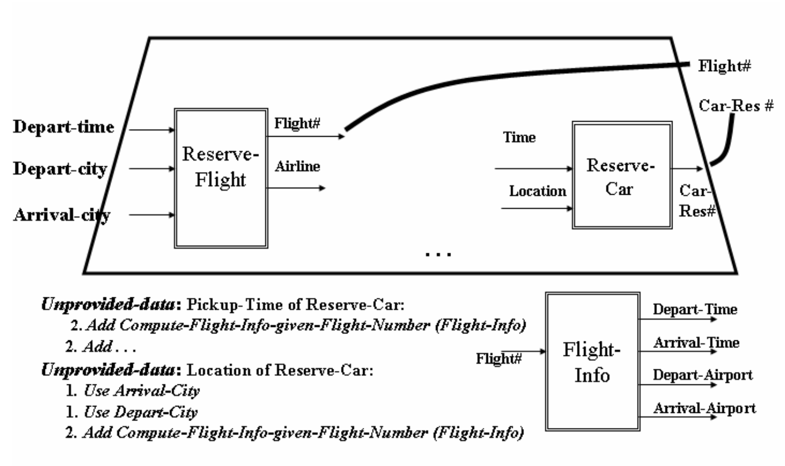


***Unprovided-data*:** Time-Span of EFM: ...
...

Fig. 10. User adds Geocoder.

Figure 10 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 become ground. As shown above, checks on unprovided-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.

5 Composing Services for Travel planning



***Unprovided-data*:** Pickup-Time of Reserve-Car:
2. *Add Compute-Flight-Info-given-Flight-Number (Flight-Info)*
2. *Add . . .*

***Unprovided-data*:** Location of Reserve-Car:
1. *Use Arrival-City*
1. *Use Depart-City*
2. *Add Compute-Flight-Info-given-Flight-Number (Flight-Info)*

Fig. 11. Travel Planning.

This section shows that how the same approach can be used in other domains. Figure 11 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 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. In this case, the system will not be able to suggest 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. For example, both Depart-Time and Arrival-Time output from Flight-Info are consistent with the Time input needed for Reserve-Car. 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. In addition, 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 Unprovided-data in Figure 11).

6 Related Work

Most of existing service composition approaches focus on developing automated techniques for generating compositions that satisfy user given goals [12, 4, 14, 17]. Our work provides an alternative and complementary approach addressing the issues that arise when users want to influence the composition process including selection of components and their configuration. Its interactive framework also helps when the user has an incomplete description of the task or even an incompatible one since he/she can start from high-level task types and then specialize them while the pathway is being built, based on the constraints that are gradually introduced in the pathway.

CommonKADS[2]'s hierarchy of generic problem solving tasks and our task ontology are similar in spirit in that both are built to help users identify appropriate task types that can be used for solving a given problem. Recent work in UPML[6] also provides a framework for helping users access and reuse libraries of generic problem solving components available on the internet. However, their work does not directly address issues that arise in interactive service composition, including how to exploit task ontologies to interactively compose a solution, how to handle errors and gaps, etc.

Web Service Flow Language (WSFL) provides an XML language for describing web service composition, defining types of interrelations between services. Although the language provides a useful tool, it is not designed to exploit domain knowledge and ontologies which we believe is a key technology to providing strong guidance.

7 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 DAML ontology library[5] that are reusable across different applications. Task ontologies are relatively new in the field and rare with the exception of CommonKADS ontology[2]. 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 given operations take.

In generating error messages and suggestions, we plan to develop heuristics on prioritizing them according to their relations. For example, if an action can fix more than one problems it will be more desirable than other actions. This capability will be useful when there are many problems and fixes found. We also plan to build a more comprehensive list of fixes including modifying steps, changing orderings, etc., based on dependencies between the problems.

We are also working on a graphical user interface that will be used in entering computational pathways and presenting feedback to the user.

Acknowledgements

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

1. Ankolekar, A., Huch, F., Sycara., K.: Concurrent Execution Semantics for DAML-S with Subtypes. The First International Semantic Web Conference (2002).
2. Breuker J. and Van de Velde, W.(eds): CommonKADS Library for Expertise Modeling. IOS Press, (1994).
3. Burstein, M., Hobbs, J., Lassila, O., Martin, D., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K.: DAML-S: Web Service Description for the Semantic Web. International Semantic Web Conference (2002) 348-363.
4. Burstein, M., McDermott, D., Smith, D.: Derivation of glue code for agent interoperation. Agents (2000) 277-284.
5. DAML Ontology Library: http://www.daml.org/ontologies (2003).
6. Fensel, D. Benjamins, R., Motta, E., and Wielinga, B.: UPML: A Framework for knowledge system reuse. Proceedings of the International Joint Conference on AI (1999).
7. Hendelr, J.: Agents and the Semantic Web. IEEE Intellignet Systems (Special Issue on Semantic Web), 16(2), 2001.
8. Leymann, F. (eds): Web Services Flow Language (WSFL 1.0). IBM Software Group (2001).
9. Kim, J. and Gil, Y.: Acquiring problem-solving knowledge from end users: Putting interdependency models to the test. Proceedings of AAAI-2000, (2000).
10. Kim, J. and Gil, Y.: Knowledge Analysis on Process Models. Proceedings of International Joint Conference on AI (2001).
11. MacGregor, R.: LOOM User Manual. Working Paper ISI/WP-22 (1990)
12. McDermott, D..: Estimated-Regression Planning for Interactions with Web Services. AI planning systems Conference (2002).
13. Myers, K.: Planning with Conflicting Advice. Proceedings of the Fifth International Conference AI Planning and Scheduling (2000).
14. Paolucci, M., Kawamura, T., Payne, T., Sycara K.: Semantic Matching of Web Services Capabilities. The First International Semantic Web Conference (2002).
15. Payne, T., Singh, R., and Sycara, K.: Browsing Schedules - An Agent-based approach to navigating the Semantic Web. The First International Semantic Web Conference (2002).

16. Smith, S., Lassila, O., and Becker, M: Configurable mixed initiative systems for planning and scheduling. In A. Tate, editor, Advanced Planning Technology. AAAI Press, (1996).
17. Sycara, K., Lu, Klusch, M., and Widoff S.: Dynamic Service Matchmaking among Agents in Open Information Environments. Journal ACM SIGMOD Record, Special Issue on Semantic Interoperability in Global Information Systems, A. Ouksel, A. Sheth (Eds.), (1999).
18. W3C: WSDL specification. http://www.w3c.org/TR/WSDL/.