Wings Demo Walkthrough

For a brief overview of Wings, see http://www.isi.edu/~gil/slides/WingsTour-8-08.pdf

Last Update: August 22, 2008
Summary of Wings Demonstration

- **Wings can:**
  - Express high-level reusable workflow templates
  - Based on those templates, express high-level user requests that only partially specify what datasets, parameters, or software components are to be used
  - From a user request, generate automatically possible workflow candidates by searching for:
    - Choices of datasets
    - Choices of parameter values
    - Choices of software components
  - During that search, eliminate workflow candidates that are not viable because they contain invalid combinations of choices
  - For valid workflow candidates generated, translate to a format for submission to an execution engine
Outline of Demonstration

- Some Background
  - Data catalog and software component catalog

- Demo
  - Reusable high-level workflow templates
    - May leave unassigned datasets, parameters, and components
  - Seeds that a user can submit for automatic generation
    - Automatic assignment of parameter values
    - Automatic generation of dataset choices
    - Automatic selection of software components
    - Elimination of workflow candidates during automatic generation
  - Any workflow generated can become a template or a seed
Wings architecture assumes the existence of:

- An external data catalog that can answer to Wings API calls about datasets and their properties
- An external software component catalog (aka component catalog) that can answer to Wings API calls about software components and their properties

Therefore, Wings does not include an editor/browser for data catalogs or component catalogs

For this demo, we use two in-house catalogs built with the widely-known Irvine datasets and Weka software for machine learning and data mining

- Built in-house using ontologies and rules (can view in OWL editor)
- Could be built in any manner as long as compliant with Wings API
Background: Data Catalog Contents

- Datasets have types and other metadata properties
Components have arguments
- Can be input or output datasets or parameters
- Arguments have type constraints
- Each has a unique ID

Component ontology shows abstract classes of components as well as concrete instances
Software components have complex constraints about their use and behavior: how to set parameters based on data properties, for what kinds of datasets they are appropriate, etc.

- Can be implemented as rules, code, etc.

These constraints can be classified as:

- Forward propagation: use metadata properties of input datasets to infer properties of other input arguments and output arguments
- Backward propagation: use the metadata properties that describe desired output data to infer properties of input arguments

Constraints can:

- Choose parameter values
- Infer required and predicted metadata properties
- Check valid use of a component within a workflow based on inferred and predicted metadata properties of its arguments
Outline of Demonstration

- Some Background
  - Data catalog and software component catalog

- Demo
  - Reusable high-level workflow templates
    - May leave unassigned datasets, parameters, and components
  - Seeds that a user can submit for automatic generation
    - Automatic assignment of parameter values
    - Automatic generation of dataset choices
    - Automatic selection of software components
    - Elimination of workflow candidates during automatic generation
  - Any workflow generated can become a template or a seed
- Workflow templates are high-level reusable workflow structures/patterns
- Workflow seeds are user requests for creating an executable workflow
A Simple Workflow Template

- Workflows have **Nodes** that indicate software component to be used
- **Links** that show dataflow among components
- **Data variables** (stubs)
- **Parameter variables** (stubs)

Note that the data type constraints coming from the components are not shown in this view.
Data variables can have type constraints, expressed as RDF triples.
Workflow Templates can Include Abstract Components

- Templates can include abstract component classes as well as concrete components (shown with a star)
Templates can specify datasets for data variables and values for parameters.

- Templates can specify values for parameter variables (to configure components), or indicate what datasets to use (to bind data variables). This is indicated with a star.

- Templates can be created from existing templates (show this here by creating this new template starting with the general one and adding the parameter value at the bottom).
Advanced Constraints in a Workflow Template

- Templates can include advanced constraints, which in Wings are represented as rules.
Outline of Demonstration

- Some Background
  - Data catalog and software component catalog

- Demo
  - Reusable high-level workflow templates
    - May leave unassigned datasets, parameters, and components
  - Seeds that a user can submit for automatic generation
    - Automatic assignment of parameter values
    - Automatic generation of dataset choices
    - Automatic selection of software components
    - Elimination of workflow candidates during automatic generation
  - Any workflow generated can become a template or a seed
A seed is formed by a workflow template combined with additional type constraints, parameter configurations, or dataset selections.

System will automatically search for possible choices for unspecified data and parameters.
Automatic Generation of Executable Workflows by Assigning Parameter Values

- System sets the value of the unassigned parameter automatically based on metadata properties of that dataset (configured workflows).
- Any configured workflow can be executed.
- Wings can generate a DAX for the Pegasus workflow mapping and execution system.
Viewing Configured Workflows

- Configured workflows have values for all parameters so all components are configured.
Configured Workflow in RDF and as an Executable DAX for Pegasus

RDF

DAX
Outline of Demonstration

- Some Background
  - Data catalog and software component catalog

- Demo
  - Reusable high-level workflow templates
    - May leave unassigned datasets, parameters, and components
    - Seeds that a user can submit for automatic generation
    - Automatic assignment of parameter values
    - Automatic generation of dataset choices
    - Automatic selection of software components
    - Elimination of workflow candidates during automatic generation
    - Any workflow generated can become a template or a seed
A User Seed Does Not Have to Specify All Datasets to be Used

- User does not have to specify all dataset selections (i.e., they may specify bindings only for some data variables)

- System will automatically search for possible choices for unspecified data (and parameters) that are compatible with other user choices
Automatic Generation of Workflow Candidates by Finding Dataset Choices

- System generates several workflow candidates each based on a different choice of training datasets (bound workflows).
- System sets the value of the unassigned parameter automatically based on metadata properties of that dataset (configured workflows).
- Any configured workflow can be executed (i.e., through a DAX for Pegasus).
Outline of Demonstration

- Some Background
  - Data catalog and software component catalog

- Demo
  - Reusable high-level workflow templates
    - May leave unassigned datasets, parameters, and components
    - Seeds that a user can submit for automatic generation
      - Automatic assignment of parameter values
      - Automatic generation of dataset choices
  - Automatic selection of software components
  - Elimination of workflow candidates during automatic generation
  - Any workflow generated can become a template or a seed
A User Does Not Have to Specify the Algorithms to be Used

- Users do not have to specify which algorithms to use, the seeds can use abstract components.
- System will automatically search for possible choices of components that are compatible with the datasets chosen.
Automatic Generation of Workflow Candidates by Finding Candidate Components

First, system finds different choices of algorithm instances of those abstract components and generates several workflow candidates (specialized workflows).
Then, system finds datasets and assigns parameter values for each candidate specialized workflow.

If a workflow candidate is not viable for that seed (i.e., its assignments are inconsistent) it would be eliminated.

In this example all candidates generated are valid, but not in the next example.
Outline of Demonstration

- Some Background
  - Data catalog and software component catalog

- Demo
  - Reusable high-level workflow templates
    - May leave unassigned datasets, parameters, and components
    - Seeds that a user can submit for automatic generation
    - Automatic assignment of parameter values
    - Automatic generation of dataset choices
    - Automatic selection of software components
    - Elimination of workflow candidates during automatic generation
  - Any workflow generated can become a template or a seed
Eliminating Candidate Workflows During the Generation Process

- Users do not have to specify which algorithms to use, the seeds can use abstract components.
- System will automatically search for possible choices of components that are compatible with the datasets.
Automatic Generation and Elimination of Workflow Candidates

- When a workflow candidate is not viable for that seed (i.e., its assignments are inconsistent), it is eliminated.
- Only feasible consistent choices of datasets, components, and parameter values lead to executable workflows.
Outline of Demonstration

- Some Background
  - Data catalog and software component catalog

- Demo
  - Reusable high-level workflow templates
    - May leave unassigned datasets, parameters, and components
  - Seeds that a user can submit for automatic generation
    - Automatic assignment of parameter values
    - Automatic generation of dataset choices
    - Automatic selection of software components
    - Elimination of workflow candidates during automatic generation
  - Any workflow generated can become a template or a seed
Workflow Candidates Can Be Selected to Become Templates and Seeds

- They can be edited and saved as templates and seeds for future reuse
Summary of Wings Demonstration

Wings can:

- Express high-level reusable workflow templates
- Express high-level user requests as seeds that only partially specify what datasets, parameters, or software components are to be used
  - A seed consists of a reusable template with additional specifications of datasets, parameter values, or data types
- Generate automatically possible workflow candidates for a seed by searching for:
  - Choices of datasets
  - Choices of parameter values
  - Choices of software components
- Eliminate workflow candidates that are not viable because they contain invalid combinations of choices
- Translate workflow candidates to a format for submission to an execution engine
Wings Uses External Services
## Wings API Calls to Component Services and their Use

<table>
<thead>
<tr>
<th>Function in Component Services</th>
<th>Purpose in Automatic Generation Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>c:inputs</td>
<td>Validate workflow template in request</td>
</tr>
<tr>
<td>c:parameters</td>
<td>Ground workflows to be submitted for execution</td>
</tr>
<tr>
<td>c:outputs</td>
<td>Use of abstract components in workflow templates that can be specialized in backward and forward sweep</td>
</tr>
<tr>
<td>c:args</td>
<td>Generate binding-ready workflows in backward sweep</td>
</tr>
<tr>
<td>c:invocation-command</td>
<td>Generate configured workflows in forward sweep</td>
</tr>
<tr>
<td>c:is-concrete</td>
<td></td>
</tr>
<tr>
<td>c:specialize</td>
<td></td>
</tr>
<tr>
<td>c:specialize-to-concrete</td>
<td></td>
</tr>
<tr>
<td>c:is-backward-enabled</td>
<td></td>
</tr>
<tr>
<td>c:find-DODs-given-output-requirements</td>
<td></td>
</tr>
<tr>
<td>c:is-forward-enabled</td>
<td></td>
</tr>
<tr>
<td>c:predict-DODs-given-input-requirements</td>
<td></td>
</tr>
<tr>
<td>c:is-configurable</td>
<td></td>
</tr>
<tr>
<td>c:configure</td>
<td></td>
</tr>
<tr>
<td>c:is-configured</td>
<td></td>
</tr>
<tr>
<td>c:estimate-performance</td>
<td>Rank candidate workflows</td>
</tr>
</tbody>
</table>
## Wings API Calls to Medatada Services and their Use

<table>
<thead>
<tr>
<th>Function in Metadata Services</th>
<th>Purpose in Automatic Generation Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>d:combine-DODs</td>
<td>Seed workflow templates</td>
</tr>
<tr>
<td>d:entity-DODs</td>
<td>Filter relevant data properties to be propagated in the workflow enables workflow candidate addition and elimination</td>
</tr>
<tr>
<td>d:assign-identifier</td>
<td>Create unique identifiers and properties for workflow data products so they can be reused in future workflows</td>
</tr>
<tr>
<td>d:assert-predicted-DODs</td>
<td>Selection of input data enables creation of bound workflows</td>
</tr>
<tr>
<td>d:find-data-objects</td>
<td>Propagation of input data properties in forward sweep enables component specialization and workflow candidate elimination</td>
</tr>
<tr>
<td>d:obtain-DODs</td>
<td></td>
</tr>
</tbody>
</table>