Domain-Specific Languages for Composing Signature Discovery Workflows

Ferosh Jacob*, Adam Wynne+, Yan Liu+, Nathan Baker+, and Jeff Gray*

*Department of Computer Science, University of Alabama, AL
+Pacific Northwest National Laboratory, Richland, WA
The most widely understood signature is the human fingerprint. Biomarkers can be used to indicate the presence of disease or identify a drug resistance. A signature is a unique or distinguishing measurement, pattern or collection of data that identifies a phenomenon (object, action or behavior) of interest. Anomalous network traffic is often an indicator of a computer virus or malware.

**Signature Discovery Initiative (SDI)**
• **Anticipate** future events by detecting precursor signatures, such as combinations of line overloads that may lead to a cascading power failure

• **Characterize** current conditions by matching observations against known signatures, such as the characterization of chemical processes via comparisons against known emission spectra

• **Analyze** past events by examining signatures left behind, such as the identity of cyber hackers whose techniques conform to known strategies and patterns
Challenge:
An approach that can be applied across a broad spectrum to efficiently and robustly construct candidate signatures, validate their reliability, measure their quality and overcome the challenge of detection.

Solution: Analytic Framework (AF)
- Legacy code in a remote machine is wrapped and exposed as web services,
- Web services are orchestrated to create re-usable tasks that can be retrieved and executed by users.
• Accidental complexity of creating service wrappers
  ❖ In our system, manually wrapping a simple script that has a single input and output file requires 121 lines of Java code (in five Java classes) and 35 lines of XML code (in two files).

• Lack of end-user environment support
  ❖ Many scientists are not familiar with service-oriented software technologies, which force them to seek the help of software developers to make Web services available in a workflow workbench.

Challenges for scientists in using AF
We applied Domain-Specific Modeling (DSM) techniques to

- **Model** the process of wrapping remote executables.
  The executables are wrapped inside AF web services using a Domain-Specific Language (DSL) called the Service Description Language (SDL).

- **Model** the SDL-created web services
  The SDL-created web services can then be used to compose workflows using another DSL, called the Workflow Description Language (WDL).

**A domain-specific modeling approach**
1. Submit job

2. Check status

3. Download files

Output generated as Taverna workflow executable
Example application: BLAST execution

Service description (SDL) for BLAST submission

```plaintext
service submitBlast {
  use ssh_oly;
  cmd "sh runJob.sh";
  resource "jobScript.sh", "runJob.sh";
  in doc blossum, params, fasta;
  out jobID, outDir;
  /* Inside runJob.sh
     * echo "jobID=$SLURM_JOBID" >.properties
     * echo "outDir=$OUTDIR" >>.properties
     */
}
```

Workflow description (WDL) for BLAST

```plaintext
use "SigAnalysis.sdl"
workflow BlastSearch (
  in blossum, in params, in fasta,
  out outFile, out status)
```
Implementation

Script metadata (e.g., name, inputs)
SDL (e.g., blast.sdl)
WDL (e.g., blast.wdl)

Inputs

Web services (e.g., checkJob)
Taverna workflow (e.g., blast.t2flow)

Outputs

Retrieve documents (AF framework)
Apply templates (Template engine)
Web services (Taverna engine)

Execution

@Runtime
• Compared to domain-independent workflows like JBPM and Taverna, our framework has the advantage that it is configured only for scientific signature discovery workflows.

• Most of these tools assume that the web services are available. Our framework configures the workflow definition file that declares how to compose services wrappers created by our framework.

Related works
We successfully designed and implemented two DSLs (SDL and WDL) for converting remote executables into scientific workflows. SDL can generate services that are deployable in a signature discovery workflow using WDL. We separated the domain-specific information required to create the workflows from the accidental complexities introduced by webservices and the Taverna workflow engine, which allows end-users (scientists) to design and develop workflows.
Questions ?
Example application: BLAST execution

1. Submit BLAST job in a cluster
2. Check the status of the job
3. Download the output files upon completion of the job.
Xtext grammar for WDL
<table>
<thead>
<tr>
<th>No</th>
<th>Service</th>
<th>Util/Script</th>
<th>Inputs (type)]</th>
<th>Outputs(type)]</th>
<th>LOC</th>
<th>Total LOC (files)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>echoString</td>
<td>echo</td>
<td>[0][1 (doc)]</td>
<td></td>
<td>10+13+1+6</td>
<td>30(4)</td>
</tr>
<tr>
<td>2</td>
<td>echoFile</td>
<td>echo</td>
<td>[1 (String)]</td>
<td>[1 (doc)]</td>
<td>10+14+1+6</td>
<td>31(4)</td>
</tr>
<tr>
<td>3</td>
<td>aggregate</td>
<td>cat</td>
<td>[1(List doc)]</td>
<td>[1 (doc)]</td>
<td>10+20+1+7</td>
<td>38(4)</td>
</tr>
<tr>
<td>4</td>
<td>classifier_Training</td>
<td>R</td>
<td>[2 (doc), 1 (String)]</td>
<td>[1 (doc)]</td>
<td>11+24+2+8</td>
<td>45(4)</td>
</tr>
<tr>
<td>5</td>
<td>classifier_Testing</td>
<td>R</td>
<td>[3 (doc), 1 (String)]</td>
<td>[1 (doc)]</td>
<td>12+29+2+8</td>
<td>51(4)</td>
</tr>
<tr>
<td>6</td>
<td>accuracy</td>
<td>R</td>
<td>[1 (doc)]</td>
<td>[1 (doc)]</td>
<td>11+19+1+6</td>
<td>37(4)</td>
</tr>
<tr>
<td>7</td>
<td>submitBlast</td>
<td>SLURM, sh</td>
<td>[3 (doc)]</td>
<td>[2 (String)]</td>
<td>17+27+2+8+18</td>
<td>72(5)</td>
</tr>
<tr>
<td>8</td>
<td>jobStatus</td>
<td>SLURM, sh</td>
<td>[1 (String)]</td>
<td>[1 (String)]</td>
<td>10+14+1+6</td>
<td>31(4)</td>
</tr>
<tr>
<td>9</td>
<td>blastResult</td>
<td>cp</td>
<td>[1 (String)]</td>
<td>[1 (doc)]</td>
<td>10+14+1+6</td>
<td>31(4)</td>
</tr>
<tr>
<td>10</td>
<td>mafft</td>
<td>mafft</td>
<td>[1 (doc)]</td>
<td>[1 (doc)]</td>
<td>10+18+1+6</td>
<td>35(4)</td>
</tr>
</tbody>
</table>

An overview of SDL code generation
Taverna classification workflow