TQL: A Query Language to Support Traceability

Jonathan I. Maletic
Michael L. Collard
Artifact Representation

- Models are represented in XML
- Any structured/semi-structured type of model is supported

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source code</td>
<td>srcML [Maletic et al. 02]</td>
</tr>
<tr>
<td>UML design documents</td>
<td>classML …</td>
</tr>
<tr>
<td>Requirements</td>
<td>UseCaseML …</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Link Representation

- XPath forms address paths between elements of models

- Links are stored external to models
  - Original model is preserved
  - Multiple views are possible
Traceability Link Model

- **link**
  - id : Xml:id
  - title : string
  - description : string
  - validationDateTimeStamp : Date
  - stabilityLevel : int
  - priority : int

- **metadata**
  - creationMode
  - linkRationale
  - authorInfo
  - createDateTimeStamp
  - lastmodifiedDateTimeStamp

- **locator**
  - base : metamodel_path
  - href : xpath_expression
  - label : string

- **relationship**
  - from : label
  - to : label
  - role : linkType
  - description : string
  - isDirected : bool

- **linkType**
  - name
  - rationale
  - category
Link Storage and Addressing
Sample Link Record

<table>
<thead>
<tr>
<th>id</th>
<th>myex1</th>
</tr>
</thead>
<tbody>
<tr>
<td>stability</td>
<td>1</td>
</tr>
<tr>
<td>priority</td>
<td>1</td>
</tr>
<tr>
<td>valid-date</td>
<td>1/14/2007 12:43:20 EDT</td>
</tr>
<tr>
<td>title</td>
<td>method name must match</td>
</tr>
<tr>
<td>metadata</td>
<td>ex1.dcmi.xml</td>
</tr>
<tr>
<td>from-locator</td>
<td>Mailbox::Validate_Password</td>
</tr>
<tr>
<td>to-locator</td>
<td>Mailbox::Validate_Password</td>
</tr>
<tr>
<td>role</td>
<td>must-agree</td>
</tr>
</tbody>
</table>

```
//class[@name='Mailbox']/
  method[@name='Validate_Password']
  <class>
    <method name="Validate_Password">
      <return type="int" />
    </method>
  </class>

//function [name='Mailbox::Validate_Password']
  <function>
    <type>int</type>
    <name>Mailbox::Validate_Password</name>
    <formal_params>...</formal_params>
  </function>

//function_decl [name='Validate_Password']
  <function_decl>
    <type>int</type>
    <name>Validate_Password</name>
    <formal_params>...</formal_params>;
  </function_decl>
```
Traceability Graph
Artifact Primitives

- `art:UseCase()`
- `art:Requirement()`
- `art:Design()`
- `art:Code()`
- `art:TestCase()`
- `art:BugReport()`
Traceability Primitives

- **tql:traceTo(source, sink)**
  - Returns all the artifacts in the traceability graph in the set source that trace directly or indirectly to an artifact in the set sink.

- **tql:traceFrom(sink, source)**
  - Returns all the artifacts in the traceability graph in the set sink that trace directly or indirectly from an artifact in the set source.

- Computes transitive closure
Link Primitives

- `tql:link(source, sink)`
  - Transitive closure of all links between the two sets of artifacts

- `tql:directlink(source, sink)`
  - Just directly linked artifacts

- `tql:link(artifact)`
  - All links to or from an artifact

- `tql:artifact(links)`
  - All artifacts involved in a given set of links
Are all requirements covered by a test case?

\[
\text{set: difference (}
\text{Requirement(),}
\text{traceFrom(Requirement(),}
\text{TestCase())}
\]
Are all non-functional requirements addressed by one or more parts of the implementation?

tql:traceTo(
    Requirement()[NFR()],
    Code())

tql:traceTo(
    Requirement()[NFR()],
    tql:traceTo(TestCase()
        Code()))
Which parts of code covered by a requirement have documented pre and post condition?

tql:traceFrom(
    Requirement(R),
    Code()//src:function
        [@requires | @ensures])
Are any test cases missing?

\[
\text{set}:\text{difference}(\text{Requirement}(), \text{tql}:\text{traceTo}(\text{Requirement}(), \text{_TestCase}())
\]
What is the impact of changing a requirement on the safety of the system?

tql:impactAnalysis(
    tql:traceFrom(
        Code(),
        Tql:traceFrom(
            Design(),
            Requirement(R))))
Related Work

- Constraint based mechanism to indicate when artifacts are inconsistent with one another [Reiss 2002, 2005, 2006]

- Consistency Checkers - xLinkit [Nentwich et al. 2002]

- Policy centric approach - ArchTrace [Murta et al. 2006]
Conclusions & Future

- Working on implementation – release as a tool
- Working on other primitives
- Applying to commercial software system

- This work is funded in part by the National Science Foundation under NSF grant CCF 08-11-21.