A Case Study for Evaluating Feature Location Techniques

Meghan Revelle

Nov. 20, 2008
How can we locate features?

- **Textual similarity**
  
  - $\cos(a) = \text{sim(text A, text B)}$

- **Dynamically**

- **Statically**

- **Combinations of the three**
  - IR + static
  - IR + dynamic
  - Static + dynamic
  - IR + dynamic + static
  - ...
Evaluation Approaches

- Precision and recall
- Position of first relevant method on ranked list
- Compare with methods in a patch
- Programmer agreement
  - This is where you come into the picture
Example

- System
  - Eclipse

- Feature
  - Add files and folders to UnifiedTree
# Classifying Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Relevant</th>
<th>Somewhat Relevant</th>
<th>Not Relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.eclipse.core.internal.localstore.UnifiedTree.addChildrenFromFileSystem</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.dtree.AbstractDataTreeNode.childAtOrNull</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.localstore.UnifiedTreeNode.reuse</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.localstore.UnifiedTree.createNode</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.localstore.UnifiedTreeNode.getLocalLocation</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.localstore.UnifiedTree.addChildrenToFileSystem</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.localstore.UnifiedTreeNode.reuse</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.localstore.RefreshLocalVisitor.visit</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.localstore.RefreshLocalVisitor.synchronizeExistence</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.localstore.UnifiedTree.addRootToQueue</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.dtree.DeltaDataTree.getChildNodes</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.internal.localstore.UnifiedTreeNode.existsInFileSystem</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Is this method relevant?

```java
/**
 * Factory method for creating a node for this tree.
 */
protected UnifiedTreeNode createNode(IResource resource, long stat, String loc)
{
    // first check for reusable objects
    UnifiedTreeNode node = null;
    int size = freeNodes.size();
    if (size > 0) {
        node = (UnifiedTreeNode) freeNodes.remove(size-1);
        node.reuse(this, resource, stat, localLocation, localName, existsWork);
        return node;
    }
    // none available, so create a new one
    return new UnifiedTreeNode(this, resource, stat, localLocation, localName,
```
Is this method relevant?

```java
/**
 * Reuses this object by assigning all new values for the fields.
 */
public void reuse(UnifiedTree tree, IResource resource, long stat, String local)
    this.tree = tree;
    this.child = null;
    this.resource = resource;
    this.stat = stat;
    this.existsWorkspace = existsWorkspace;
    this.localLocation = localLocation;
    this.localName = localName;
```
Is this method relevant?

```java
/**
 * Returns the child nodes of a node in the tree.
 */

protected AbstractDataTreeNode[] getChildNodes(IPath parentKey) {

    /* Algorithm:
     * for each delta in chain (going backwards),
     * get list of child nodes, if any in delta
     * assemble with previously seen list, if any
     * break when complete tree found,
     * report error if parent is missing or has been deleted
     */

    AbstractDataTreeNode[] childNodes = null;
    int keyLength = parentKey.segmentCount();
    for (DeltaDataTree tree = this; tree != null; tree = tree.parent) {
        AbstractDataTreeNode node = tree.rootNode;
        boolean complete = !node.isDelta();
        for (int i = 0; i < keyLength; i++) {
            node = node.childAtOrEmpty(parentKey.segment(i));
            if (node == null) {
                break;
            }
        }
        if (!node.isDelta()) {
```
We need your help

- Evaluate our feature location technique against others
- Classify methods from 10 lists
- System
  - jEdit
- Feature
  - Configurable thick cursor option
jEdit in action...
Thick Caret Option
Can you spot the difference?
In Summary

- **Task**
  - Determine relevance of methods to a feature

- **Time commitment**
  - 1-2 hours

- **Incentive**
  - Extra points
  - Cookies and brownies