AN EXPLORATORY INVESTIGATION ON INVASIVENESS OF SCIENTIFIC MODELING FRAMEWORKS

April 2009

Wes J. Lloyd
Framework Invasiveness

- Coupling between application code and framework code
  - Use of framework functions/methods
  - Use of framework specific data types
  - Implementation of framework interfaces
  - Extension of framework classes
  - Import/Include of framework libraries
Framework Invasiveness

- We presume that application code coupled to framework code is more difficult to:
  - Understand
  - Maintain
    - Upgrade framework versions
    - Bug defects / Feature Enhancements
  - Port to other frameworks
  - Reuse outside the framework
Research Question

- How does framework to application invasiveness impact Software Quality?
- Software Quality in terms of:
  - Maintainability
  - Understandability
  - Portability
  - Reusability
Why measure invasiveness?

- Quantify the burden for the framework user
- To evaluate framework design tradeoffs and new technologies

- **Heavy weight frameworks**
  - Framework overloads native language datatypes
  - Large APIs
  - Many imports

- **Light weight frameworks**
  - Smaller APIs
  - Native language datatypes
  - Dependency Injection
  - Inversion of Control design pattern
  - Annotations/POJOs
Measuring Invasiveness

- Are Object Oriented Coupling Measures useful?
  - Coupling Between Object Classes (CBO)
  - Efferent Coupling / Fan Out
  - Afferent Coupling / Fan In
  - Response for a Class (RFC)
  - Message Passage Coupling (MPC)
Measuring Invasiveness

- OO Coupling measures, measure coupling between all classes in a system

- “Invasiveness measures” needed
  - We desire to measure coupling between only application and framework classes
Invasiveness Metric: FDT

Framework Data Types

- **Used (FDT-Used)**
  - Raw count
  - Per 1000 LOC (kloc)
  - As a % of all data types used

- **Uses (FDT-Uses)**
  - Raw count
  - Per 1000 LOC (kloc)
  - As a % of all data types used
Invasiveness Metric: FF

Framework Functions

- **Used (FF-Used)**
  - Raw count
  - Per 1000 LOC (kloc)
  - As a % of all data types used

- **Uses (FF-Uses)**
  - Raw count
  - Per 1000 LOC (kloc)
  - As a % of all data types used
Invasiveness Metric: FDLOC

Framework Dependent Lines of Code

- Any line of code which would not compile if the framework were removed (FDLOC)
  - Raw count
  - As a % of all LOC

- Boilerplate code
  - Tempting to measure due its undesirability
  - Hard to define precisely in order to count
Other Measures

- Framework Interfaces
  - Used/Uses
- Framework Classes
  - Extended/Extensions
- Framework library include/imports
  - Used/Uses
- Non-framework library include/imports
  - Used/Uses
Evaluation of Measures

- How are invasiveness measures related?
- Are they unique measurements?
- How do invasiveness measures relate to:
  - Application Size (LOC)
  - Application Complexity
  - Object Oriented Coupling Measures
Empirical Study

- Domain: Scientific Modeling Frameworks
- Scientific Modeling Frameworks
  - Support aggregation of models into classes (components)
  - Component interaction/communication
  - Time/spatial data looping
  - Regridding arrays and spatial data
  - Multithreading/multiprocessor support
  - Cross language interoperability
Scientific Modeling Frameworks

- CCA 0.6.6: Common Component Architecture - Java
- ESMF-C/Fortran 3.1.1: Earth Science Modeling Framework
- OpenMI 1.4: Open Modeling Interface - Java
- OMS 2.2: Object Modeling System - Java
- OMS 3.0: Object Modeling System - Java
# Modeling Frameworks

<table>
<thead>
<tr>
<th>Framework</th>
<th>size (LOC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESMF 3.1.1 C</td>
<td>268146</td>
</tr>
<tr>
<td>ESMF 3.1.1 Fortran</td>
<td>268146</td>
</tr>
<tr>
<td>CCA 0.6.6</td>
<td>128286</td>
</tr>
<tr>
<td>OpenMI 1.4</td>
<td>6489</td>
</tr>
<tr>
<td>OMS 3.0</td>
<td>2983</td>
</tr>
<tr>
<td>OMS 2.2</td>
<td>376749</td>
</tr>
</tbody>
</table>
Modeling Application: Thornthwaite

- Thornthwaite Water balance model
  - Models allocation of water among components of hydrological system

- Model
  - 8 Components
    - Climate, Daylen, HamonET, Snow, Soil moisture, Runoff, Output, Controller

- FORTRAN Implementation = 244 LOC
Modeling Application: Thornthwaite

- All implementations produce identical numeric output
- No language specific output formatting

- Source code repository:
  - http://svn.javaforge.com/svn/invasive/trunk/
Analysis Tools

- **SLOCCOUNT**
  - LOC for FORTRAN, C, C++, Java
- **Understand 2.0 Analyst**
  - metrics for FORTRAN, C, C++, Java
- **Custom tool**
  -Parsed Understand 2.0 function and data type usage reports to provide data for FF and FDT usage measures
# Model implementations

<table>
<thead>
<tr>
<th></th>
<th>Total LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN</td>
<td>244</td>
</tr>
<tr>
<td>OMS 3.0 *</td>
<td>295</td>
</tr>
<tr>
<td>C++</td>
<td>405</td>
</tr>
<tr>
<td>OMS 2.2 *</td>
<td>450</td>
</tr>
<tr>
<td>ESMF 3.1.1 C</td>
<td>583</td>
</tr>
<tr>
<td>ESMF 3.1.1 Fortran *</td>
<td>683</td>
</tr>
<tr>
<td>OpenMI 1.4 *</td>
<td>880</td>
</tr>
<tr>
<td>CCA 0.6.6 user java</td>
<td>1635</td>
</tr>
<tr>
<td>CCA 0.6.6 java only</td>
<td>9914</td>
</tr>
<tr>
<td>CCA 0.6.6</td>
<td>62809</td>
</tr>
</tbody>
</table>

* Code checked by framework developer/collaborator
Framework Dependent Code

<table>
<thead>
<tr>
<th>Framework</th>
<th>% FDLOC</th>
<th>FDLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMS 3.0</td>
<td>14.84%</td>
<td>44</td>
</tr>
<tr>
<td>ESMF 3.1.1 C</td>
<td>30.85%</td>
<td>178</td>
</tr>
<tr>
<td>CCA 0.6.6 User Java</td>
<td>32.60%</td>
<td>533</td>
</tr>
<tr>
<td>OMS 2.2</td>
<td>32.67%</td>
<td>147</td>
</tr>
<tr>
<td>OpenMI 1.4</td>
<td>38.41%</td>
<td>338</td>
</tr>
<tr>
<td>ESMF 3.1.1 Fortran</td>
<td>41.42%</td>
<td>280</td>
</tr>
</tbody>
</table>
## Framework Data Types Used

<table>
<thead>
<tr>
<th>Framework</th>
<th>FDT Used</th>
<th>% FDT Used</th>
<th>FDT Ref/KLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMS 3.0</td>
<td>1</td>
<td>4.76%</td>
<td>3.39</td>
</tr>
<tr>
<td>ESMF 3.1.1 Fortran</td>
<td>3</td>
<td>27.27%</td>
<td>4.39</td>
</tr>
<tr>
<td>OMS 2.2</td>
<td>5</td>
<td>41.67%</td>
<td>11.11</td>
</tr>
<tr>
<td>OpenMI 1.4</td>
<td>8</td>
<td>23.53%</td>
<td>9.09</td>
</tr>
<tr>
<td>ESMF 3.1.1 C</td>
<td>10</td>
<td>30.30%</td>
<td>17.15</td>
</tr>
<tr>
<td>CCA 0.6.6 User Java</td>
<td>15</td>
<td>46.88%</td>
<td>9.17</td>
</tr>
</tbody>
</table>
## Framework Data Type Uses

<table>
<thead>
<tr>
<th>Framework</th>
<th>FDT Uses</th>
<th>% FDT Uses</th>
<th>FDT Refs/KLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMS 3.0</td>
<td>1</td>
<td>1.35%</td>
<td>3.39</td>
</tr>
<tr>
<td>OMS 2.2</td>
<td>72</td>
<td>64.29%</td>
<td>160.00</td>
</tr>
<tr>
<td>OpenMI 1.4</td>
<td>73</td>
<td>32.30%</td>
<td>82.95</td>
</tr>
<tr>
<td>ESMF 3.1.1 Fortran</td>
<td>109</td>
<td>51.90%</td>
<td>159.59</td>
</tr>
<tr>
<td>ESMF 3.1.1 C</td>
<td>122</td>
<td>49.59%</td>
<td>209.26</td>
</tr>
<tr>
<td>CCA 0.6.6 User Java</td>
<td>135</td>
<td>49.82%</td>
<td>82.57</td>
</tr>
</tbody>
</table>
## Framework Functions Used

<table>
<thead>
<tr>
<th>Framework</th>
<th>FF Used</th>
<th>% FF Used</th>
<th>FF Used/KLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMS 2.2</td>
<td>7</td>
<td>50.00%</td>
<td>15.56</td>
</tr>
<tr>
<td>OMS 3.0</td>
<td>8</td>
<td>26.67%</td>
<td>27.12</td>
</tr>
<tr>
<td>ESMF 3.1.1 Fortran</td>
<td>11</td>
<td>78.57%</td>
<td>16.11</td>
</tr>
<tr>
<td>ESMF 3.1.1 C</td>
<td>13</td>
<td>46.43%</td>
<td>22.30</td>
</tr>
<tr>
<td>OpenMI 1.4</td>
<td>20</td>
<td>37.74%</td>
<td>22.73</td>
</tr>
<tr>
<td>CCA 0.6.6 User Java</td>
<td>48</td>
<td>70.59%</td>
<td>29.36</td>
</tr>
</tbody>
</table>
## Framework Function Uses

<table>
<thead>
<tr>
<th>Framework</th>
<th>FF Uses</th>
<th>% FF Uses</th>
<th>FF Uses/KLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMS 3.0</td>
<td>21</td>
<td>40.38%</td>
<td>71.19</td>
</tr>
<tr>
<td>OMS 2.2</td>
<td>33</td>
<td>73.33%</td>
<td>73.33</td>
</tr>
<tr>
<td>ESMF 3.1.1 C</td>
<td>77</td>
<td>76.24%</td>
<td>132.08</td>
</tr>
<tr>
<td>ESMF 3.1.1 Fortran</td>
<td>148</td>
<td>96.10%</td>
<td>216.69</td>
</tr>
<tr>
<td>CCA 0.6.6 User Java</td>
<td>215</td>
<td>69.58%</td>
<td>131.50</td>
</tr>
<tr>
<td>OpenMI 1.4</td>
<td>280</td>
<td>79.10%</td>
<td>318.18</td>
</tr>
</tbody>
</table>
Invasiveness Measures

- Combine measures to generate an overall invasiveness ranking
  - Invasiveness 1: raw counts
    - FDLOC, FDT Used, FDT Uses, FF Used, FF Uses
  - Invasiveness 2: framework usage density
    - Framework to non-framework data type and function usage
      - \%FDLOC, \%FDT Used, \%FDT Uses, \%FF Used, \%FF Uses
  - Invasiveness 3: code density
    - Framework data type and function usage per kloc
      - FDLOC/kloc, FDT Used/kloc, FDT Uses/kloc, FF Used/kloc, FF Uses/kloc
For ranking Invasiveness

\[
\frac{(\text{FDT Used/kloc} + \text{FDT Uses/kloc})}{2} + 
\frac{(\text{FF Used/kloc} + \text{FF Uses/kloc})}{2} + 
\% \text{ FDLOC}
\]

- To generate invasiveness:
  - Calculate averages, standard deviations for each metric, for each model implementation
  - Use the number of standard deviations away from average in place of metric value
  - Sum (or average) the standard deviations
  - Larger values indicate more invasive implementations when compared with others in the set
## Invasiveness rankings

<table>
<thead>
<tr>
<th></th>
<th>Inv 1</th>
<th>Inv 2</th>
<th>Inv 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMS 3.0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OMS 2.2</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>ESMF 3.1.1 C</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>OpenMI 1.4</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>CCA 0.6.6</td>
<td>6</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>ESMF 3.1.1 Fortran</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>
Invasiveness Measure Independence

- **H0:** There is no relationship between invasiveness measures
- 120 possible relationships
  - 8 significant (pearson)
    - multiple r>.811, df=4, p<.05
  - 9 significant (spearman rank)
    - rho>.811, df=4, p<.05
- Random chance would be 6 (5%)
Invasiveness Measure Dependence

- Only six relations cross measure categories:
  - Both Pearson & Spearman Rank
    - FDLOC -> FF Used
    - FDLOC -> FF Uses
    - %FDLOC -> %FF Uses
    - %FF Uses -> FDLOC/kloc
  - Pearson
    - FDT Used -> FF Used
  - Spearman Rank
    - %FDT Uses -> %FF Used
Invasiveness and size (LOC)

- **H₀**: There is no relationship between invasiveness measures and LOC

<table>
<thead>
<tr>
<th></th>
<th>Total LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inv 1</td>
<td>0.837</td>
</tr>
<tr>
<td>Inv 2</td>
<td>0.456</td>
</tr>
<tr>
<td>Inv 3</td>
<td>0.460</td>
</tr>
</tbody>
</table>

- Significant correlations
  - FDLOC, FDT Used, FF Used  -> LOC

- Raw values seem to correlate with LOC, others do not.
Invasiveness and Complexity

- **$H_0$:** There is no relationship between invasiveness measures and Cyclomatic Complexity

<table>
<thead>
<tr>
<th></th>
<th>Avg CC/method</th>
<th>Total CC</th>
<th>CC/kloc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inv 1</td>
<td>-0.182</td>
<td>0.715</td>
<td>0.109</td>
</tr>
<tr>
<td>Inv 2</td>
<td>-0.601</td>
<td>0.394</td>
<td>0.221</td>
</tr>
<tr>
<td>Inv 3</td>
<td>-0.423</td>
<td>0.317</td>
<td>0.000</td>
</tr>
</tbody>
</table>

- Significant correlations
  - FF Used/KLOC -> Avg CC/method
  - FDLOC, FDT Used, FF Used -> Total CC

- **Cyclomatic complexity and invasiveness do not appear to be related**
Invasiveness and Coupling

- **H₀**: There is no relationship between application to framework coupling measures and Object Oriented Coupling Measures (CBO, Fan-In, Fan-Out)

- Significant correlations
  - FDLOC, FDT Used, %FF Used -> Total Fan In
  - FDLOC, FDT Used -> Total Fan Out

- CBO: only two systems had a measured value

- Raw values seem to correlate with total fan-in/fan-out coupling, others do not.

<table>
<thead>
<tr>
<th>Fan In/method</th>
<th>Total Fan In</th>
<th>Fan Out/method</th>
<th>Total Fan Out</th>
<th>Fan In/kloc</th>
<th>Fan out/kloc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inv 1</td>
<td>-0.523</td>
<td><strong>0.928</strong></td>
<td>-0.385</td>
<td><strong>0.938</strong></td>
<td>0.202</td>
</tr>
<tr>
<td>Inv 2</td>
<td>-0.670</td>
<td>0.850</td>
<td>-0.627</td>
<td>0.573</td>
<td>0.566</td>
</tr>
<tr>
<td>Inv 3</td>
<td>-0.307</td>
<td>0.650</td>
<td>-0.165</td>
<td>0.771</td>
<td>0.426</td>
</tr>
</tbody>
</table>
Invasiveness and Indirect Measures of Software Quality

- Chidamber and Kemerer OO metrics
  - WMC: Weighted methods per class
  - CBO: Coupling between object classes
  - RFC: Response for a class
  - LCOM: Lack of cohesion in methods

- Metrics only collected for OO systems
  - n=4, df=2, very limited sample size!
Invasiveness and Indirect Measures of Software Quality

- **H₀**: The quantity of application to framework coupling/invasiveness is not related to indirect measurements of software quality.
  - df=2 requires multiple $r \geq .950$!

<table>
<thead>
<tr>
<th></th>
<th>Avg CBO/class</th>
<th>Avg WMC/class</th>
<th>Total WMC</th>
<th>Avg RFC/class</th>
<th>Total RFC</th>
<th>Avg LCOM/class</th>
<th>WMC/kloc</th>
<th>RFC/kloc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inv 1</td>
<td>-0.379</td>
<td>0.576</td>
<td>0.886</td>
<td>0.576</td>
<td>0.886</td>
<td>0.547</td>
<td>0.521</td>
<td>0.521</td>
</tr>
<tr>
<td>Inv 2</td>
<td>-0.777</td>
<td>0.855</td>
<td><strong>0.955</strong></td>
<td>0.855</td>
<td><strong>0.955</strong></td>
<td>0.843</td>
<td>0.395</td>
<td>0.395</td>
</tr>
<tr>
<td>Inv 3</td>
<td>-0.211</td>
<td>0.378</td>
<td>0.750</td>
<td>0.378</td>
<td>0.750</td>
<td>0.349</td>
<td>0.820</td>
<td>0.820</td>
</tr>
</tbody>
</table>
Experiment miscellaneous details

- **CCA implementation:** an outlier
  - Large quantity of automatically generated boilerplate code
  - To normalize size ignored files not touched by developer
    - Reduces LOC from 62809 (all) to 9914 (java only), to 1635 (user java only)

- **NGMF**
  - Did not count lines of code with just a Java annotation as a framework dependent line of code (FDLOC)
    - Treating annotations like comments
    - Annotations not used in other frameworks
    - Annotations are easily removed/ignored

- **ESMF C**
  - Implementation used global data due to incomplete framework support

- Model functions not counted as framework functions
- Model datatypes not counted as framework datatypes
- Fortran datatype usage counted manually without tool support
External threats to validity

- Thornthwaite is a simple model that does not fully exercise all framework features
- Model developer was new to developing in frameworks. Two implementations were already coded.
- Implementation languages varied, therefore metrics collection techniques varied
- Not all implementations were in an OO language
Other Limitations

- Limited experimental power
  - Framework implementations (n=6, df=4)
- Some measures were assessed manually without tool support
Summary

- Unique comparison study performed
  - Thornthwaite scientific model implemented 8 times in 5 frameworks, 4 languages

- Invasiveness metrics proposed, applied, and evaluated
  - Measures used to rank framework-based implementations
  - Measures compared with existing software metrics: LOC, complexity, OO coupling
  - Measures compared with indirect measures of software quality
Conclusions

- Individual invasiveness measures correlated significantly with each other only slightly more than random chance.
- Raw invasiveness metric values did correlate somewhat with application size (LOC).
- There was no significant relationship between invasiveness and complexity.
- Raw invasiveness metric values did correlate somewhat with total fan-in / total fan-out coupling.