April 2009

Wes J. Lloyd

AN EXPLORATORY INVESTIGATION ON INVASIVENESS OF SCIENTIFIC MODELING FRAMEWORKS

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?

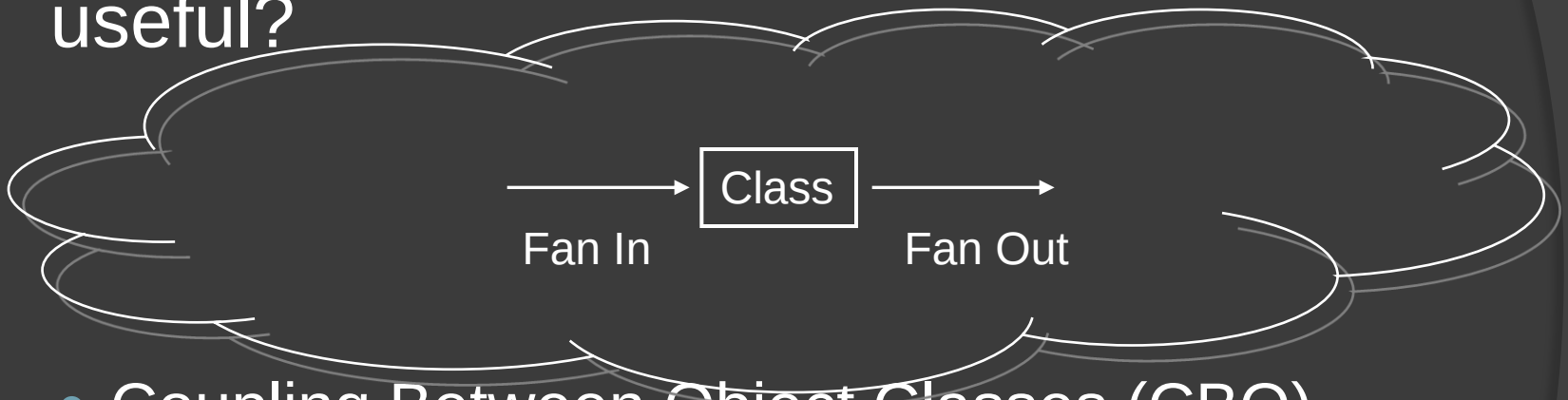
- Quantify the burden for the framework user
- To evaluate framework design tradeoffs and new technologies

  - 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

Framework	**size (LOC)**
ESMF 3.1.1 C	268146
ESMF 3.1.1 Fortran	268146
CCA 0.6.6	128286
OpenMI 1.4	6489
OMS 3.0	2983
OMS 2.2	376749

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

	Total LOC
FORTRAN	244
OMS 3.0 *	295
C++	405
OMS 2.2 *	450
ESMF 3.1.1 C	583
ESMF 3.1.1 Fortran *	683
OpenMI 1.4 *	880
CCA 0.6.6 user java	1635
CCA 0.6.6 java only	9914
CCA 0.6.6	62809

* Code checked by framework developer/collaborator

Framework Dependent Code

	% FDLOC	FDLOC
OMS 3.0	14.84%	44
ESMF 3.1.1 C	30.85%	178
CCA 0.6.6 User Java	32.60%	533
OMS 2.2	32.67%	147
OpenMI 1.4	38.41%	338
ESMF 3.1.1 Fortran	41.42%	280

Framework Data Types Used

	FDT Used	% FDT Used	FDT Ref/KLOC
OMS 3.0	1	4.76%	3.39
ESMF 3.1.1 Fortran	3	27.27%	4.39
OMS 2.2	5	41.67%	11.11
OpenMI 1.4	8	23.53%	9.09
ESMF 3.1.1 C	10	30.30%	17.15
CCA 0.6.6 User Java	15	46.88%	9.17

Framework Data Type Uses

	FDT Uses	% FDT Uses	FDT Refs/KLOC
OMS 3.0	1	1.35%	3.39
OMS 2.2	72	64.29%	160.00
OpenMI 1.4	73	32.30%	82.95
ESMF 3.1.1 Fortran	109	51.90%	159.59
ESMF 3.1.1 C	122	49.59%	209.26
CCA 0.6.6 User Java	135	49.82%	82.57

Framework Functions Used

	FF Used	% FF Used	FF Used/KLOC
OMS 2.2	7	50.00%	15.56
OMS 3.0	8	26.67%	27.12
ESMF 3.1.1 Fortran	11	78.57%	16.11
ESMF 3.1.1 C	13	46.43%	22.30
OpenMI 1.4	20	37.74%	22.73
CCA 0.6.6 User Java	48	70.59%	29.36

Framework Function Uses

	FF Uses	% FF Uses	FF Uses/KLOC
OMS 3.0	21	40.38%	71.19
OMS 2.2	33	73.33%	73.33
ESMF 3.1.1 C	77	76.24%	132.08
ESMF 3.1.1 Fortran	148	96.10%	216.69
CCA 0.6.6 User Java	215	69.58%	131.50
OpenMI 1.4	280	79.10%	318.18

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

(FDT Used/kloc + FDT Uses/kloc)/2 +
(FF Used/kloc + FF Uses/kloc)/2 +
% 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

	Inv 1	Inv 2	Inv 3
OMS 3.0	1	1	1
OMS 2.2	2	4	2
ESMF 3.1.1 C	3	3	5
OpenMI 1.4	5	2	6
CCA 0.6.6	6	5	3
ESMF 3.1.1 Fortran	4	6	4

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%)

	FDLOC	*FDT Used*	*FDT Uses*	*FF Used*	*FF Uses*	*% FDLOC*	*% FDT Used*	*% FDT Uses*	*% FF Used*	*% FF Uses*	*FDT Used/kloc*	*FDT Uses/kloc*	*FF Used/kloc*	*FF Uses/kloc*	*FDLOC/kloc*
FDLOC	1.000														
FDT Used	0.771	1.000													
FDT Uses	0.649	0.753	1.000												
FF Used	**0.932**	**0.854**	0.560	1.000											
FF Uses	**0.823**	0.545	0.448	0.639	1.000										
% FDLOC	0.494	0.305	0.700	0.188	0.634	1.000									
% FDT Used	0.573	0.709	0.786	0.552	0.261	0.558	1.000								
% FDT Uses	0.281	0.416	0.770	0.156	0.108	0.722	**0.883**	1.000							
% FF Used	0.592	0.334	0.772	0.425	0.310	0.672	0.648	0.683	1.000						
% FF Uses	0.341	0.196	0.716	0.038	0.463	**0.965**	0.492	0.743	0.710	1.000					
FDT Used/kloc	-0.012	0.582	0.532	0.077	-0.031	0.199	0.498	0.530	-0.057	0.229	1.000				
FDT Uses/kloc	-0.093	0.208	0.673	-0.209	-0.102	0.595	0.530	**0.823**	0.447	0.729	0.693	1.000			
FF Used/kloc	0.411	0.464	-0.087	0.650	0.234	-0.536	-0.163	-0.574	-0.269	-0.636	-0.082	-0.655	1.000		
FF Uses/kloc	0.424	0.134	0.242	0.142	**0.847**	0.690	-0.064	0.025	0.138	0.594	-0.069	0.051	-0.124	1.000	
FDLOC/kloc	0.494	0.305	0.700	0.188	0.634	**1.000**	0.558	0.722	0.672	**0.965**	0.199	0.595	-0.536	0.690	1.000

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)

- **H_0:** There is no relationship between invasiveness measures and LOC

	Total LOC
Inv 1	**0.837**
Inv 2	0.456
Inv 3	0.460

- 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

	Avg CC/method	*Total CC*	*CC/kloc*
Inv 1	-0.182	0.715	0.109
Inv 2	-0.601	0.394	0.221
Inv 3	-0.423	0.317	0.000

- 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

- **H_0:** There is no relationship between application to framework coupling measures and Object Oriented Coupling Measures (CBO, Fan-In, Fan-Out)

	Fan In/method	*Total Fan In*	*Fan Out/method*	*Total Fan Out*	*Fan In/kloc*	*Fan out/kloc*
Inv 1	-0.523	**0.928**	-0.385	**0.938**	0.202	0.451
Inv 2	-0.670	0.850	-0.627	0.573	0.566	0.375
Inv 3	-0.307	0.650	-0.165	0.771	0.426	0.775

- 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.**

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

- **H_0:** The quantity of application to framework coupling/invasiveness is not related to indirect measurements of software quality.
  - df=2 requires multiple r>=.950!

	Avg CBO/class	*Avg WMC/class*	*Total WMC*	*Avg RFC/class*	*Total RFC*	*Avg LCOM/class*	*WMC/kloc*	*RFC/kloc*
Inv 1	-0.379	0.576	0.886	0.576	0.886	0.547	0.521	0.521
Inv 2	-0.777	0.855	**0.955**	0.855	**0.955**	0.843	0.395	0.395
Inv 3	-0.211	0.378	0.750	0.378	0.750	0.349	0.820	0.820

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