Analysing Structural Dependencies in Software Product Lines

Using Distribution and Network Statistics

by Sven Apel, Sergiy Kolnesikov, Stefan Sobernig;

in cooperation with Jörg Liebig and Norbert Siegmund

March 21, 2012
Structure of the talk

1. Motivation
2. Gathering syntactic reference data from Java-based SPLs (Fuji)
3. Notions of reference
4. Degree statistics over reference data

“Ufff! A false x-ray apparatus for SPLs?”
(Illustration adapted from Perelman (1913/2008): Physics for Entertainment, Fig. 95, p. 132)
Data collection for Fuji-driven SPLs

See [Kol11] for details on Fuji
A generic notion of reference

- In the Fuji/Java context, relevant DeclExpressions are ...
  - Class declarations
  - Constructor and initializer declarations
  - Method declarations
  - Field declarations

- Relevant UseExpressions are ...
  - Class-instance creation expressions
  - Field access expressions
  - Method invocation expressions
An extended notion of “Fuji reference”

\[ R : F \times A \times \Gamma \mapsto F \times A \times \Gamma \]

\[ = (((\{f_i, \alpha_m, \tau\}, \{f_j, \alpha_n, \tau\}), \ldots) \]

\[ 1 \leq i, j \leq |F|, i \neq j \]

\[ 1 \leq m, n \leq |A|, m \neq n \]

\[ \tau \in \Gamma : \{\text{type, field, method, constructor}\} \]

- Intra-module references, e.g.: aRef
  \( (\{\text{Base, B.bar(), method}\}, \{\text{Base, A.foo(), method}\}) \)

- Inter-module references, e.g.: bRef
  \( (\{\text{Base, B.bar(), method}\}, \{\text{F1, A.foo(), method}\}) \)
Available reference data sets (SPLs)

- References and introductions data of 28 SPLs
- **Feature modules**: $\Delta = 99$ (BerkeleyDB) vs. $\delta = 5$ (Raroscope)
- **Introductions**: $\Delta = 9379$ (BerkeleyDB) vs. $\delta = 43$ (EPL)
- **(Unique) references**: $\Delta = 52890$ (BerkeleyDB) vs. $\delta = 80$ (EPL)
- **(Unique) inter-module references**: $\Delta = 30061$ (BerkeleyDB) vs. $\delta = 51$ (Raroscope)
Fuji reference space

The reference data can be sliced by . . .

- . . . pairs of syntactic categories:

<table>
<thead>
<tr>
<th>source → target</th>
<th>field</th>
<th>method</th>
<th>constructor</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constructor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $4 \times 4 = 16$ subsets of reference data per SPL
- Column-wise aggregation: field accesses, method accesses

- . . . the level of abstraction: per-element vs. per-feature-module
- . . . the kind of reference: intra-module vs. inter-module
- . . . the reference direction: in- vs. outgoing
- . . . or any combination of the above!
Degrees in reference networks

- For each possible reference-data slice, e.g.:
  - method → method
  - per-element,
  - intra- and inter-module references . . .

- . . . we can form a graph $G_R$
- We can identify local properties of each node $n$ (e.g., element, feature module), such as:
  - out-degree $d^+(n)$
  - in-degree $d^-(n)$
  - . . . as well as their relative forms

- and global, degree-based characteristics
  - Average, minimum, and maximum degrees: $\phi d^+(G_R)$, $\phi d^-(G_R)$,
    $\Delta d^+(G_R)$, $\Delta d^-(G_R)$, $\delta d^+(G_R)$,
    $\delta d^-(G_R)$
  - (In-/Out-) Degree distribution
An exemplary out-degree distribution

For the given example, the global, out-degree characteristics are:

- \( \Delta d^+(G_R) = 3 \)
- \( \delta d^+(G_R) = 0 \)
- \( \phi d^+(G_R) = ? \)
- The distribution of out-degree frequencies in \( G_R \):
  
<table>
<thead>
<tr>
<th>out-degree</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>rel. frequency</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Example (method->method out-degrees)
Do degrees follow a particular distribution?

Random binominal distribution (p = 0.5)

Random Poisson distribution (lambda = 2)

Random geometric distribution (p = 0.5)
Towards Smell Detection, Ex. 1 (1)

A variant of a Feature Envy [Fow03]: Excessive Inter-Module Method Invocations

1. For a given SPL, choose the appropriate data slice: per-element, \*→method references, out-degrees, inter-module
2. Indicator measures: network density, **strength** distribution (skewness)

Possible envies?

```
| F | density = 17/30 ~ 0.57 |
| f: Base | τ: method |
| α: B.bar() |
| | d+ = 2 |
| | d+ = 3 |
| f: F1 | τ: method |
| α: A.foo() |
| | d+ = 2 |
| | d+ = 3 |
| f: F2 | τ: method |
| α: C.faz() |
| | d+ = 2 |
| | d+ = 3 |
| f: F3 | τ: method |
| α: D.daz() |
| | d+ = 2 |
| | d+ = 3 |
```

Mostly ego-centric?

```
| F | density = 5/30 = 0.3 |
| f: Base | τ: method |
| α: B.bar() |
| | d+ = 0 |
| | d+ = 1 |
| f: F1 | τ: method |
| α: A.foo() |
| | d+ = 0 |
| | d+ = 1 |
| f: F2 | τ: method |
| α: C.faz() |
| | d+ = 0 |
| | d+ = 1 |
| f: F3 | τ: method |
| α: D.daz() |
| | d+ = 0 |
| | d+ = 1 |
```

Left-skewed

```
| out-degree | rel. frequency |
| 0 | 0.1 |
| 1 | 0.2 |
| 2 | 0.3 |
| 3 | 0.4 |
| 4 | 0.5 |
```

Right-skewed

```
| out-degree | rel. frequency |
| 0 | 0.4 |
| 1 | 0.3 |
| 2 | 0.2 |
| 3 | 0.1 |
```
Can one observe method-level feature Envies in the TankWar Fuji-SPL?

TankWarAll (method-only source cardinalities)

- Positive skew (right-tailed), also heavy-tailed?
- Majority of self-sufficient (cohesive) elements
- Hubs
- Outliers

(Frequent?)
Research question(s): Are we looking at…

…variants of a *power-law* distribution for our SPLs and for all/certain slices?

Why would one expect to find such a degree distribution in a Fuji SPL?
On Power Laws (or the like)

- Numerous reports for important OO program structures:
  - Many hierarchical (ownership) and non-hierarchical (statements) relations within and between class boundaries; Qualitas Corpus [TSWW11]; also states the correspondence between overall connectivity and between-class connectivity.
  - Classes and class relations such as inheritance and aggregation; JDK class libraries [WC03]
  - Variable type annotations, object creation, and parameter passing; Squeak and VisualWorks Smalltalk [MPST04].
  - Global layout of 60 object graphs, extracted from object-object references from sources as diverse as C++ frontend to GCC, the Self runtime, and Java ArgoUML [PNFB05]

- Underlying forces (development processes and practices)
  - Preferential attachment in the development process
  - OO design patterns (e.g., template & hook methods, abstract class designs)
  - Key classes
Per-feature-module statistics

- Only inter-module reference are considered
- Per-element references become aggregated/subsumed by per-module references (edge weights)
Towards Smell Detection, Ex. 2

Direct Field Accesses vs. Feature-Module Boundaries [AKL⁺12]

1. For a given SPL, choose the appropriate data slice: per-feature, *→field references, out-degrees, inter-module
2. Indicator measures: network density, degree distribution (skewness)

Relatively *high* coupling

![Diagram of a graph with nodes and edges labeled with out-degrees (d+)]

- $d^+ = 2$
- $d^+ = 3$
- $d^+ = 4$

Density $= 17/30 \approx 0.57$

IFI = 6

Relatively *low* coupling

![Diagram of a simpler graph with fewer nodes and edges, labeled with out-degrees (d+)]

- $d^+ = 1$
- $d^+ = 0$
- $d^+ = 3$

Density $= 5/30 = 0.17$

IFI = 6

Left-skewed

![Bar chart showing left-skewed distribution of out-degrees]

Right-skewed

![Bar chart showing right-skewed distribution of out-degrees]
1. Short-term objectives (Stefan, Sven, Sergiy)
   - Additional slicing dimensions: Incorporate co-occurrence dependencies between feature modules (never, always, optional)
   - Usage frequencies of introductions
   - Distribution statistics for field and method accesses per feature module
   - Local/global characteristics of derivative features (Jörg)
   - Detection/calibration of intrinsic (statistical) thresholds for degree and distribution statistics
   - Input for Performance Prediction (see Norbert’s and Sergiy’s talks): Higher-Order (e.g., triangles) and Hot-Spot Features (hubs, betweenness)

2. Mid-term objectives
   - Detection/calibration of extrinsic thresholds, by incorporating the non-functional data profiles of the SPLs (Sergiy, Norbert, Jörg)
   - Weighted network statistics (Stefan, Sven, Jörg)
     - Edge weights based on reference counts (per-element, per-feature)
     - Attribute measures as edge weights (e.g., degrees of coupling and scattering, . . . )
     - . . .
“What do you see through the walls of SPL code data?”
(Illustration adapted from Perelman (1913/2008): Physics for Entertainment, Fig. 95, p. 132)
Appendix
Fuji-driven SPLs

See [Kol11] for details on Fuji
Feature-module composition

Fuji composes feature modules using AST superimposition: \( Base \bullet F1 \)

Each declaration element are identified by ...

- ...its introducing feature module \( f: Base, F1 \)
- ...its label \( \alpha: A, B, \ldots, \text{foo()} \)
- ...its syntactic category \( \tau: \text{type, method} \)
Exploratory issue: Per-feature-module statistics (1)

Reference data (showing source and target feature per reference)

Asymmetric feature-feature matrix

cross-tabulation