Investigating how code attributes affect the effort of developers performing different activities during software maintenance

Aiko Yamashita
CWI, Netherlands
Oslo and Akershus University College, Norway
2016-12-09
STRUCTURAL PROPERTIES AFFECT SOFTWARE

- Long Method
- Large Class

- Divergent Change
- Shotgun Surgery

https://sourcemaking.com/refactoring/smells
https://xkcd.com/844/
CODE SMELLS: ONE FORM OF STRUCTURAL ANOMALY

A hint about suboptimal implementation choices that can affect negatively future maintenance and evolution.
EXAMPLE OF **CODE SMELL AND REFACTORIZING**

**Shotgun Surgery**

A change leads to another change, to another, to another.

**Move method** refactoring

Reduce the coupling between components
State of Art on Code Smells

- Code smells have deterring effects on the introduction of defects

- Larger maintenance effort
  - Deligiannis (2004), Abbes (2011)

- Larger and more frequent changes in the code
  - Olbrich (2009), Khomh (2009)

- The overall capacity of code smell analysis to explain or predict maintenance problems or maintenance effort is rather modest
Previous work: Multiple, controlled case study (Yamashita 2012, Sjøberg et al., 2013)

Context

- 4 Java Applications
- Same functionality
- Different design/code
- Size: 7KLOC to 14KLOC

Study Design

Maintenance Tasks

Task 1. Replacing external data source

Task 2. New authentication mechanism

Task 3. New Reporting functionality
Previous work: Multiple, controlled case study (Yamashita 2012, Sjøberg et al., 2013)

Variables of interest

- Code smells (num. smells** smell density**)
- Maintainability perception*
- Maintenance problems**
- Change Size**
- Defects*
- Effort**
- Eclipse activity logs
- Think aloud

Data sources

- Source code
- Open interviews Audio files/notes
- Study diary
- Daily interviews Audio files/notes
- Subversion database
- Trac (Issue tracker), Acceptance test reports
- Think aloud Video files/notes
- Task progress sheets

** System and file level
* Only at system level

Paper by Sjøberg et al. (2013)

- 50,000 Euros
- Sep-Dec, 2008
- 7 Weeks
- 6 Developers
- 2 Companies
A model that includes file size and number of changes and code smells displayed a fit of $R^2 = 0.58$.

Removing the smells from that model did not decrease the fit!! ($R^2 = 0.58$)

Only smell that remained significant was Refused Bequest, which registered a decrease in effort ($\alpha < 0.01$).

File size and number of changes remain the most significant predictors of effort ($\alpha < 0.001$).
Analysis done in previous work (Sjøberg et al., 2013)

### Explanatory model for Effort

- **Dependent variable:** Effort (time)
- **Independent variables:** 12 smells
- **Control variables:**
  - File size (LOC)
  - Number of revisions on a file
  - System
  - Developer
  - Round

**Analysis:** Multiple Linear Regression

### Results

- A model that includes file size and number of changes and code smells displayed a fit of $R^2 = 0.58$
- Removing the smells from that model did not decrease the fit!! ($R^2 = 0.58$)
- Only smell that remained significant was Refused Bequest, which registered a decrease in effort ($\alpha < 0.01$)
- File size and number of changes remain the most significant predictors of effort ($\alpha < 0.001$)

Code smells are not better at explaining sheer-effort at file level, than size and number of revisions.
Previous analysis considers **sheer effort**

- Reading
- Programming (code-related) activities during Maintenance
- Searching
- Navigating
- Editing
- Others

**COMPILING!**
Procedure for extracting activity effort

- Selection of artifacts in the package explorer
- Selection of Java elements in the editor window
- Selecting Java elements in the file outline
- Editing source files (Java files)
- Scrolling the source code window
- Switching between open files
- Running Eclipse “commands” (copy, paste, go to line)

Activity logs
Procedure for extracting activity effort

Annotation schema

<table>
<thead>
<tr>
<th>Event code</th>
<th>Kind</th>
<th>Target</th>
<th>Origin</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>command</td>
<td>com.genuitec.eclipse.ast.deploy.core.ui.action.AddDeploymentActionSite m.label.Add Deployment…</td>
<td>menu</td>
<td>Executing</td>
</tr>
<tr>
<td>2</td>
<td>command</td>
<td>com.genuitec.eclipse.easie.core.ui.action.ServerStartAction$item.label..R un Server</td>
<td>toolbar</td>
<td>Executing</td>
</tr>
</tbody>
</table>

Activity logs

Annotated activity logs

<table>
<thead>
<tr>
<th>Event</th>
<th>PC time</th>
<th>Time</th>
<th>Kind</th>
<th>Target</th>
<th>Origin</th>
<th>Delta</th>
</tr>
</thead>
</table>

Effort per activity

Truncate consecutive events with same activity and calculate elapsing time

Automated annotation
**Procedure for extracting activity effort**

Categorization of developers’ activities and sub-activities was adapted from previous work (Layman 2008)

---

**Categorization**

- **Category 1: Editing**
  - Creating a new class
  - Creating a new package
  - Creating a new project
  - Creating a non-source file
  - Creating new source folder
  - Editing manually code
  - Other editing
  - Refactoring

- **Category 2: Executing**
  - Debug action
  - Debug control
  - Debug execution
  - Executing
  - Executing test case
  - Navigate test result

- **Category 3: Navigating**
  - Other navigation
  - Switch to other file

- **Category 4: Reading**
  - Scrolling

- **Category 5: Other activity**
  - Close perspective
  - Close workbench
  - Editor-console
  - Project-environment config.
  - Handle files
  - Handle view
  - Write documentation
  - Unknown
  - Switching to Eclipse
  - Switch perspective

- **Category 6: Searching**
  - Find-Replace
  - Java-File search
  - Navigating search results

- **Category 7: Static Navigation**
  - Display static dependencies
  - Navigating a static dependency
Analysis performed

**TABLE I**

LOC per file type for all four systems.

<table>
<thead>
<tr>
<th>Systems</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>8,205</td>
<td>26,679</td>
<td>4,983</td>
<td>9,960</td>
</tr>
<tr>
<td>JSP</td>
<td>2,527</td>
<td>2,018</td>
<td>4,591</td>
<td>1,572</td>
</tr>
<tr>
<td>Others</td>
<td>371</td>
<td>1,183</td>
<td>1,241</td>
<td>1,018</td>
</tr>
<tr>
<td>Total</td>
<td>11,103</td>
<td>29,880</td>
<td>10,815</td>
<td>12,550</td>
</tr>
</tbody>
</table>

Only Java files considered

Multiple linear regression
- Forward stepwise -

**Types**

<table>
<thead>
<tr>
<th>Dependent variable</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Effort (time) in: Editing, Navigating, Reading, Searching.</td>
</tr>
</tbody>
</table>

**Independent variable**

| Number (or presence) of smells of 12 types in the files on which the developers worked on during the maintenance tasks. |

**Control variables**

| System, Developer, Round, File size (LOC), Revisions (predictor of quality). |
Results: Distribution of activity effort

- Mostly performed activities: Navigating (58.72%), Reading (28.27%), Editing (10.18%) and searching (2.47%)
- Distribution is consistent with Ko et al. 2006 (top four)
- Reading as most consuming activity in Ko et al. 2006
  - Definition of event/action belonging to an activity

For our analysis, we only consider:
Editing, Navigating, Searching and Reading
## Results: Editing Effort

### Model 0 + file size

- $R^2 = 0.11$

### Model 3 - smells

- $R^2 = 0.59$

<table>
<thead>
<tr>
<th>Feature</th>
<th>Model 0</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developers</td>
<td>NA</td>
<td>NA</td>
<td>0.56***</td>
<td>0.29***</td>
</tr>
<tr>
<td>System</td>
<td>NA</td>
<td>NA</td>
<td>0.84</td>
<td>0.69</td>
</tr>
<tr>
<td>Round</td>
<td>NA</td>
<td>NA</td>
<td>-0.23*</td>
<td></td>
</tr>
<tr>
<td>File Size</td>
<td>NA</td>
<td>0.77*</td>
<td>0.92***</td>
<td>0.71***</td>
</tr>
<tr>
<td>Changes (Revisions)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>2.15***</td>
</tr>
<tr>
<td>Data Class</td>
<td>NA</td>
<td>0.77*</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>Data Clump</td>
<td>NA</td>
<td>NA</td>
<td>0.71***</td>
<td>0.71***</td>
</tr>
<tr>
<td>Duplicated Code in conditional branches</td>
<td>NA</td>
<td>NA</td>
<td>1.28**</td>
<td>0.69</td>
</tr>
<tr>
<td>Feature Envy</td>
<td>NA</td>
<td>0.92***</td>
<td>1.39***</td>
<td>1.06**</td>
</tr>
<tr>
<td>God Class</td>
<td>NA</td>
<td>1.84***</td>
<td>1.28**</td>
<td>0.69</td>
</tr>
<tr>
<td>God Method</td>
<td>NA</td>
<td>NA</td>
<td>0.69</td>
<td>0.69</td>
</tr>
<tr>
<td>ISP Violation</td>
<td>NA</td>
<td>1.39***</td>
<td>1.06**</td>
<td>0.55</td>
</tr>
<tr>
<td>Misplaced Class</td>
<td>NA</td>
<td>-0.58*</td>
<td></td>
<td>-0.53</td>
</tr>
<tr>
<td>Refused Bequest</td>
<td>NA</td>
<td>-0.58*</td>
<td></td>
<td>-0.53</td>
</tr>
<tr>
<td>Shotgun Surgery</td>
<td>NA</td>
<td>0.78*</td>
<td>0.68*</td>
<td>0.58*</td>
</tr>
<tr>
<td>Temporary variable is used for several purposes</td>
<td>NA</td>
<td>0.68*</td>
<td>0.58*</td>
<td></td>
</tr>
<tr>
<td>Use interface instead of implementation</td>
<td>NA</td>
<td>0.78*</td>
<td>0.68*</td>
<td>0.58*</td>
</tr>
</tbody>
</table>

### Adjusted $R^2$

- Model 0: 0.009
- Model 1: 0.26
- Model 2: 0.29
- Model 3: 0.61

$\alpha = 0.001$ (***), $\alpha = 0.01$ (**), $\alpha = 0.05$ (*)
Results: Editing Effort

Model 0 + file size
R² = 0.11

Model 3 - smells
R² = 0.59

Compared to code smells, file size has limited impact.
Contrast with previous study by Sjøberg et al [2013]
Effect of code smells vanishes when the revisions is included
God Class, ISP Violation, and Use interface instead of implementation can indicate larger effort.
Results: Navigating Effort

<table>
<thead>
<tr>
<th></th>
<th>Model 0</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developers</td>
<td>-0.17 ***</td>
<td>-0.17 ***</td>
<td>-0.18 ***</td>
<td>-0.20 ***</td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
<td>0.12 *</td>
<td></td>
</tr>
<tr>
<td>Round</td>
<td>-0.29 *</td>
<td>-0.29 *</td>
<td>-0.34 **</td>
<td></td>
</tr>
<tr>
<td>File Size</td>
<td>NA</td>
<td>NA</td>
<td>0.72 ***</td>
<td>0.52 ***</td>
</tr>
<tr>
<td>Changes (Revisions)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Data Class</td>
<td></td>
<td>NA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Clump</td>
<td></td>
<td>NA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duplicated Code in conditional branches</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>1.60 ***</td>
</tr>
<tr>
<td>Feature Envy</td>
<td>NA</td>
<td>0.92 ***</td>
<td>0.65 ***</td>
<td></td>
</tr>
<tr>
<td>God Class</td>
<td>NA</td>
<td>1.99 ***</td>
<td>1.27 **</td>
<td>0.83 *</td>
</tr>
<tr>
<td>God Method</td>
<td>NA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISP Violation</td>
<td>NA</td>
<td>1.09 **</td>
<td>0.66 *</td>
<td></td>
</tr>
<tr>
<td>Misplaced Class</td>
<td>NA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refused Bequest</td>
<td>NA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shotgun Surgery</td>
<td>NA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temporary variable is used for several purposes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Use interface instead of implementation</td>
<td>NA</td>
<td>NA</td>
<td>0.72 *</td>
<td>0.53 *</td>
</tr>
<tr>
<td>Adjusted $R^2$</td>
<td>0.02</td>
<td>0.26</td>
<td>0.31</td>
<td>0.50</td>
</tr>
</tbody>
</table>

$\alpha = 0.001 (***)$, $\alpha = 0.01 (**)$, $\alpha = 0.05$ (*)
Results: Navigating Effort

Revisions still impacts more than file size and code smells

Only God Class remains significant after revisions
## Results: Reading Effort

<table>
<thead>
<tr>
<th>Feature Envy</th>
<th>Model 0</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>God Class</td>
<td>NA</td>
<td>NA</td>
<td>2.31 ***</td>
<td>1.01 *</td>
</tr>
<tr>
<td>God Method</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISP Violation</td>
<td>NA</td>
<td>NA</td>
<td>0.87 *</td>
<td></td>
</tr>
<tr>
<td>Misplaced Class</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refused Bequest</td>
<td>NA</td>
<td>NA</td>
<td>-0.69 *</td>
<td></td>
</tr>
<tr>
<td>Shotgun Surgery</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temporary variable is used for several purposes</td>
<td>NA</td>
<td>NA</td>
<td>-0.22 *</td>
<td></td>
</tr>
<tr>
<td>Use interface instead of implementation</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adjusted $R^2$</td>
<td>0.03</td>
<td>0.22</td>
<td>0.37</td>
<td>0.47</td>
</tr>
</tbody>
</table>

$\alpha = 0.001$ (***) $\alpha = 0.01$ (**), $\alpha = 0.05$ (*)
### Results: Reading Effort

Revisions and file size explain more the effort than code smells.

Change size explains the effort more than the file size.

<table>
<thead>
<tr>
<th></th>
<th>Model 0</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developers</td>
<td>-0.19 ***</td>
<td>-0.19 ***</td>
<td>-0.23 ***</td>
<td>-0.24 ***</td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Round</td>
<td>-0.36 *</td>
<td>-0.35 *</td>
<td>-0.35 *</td>
<td>-0.39 **</td>
</tr>
<tr>
<td>File Size</td>
<td>NA</td>
<td>NA</td>
<td>1.30 ***</td>
<td>1.14 ***</td>
</tr>
<tr>
<td>Change size</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daily</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Go</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISV Violation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misplaced Class</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refused Bequest</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shotgun Surgery</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temporary variable is used for several purposes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use interface instead of implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adjusted $R^2$</td>
<td>0.03</td>
<td>0.22</td>
<td>0.37</td>
<td>0.47</td>
</tr>
</tbody>
</table>

$\alpha = 0.001$ (***) , $\alpha = 0.01$ (**) , $\alpha = 0.05$ (*)
## Results: Searching Effort

<table>
<thead>
<tr>
<th>Feature Envy</th>
<th>Model 0</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjusted $R^2$</td>
<td>0.11</td>
<td>0.24</td>
<td>0.27</td>
<td><strong>0.35</strong></td>
</tr>
</tbody>
</table>

$\alpha = 0.001$ (***) $\alpha = 0.01$ (**), $\alpha = 0.05$ (*)
Results: Searching Effort

Revisions impacts more than file size (magnitude and model fit)
Only Feature Envy smell affects the searching effort
Maintenance problems in previous work related to increased effort for editing, navigating and reading.

<table>
<thead>
<tr>
<th>Feature Envy</th>
<th>Data Class</th>
<th>Data Clump</th>
<th>Duplicated Code in conditional branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISP Violation</td>
<td>Misplaced Class</td>
<td>Refused Bequest</td>
<td>Shotgun Surgery</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Temporary variable is used for several purposes, Use interface instead of implementation</td>
</tr>
</tbody>
</table>

**Summary of Results**

Smells explain betteroring effort than file size, but not for Reading and Searching.

*“+”: require more effort*

*“-”: required less effort*

*“empty”: no effect on the effort*
Threats to validity

‣ Learning effect (accounted with rounds)
‣ Instrumentation and log processing accuracy
‣ Choice of tools for code smells (inCode and Together)
‣ Generalization is limited to context

Future work

‣ Think-aloud + Log analysis = how smells affect the activities
‣ Taxonomies on programming problems during maintenance
‣ Explore GLM for better explanatory models
‣ More replications!
What to take home today...

› Structural attributes represented in the form of different code smells do indeed have an effect on the developers’ effort for certain kinds of activities.

› Different code smells significantly impact the effort of different activities. For example, we found that “Feature Envy” affects searching effort while “Data Clumps” affects editing effort.

› The effect of code smells on editing and navigating effort is, in fact, larger than file size, whiles the opposite is the true for reading and searching effort.

› If the effect of code smells is contingent on the type of activity, this may mean that is contingent on the task at hand (e.g., some tasks may require more reading than others).
Thank you!

Contact: aiko.yamashita@cwi.nl