Mining Rails - Learning from your App’s Lifeline

Michael Feathers - Obtiva Corporation
Corey Haines - Corey Haines
What do we know about how we write code?
Code Metrics
## Lines of Code

<table>
<thead>
<tr>
<th>C</th>
<th>COBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>#include &lt;stdio.h&gt;</td>
<td>000100 IDENTIFICATION DIVISION.</td>
</tr>
<tr>
<td>int main() {</td>
<td>000200 PROGRAM-ID. HELLO WORLD.</td>
</tr>
<tr>
<td>printf(&quot;\nHello\n&quot;);</td>
<td>000300</td>
</tr>
<tr>
<td>}</td>
<td>000400*</td>
</tr>
<tr>
<td></td>
<td>000500 ENVIRONMENT DIVISION.</td>
</tr>
<tr>
<td></td>
<td>000600 CONFIGURATION SECTION.</td>
</tr>
<tr>
<td></td>
<td>000700 SOURCE-COMPUTER. RM-COBOL.</td>
</tr>
<tr>
<td></td>
<td>000800 OBJECT-COMPUTER. RM-COBOL.</td>
</tr>
<tr>
<td></td>
<td>000900</td>
</tr>
<tr>
<td></td>
<td>001000 DATA DIVISION.</td>
</tr>
<tr>
<td></td>
<td>001100 FILE SECTION.</td>
</tr>
<tr>
<td></td>
<td>001200</td>
</tr>
<tr>
<td></td>
<td>100000 PROCEDURE DIVISION.</td>
</tr>
<tr>
<td></td>
<td>100100</td>
</tr>
<tr>
<td></td>
<td>100200 MAIN-LOGIC SECTION.</td>
</tr>
<tr>
<td></td>
<td>100300 BEGIN.</td>
</tr>
<tr>
<td></td>
<td>100400 DISPLAY &quot; &quot; LINE 1 POSITION 1 ERASE EOS.</td>
</tr>
<tr>
<td></td>
<td>100500 DISPLAY &quot;Hello world!&quot; LINE 15 POSITION 10.</td>
</tr>
<tr>
<td></td>
<td>100600 STOP RUN.</td>
</tr>
<tr>
<td></td>
<td>100700 MAIN-LOGIC-EXIT.</td>
</tr>
<tr>
<td></td>
<td>100800 EXIT.</td>
</tr>
</tbody>
</table>

Lines of code: 4  
(excluding whitespace)  
Lines of code: 17  
(excluding whitespace)
Cyclomatic Complexity

\[ M = E - N + 2P \]

where

- \( M \) = cyclomatic complexity
- \( E \) = the number of edges of the graph
- \( N \) = the number of nodes of the graph
- \( P \) = the number of connected components

A control flow graph of a simple program. The program begins executing at the red node, then enters a loop (group of three nodes immediately below the red node). On exiting the loop, there is a conditional statement (group below the loop), and finally the program exits at the blue node. For this graph, \( E = 9 \), \( N = 8 \) and \( P = 1 \), so the cyclomatic complexity of the program is 3.
Halstead Complexity Measures

- $n1$ = the number of distinct operators
- $n2$ = the number of distinct operands
- $N1$ = the total number of operators
- $N2$ = the total number of operands

From these numbers, five measures can be calculated:

- Program length: $N = N1 + N2$
- Program vocabulary: $n = n1 + n2$
- Volume: $V = N \times \log_2 n$
- Difficulty: $D = \frac{n1}{2} \times \frac{N2}{n2}$
- Effort: $E = D \times V$

The difficulty measure is related to the difficulty of the program to write or understand, e.g. when doing code review.
Flog shows you the most torturous code you wrote. The more painful the code, the higher the score. **The higher the score, the harder it is to test.**

Run it against your best stuff. I *double-dog* dare you.

Flog essentially scores an ABC metric: Assignments, Branches, Calls, with particular attention placed on calls.

Run flog on all your code. Try this:

```
find lib -name "/.*rb" | xargs flog
```
We have our history
Want to get a parole? Have your attorney schedule your parole hearing after lunch. A new study revealed that the timing of a parole hearing is quite crucial in determining the outcome.

As a case study, one of the judges started in the morning by granting parole to about 65 percent of the prisoners; that percentage dropped to near zero by the end of the first session, then rebounded to about 65 percent after the snack break. The same pattern repeated in the second and third sessions.

The researchers suggest that as the number of rulings in a session increase, the judges become mentally fatigued. Once their mental resources are depleted, the judges are more likely to simplify their decisions. Ruling in favor of the status quo—denying parole—is the "easier" decision, the authors argue, since these rulings take generally take less time and require shorter written verdicts. After taking a break, their faculties are restored, and they are more likely to make "harder" decisions and grant parole requests again.

Link
Each judge took two breaks. One at mid-morning beginning as early as 9:45 a.m. or as late as 10:30 a.m., and a lunch break that began between 12:45 p.m. and 2:10 p.m.

"You're always surprised when you find effects where you don't want to find them," Jonathan Levav of Columbia University said in a telephone interview. "If you're a social scientist it gets you excited. But, as an ordinary citizen, you don't want to find this."
What do we know about how we write code?
https://github.com/michaelfeathers/repodepot-ruby
r = RepoDepot::Repository.new('ma_data', events)

r.events.map { |e| e.date.hour }.freq
events.first
Finding Situational Truths
What do we know about how we write code?
WARNING

This Area Contains Chemicals Known To The State Of California To Cause Cancer And Birth Defects Or Other Reproductive Harm.

California Health & Safety Code Section 25249.6

More Information On Specific Exposures Has Been Provided To Tenants And Is Available At www.prop65apt.org
Destroying Teams by Doing Stats on “Committer”

(problem or not)
What Do “Committer” and “Author” Mean When We Are Pairing?
People/Teams Have Different Commit Styles
What do we know about how we write code?
What else can we tell?
Thanks!

https://github.com/michaelfeathers/repodepot-ruby