Seminar: Code Smells and Refactoring Techniques

October 8, 2019

TIMELINE

- Introduction/Purpose
- Common Code smells (live)
- Refactoring techniques (live)

INTRODUCTION

Materials

- Seminar Repository
- refactoring.guru
- sourcemaking.com
- Clean Code Architecture for Node
- NodeJS Events API
- PubSubJS

What is a Code Smell?

Types

- Implementation Smells
- Design Smells
- Architecture Smells

Motivation

http://swat.polymtl.ca/publications.html
http://swat.polymtl.ca/~foutsekh/docs/Saboury-SANER.pdf

An Empirical Study of Code Smells in JavaScript Projects

Amir Saboury, Pooya Musavi, Foutse Khomh and Giulio Antoniol
Polytechnique Montreal, Quebec, Canada
{amir.saboury, pooya.musavi, foutse.khomh, giuliano.antoniol}@polymtl.ca

***Abstract*—JavaScript is a powerful scripting programming language that has gained a lot of attention this past decade. Initially used exclusively for client-side web development, it has evolved to become one of the most popular programming languages, with developers now using it for both client-side and server-side application development. Similar to applications written in other programming languages, JavaScript applications contain *code smells*, which are *poor* design choices that can negatively impact the quality of an application. In this paper, we investigate code smells in JavaScript server-side applications with the aim to understand how they impact the fault-proneness of**

developers are not equipped with a compiler that can help them spot erroneous and unoptimized code. As a consequence of all these characteristics, JavaScript applications often contain code smells [4], *i.e.*, poor solutions to recurring design or implementation problems. However, despite the popularity of JavaScript, very few studies have investigated code smells in JavaScript applications, and to the best of our knowledge, there is no work that examines the impact of code smells on the fault-proneness of JavaScript applications. This paper

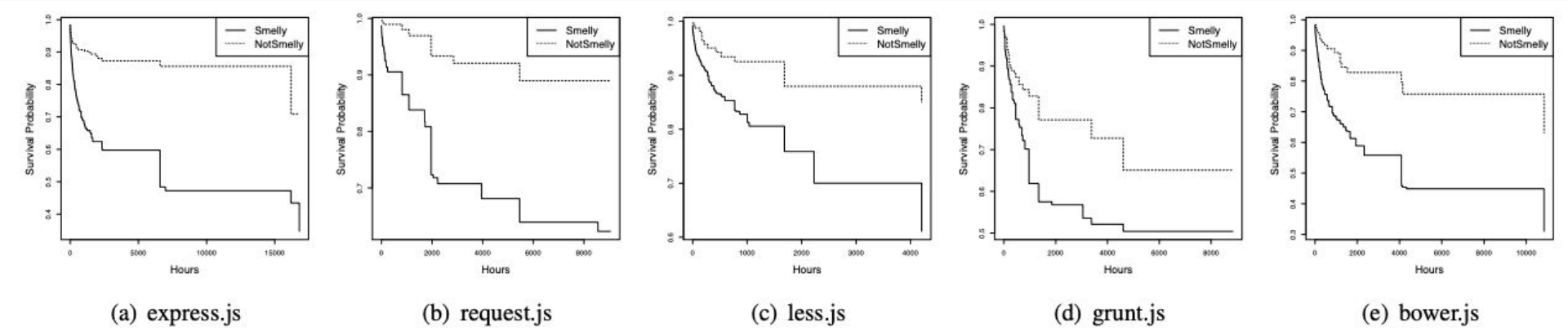
Study

1. Is the risk of fault higher in files with code smells in comparison with those without code smell?

2. Are JavaScript files with code smells equally fault-prone?

Answer 1

On average, JavaScript files without code smells have hazard rates **65%** lower than JavaScript files with code smells.



(a) express.js (b) request.js (c) less.js (d) grunt.js (e) bower.js

Answer 2

Empirically:

- Variable Re-assign
- Assignment in Conditional Statements

Perceptually:

- Nested Callbacks,
- Variable Re-assign and
- Long Parameter List

module	covariate	$exp(coef)$	p-value (Cox hazard model)	p-value (Proportional hazards assumption)
express	No.Previous-Bugs	1.013	0.05e-3	0.870
	Chained Methods	7.931	0.003	0.961
	This Assign	2.584	0.038e-8	0.716
	Variable Re-assign	1.488	0.007	0.253
grunt	Nested Callbacks	3.534	0.002	0.204
	Variable Re-assign	1.514	0.039	0.913
	Assign. in Cond. State.	2.212	0.001	0.829
bower	No.Previous-Bugs	1.019	0.019	0.451
	Depth	7.786	0.065e-4	0.910
	LOC	1.008e-1	0.029	0.241
less	No.Previous-Bugs	1.036	0.02e-14	0.741
	Complex Switch Case	0.481	0.027	0.417
	Assign. in Cond. State.	1.646	0.019e-2	0.940
request	No.Previous-Bugs	1.067	0.002	0.407
	Depth	0.172	0.052e-3	0.620
	Variable Re-assign	3.277	0.088e-2	0.733

Related Work

- https://archive.fosdem.org/2019/schedule/track/ml_on_code/
- https://archive.fosdem.org/2019/schedule/event/ml_on_code_smelling_source/

Theory :(

Technical Debt

Originally suggested by Ward Cunningham

Causes

- Business Pressure
- Failing to combat the strict coherence of components
- Lack of tests
- Lack of documentation

Technical Debt

Originally suggested by Ward Cunningham

Causes (2)

- Long-term simultaneous development in several branches
- Delayed refactoring
- Lack of compliance monitoring

When to refactor

1. Rule of Three
2. When adding a feature
3. When fixing a bug
4. During a code review

How to refactor

1. The code should become cleaner.
2. New functionality shouldn’t be created during refactoring.
3. When fixing a bug
4. All existing tests must pass after refactoring.

Common Code Smells

Live demo

Refactoring Techniques

Live demo

Node App Architectures

- Layered
- MVC
- Clean Architecture

Tools

- https://github.com/danielstjules/jsinspect
- https://github.com/kucherenko/jscpd
- https://github.com/elijahmanor/eslint-plugin-smells