Basic Premise:

Some code patterns cost more to maintain than others. Quantifying them can be complex and time-consuming. This app helps software developers and managers quantify costs in any JavaScript Github repo visible to them.

Approach:

Single html page

React for HTML+SVG element rendering

Experimental functional UI toolset wrapping React

D3 for path generation/interpolation calculations

Eslint rules to find patterns in code

Basic Usage:

User inputs a github token

User inputs a github repo path

User clicks load

App recursively requests the file tree from Github's GraphQL API.

App renders visualization and metrics.

User can hover on on files, rules, or nothing. (Mockups of each hover state attached.)

Success criteria:

Folks I know actually find it useful, or potentially useful.

Addresses a problem I know to exist.

Publicly available on the web.

Truly serverless, for simplicity.

Data Types

name	data_type	attribute_type1	attribute_type2	attribute_type3	description
file_name	attribute	ordered	sequential		self-explanatory
rule_name	attribute	ordered	sequential		eslint rules used to parse code, customized to output changeability and reliability metrics. For more about eslint and rules, see https://eslint.org/docs/about/
file_affected_locations	position				locations in file where code triggered rule
repo_affected_locations	position				locations in repo where code triggered rule
file_lines_count	attribute	ordered	sequential	quantitative	total lines of code in file
repo_lines_count	attribute	ordered	sequential	quantitative	total lines of code in repo
file_lines_affected_count	attribute	ordered	sequential	quantitative	total lines of file where code triggers rule
repo_lines_affected_count	attribute	ordered	sequential	quantitative	total lines of repo files where code triggers rule
file_lines_affected_percent	attribute	ordered	sequential	quantitative	file_lines_affected_count / file_lines_count
repo_lines_affected_percent	attribute	ordered	sequential	quantitative	repo_lines_affected_count / repo_lines_count
cost_per_hour	item				user-estimated developer cost per hour
hours_per_change	item				user-estimated person-hours required to go from code change decision to production code deployed
rule_reliability_per_line	attribute	ordered	sequential	quantitative	estimated amount of reliability lost (i.e. non-determinism introduced) by this rule affecting one line
rule_changeability_per_line	attribute	ordered	sequential	quantitative	Hours per change multiplier. Estimated amount changeability (aka modifiability, maintainability, flexibility) lost by this rule affecting one line.
file_reliability	attribute	ordered	sequential	quantitative	total file reliability metric, calculation tbd... something like map(rules => rule_reliability_per_line * rule_lines_affected / file_lines_count)).sum()
repo_reliability	attribute	ordered	sequential	quantitative	sum of file reliabilities / total file count
file_changeability	attribute	ordered	sequential	quantitative	total file changeability metric, calculation tbd... something like map(rules => rule_changeability_per_line * rule_lines_affected / file_lines_count)).sum()
repo_changeability	attribute	ordered	sequential	quantitative	sum of file changeabilities / total file count
file_user_impact	attribute	ordered	sequential	quantitative	1 - file_reliability
repo_user_impact	attribute	ordered	sequential	quantitative	1 - repo_reliability
file_cost_per_change	attribute	ordered	sequential	quantitative	tbd... something like file_changeability * cost_per_hour * hours_per_change
repo_cost_per_change	attribute	ordered	sequential	quantitative	tbd... something like repo_changeability * cost_per_hour * hours_per_change

Vis Combinations

question	state	actionTypes	targetTypes	dataTypes in	operations	idiom
Which are the highest cost files for all rules across the repo?	nothing hovered			file & repo attributes derived from user input items and eslint rules		
		identify	extremes	file_cost_per_change	order	table
		compare	extremes	file_cost_per_change	saturation	tree
		compare	distribution	file_cost_per_change	saturation+size	bars within table
		compare	distribution	file_user_impact	saturation+size	bars within table
		compare	features	rule_changeability_per_line	order+saturation	table
Which are the highest cost files for one rule across the repo?	rule hovered			chained from previous		
		identify	extremes	file_cost_per_change	order	table
		compare	distribution	file_cost_per_change	size	bars within table
		compare	distribution	file_user_impact	size	bars within table
		locate	features	file_cost_per_change	saturation	tree

Note1: Tables are sorted by changeability. I'd like to add a way to sort by specific quality attribute (e.g., changeability, reliability, etc.), but I'm unsure how to best do that. Maybe sortBy arrows on each table column header. I'm hesitant to make the table sortable since it can add significant implementation complexity.

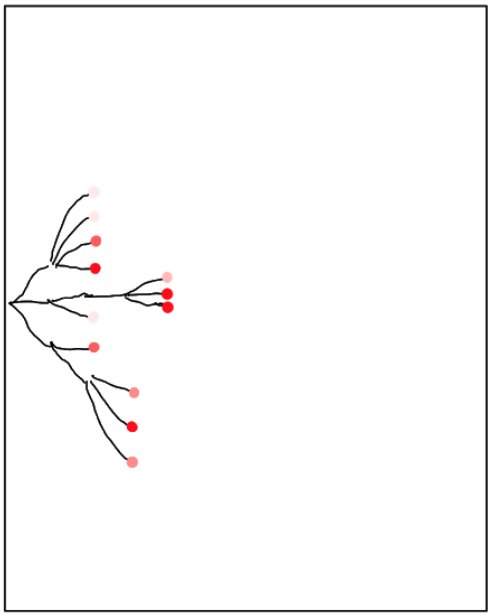
Note2: What the mockups don't show is comparing two repos. The code currently allows adding another repo, so some comparison is possible. I'll likely use the repo stats for that.

Cost per Change

Auth Token

Cost Per Hour

GitHub Repository Path



Rule

- Cyclomatic Complexity
- Low Cohesion
- Function Length
- Many Params
- Mutable Params
- Deep Inheritance
- Deep Scoping

Highest Cost Files Per Month

% of rules affecting	File Name	Commits	Cost Per Commit	User Impact	Cost to change	Cost to Fix
	Filename					
	Filename					
	Filename					
	Filename					
	Filename					
	Filename					
	Filename					

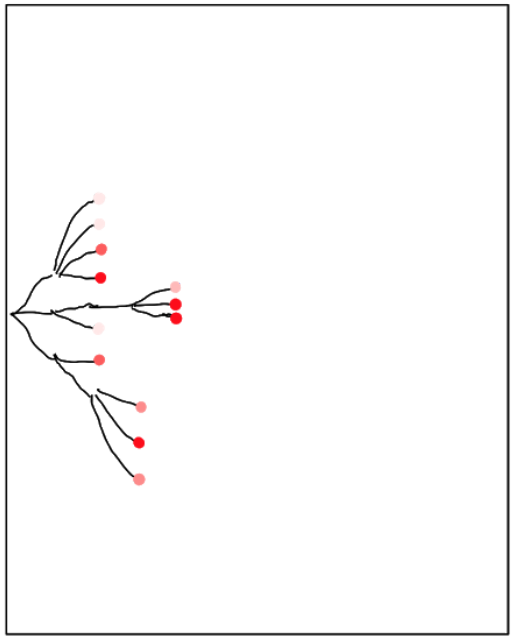
Hovering Nothing

Cost per Change

Auth Token

Cost Per Hour

GitHub Repository Path



Rule

- Cyclomatic Complexity
- Low Cohesion
- Function Length
- Many Params
- Mutable Params
- Deep Inheritance
- Deep Scoping

Highest Cost Files Per Month

% of File affected	File Name	Commits	Cost Per Commit	User Impact	Cost to change	Cost to Fix
	Filename					
	Filename					
	Filename					
	Filename					
	Filename					
	Filename					
	Filename					

Hovering Rule

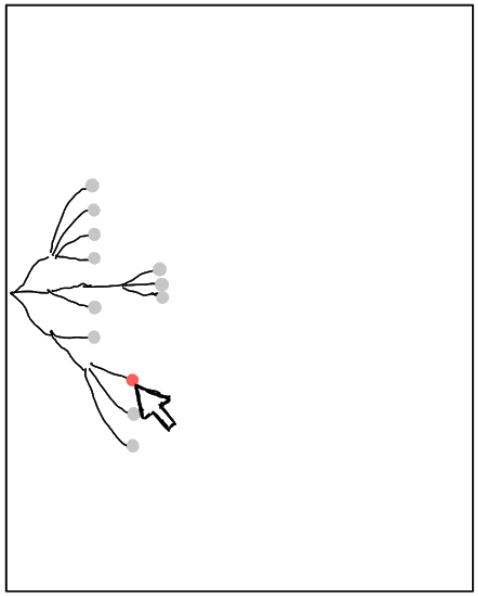
The more I consider this one, the more I think the value of going down to specific lines isn't worth the extra implementation complexity.

Cost per Change

Auth Token

Cost Per Hour

GitHub Repository Path



Rule

- Cyclomatic Complexity
- Low Cohesion
- Function Length
- Many Params
- Mutable Params
- Deep Inheritance
- Deep Scoping

Highest Cost Code Sections in File

% of rules affecting	Lines	# Other Locations Impacted	user impact	cost to Change	Cost To Fix
	60-84				
	93-117				
	156-204				
	313-322				
	386-418				
	512-586				

Hovering File