Client
NASA

Date
Fall 2020

Code Health Check –

Selected Open Source NASA Applications

sema

About this document

- This is a demonstration project of Sema's capabilities based on analyzing certain publicly available (open source) NASA applications
- Sema analyzed 9 repositories- Apollo-11, cumulus, delta, DERT, Kamodo, mct, openmct, vscode-pvs
- 311 developers have made changes to these applications over the last 90 days
- Time to complete this proof of concept: 30 min of setup, 1 hour of processing, 5 hours of report writing

Executive Summary - Code

Element of Code Quality	Discussion
Size	• 9 repositories, 3,711,197 lines, 11,280 files, and 485,634,428 bytes.
Language Composition	• Overall Strength – the organization uses popular, well-supported programming languages (JavaScript, and Java). Updates, security patches, and skilled developers should remain available for the foreseeable future. • Potential Risk – The Apollo code was written in Assembly. This is a high-risk language for recruiting talent: there are few available developers. It is still a language commonly in use, however, and is rated highly by developers.
Core Technical Debt	• Discuss/Low Risk – Cost per Line of Code for the organization is at $3.38. • The bulk of the core technical debt is in the "delta" repository; the cost per line of code for "DERT", "Kamodo" and "delta" is over $5 while "vscode-pvs" and "cumulus" are under $2.
Line Level Warnings	• 229,830 instances of line level warnings were identified. Stylistic warnings represent 52% of that number and are not considered technical debt. • Misleading is the next largest category, representing 29% of line level warnings. • High Risk – Investigate and as needed remediate 45 Security warnings. • Low Risk – Investigate and as needed remediate 23,460 Potential Bug warnings, 66,082 Misleading, 18,896 Smells, and 1,487 Environment Sensitive warnings.
Third-Party Code	• Low Risk – Less than 1.5% of code is from third-party libraries. Besides Documentation, VS Code workspace files is the most prevalent third-party library.
Package Dependencies	• Low risk– the DERT repository, shown here, as an example does not have a particularly pronounced structure and could be a candidate for architectural improvement
Distributed Repositories and Smaller Packages	• Low Risk – the "mct" repository has 10 large packages with over 1,000 self-referential or outgoing dependencies. This could be a candidate to be broken up.

sema

Executive Summary – Process and Team

Element of Code Quality	Discussion
Commit Analysis	• Discuss two particular patterns: decline in December 2019, likely due to holiday breaks, and in April 2019, likely due to COVID disruption.
Commit Management	• Low Risk– 3 of 7 repositories average more than 5 files per commit over the past year and over the past 90 days • Low Risk – 8 developer averages more than 10 files per commit over the past 90 days
Ticket Referencing	• High Risk – 1 of 7 repositories meets the recommended ticket reference benchmark over the past 90 days (openmct) .
Test Referencing	• Medium Risk – 1 of 7 repositories meets the recommended ticket and test reference benchmark over the past 90 days (openmct). Are these acceptable levels in light of the code being open source?
Email Aliases	• High Risk – ~54% commits in the past 90 days have come from non-organization accounts (29).
Developer Contribution and Skill	• The top 5 contributors all-time are Mike McGann, Paolo M. Masci, lkeelyme, Victor Woeltjen, and Scisco. • Mike McGann has created more files, changed the most of their own files, has the most first commits, and scores highest in experience per repository. This indicates that Victor has significant knowledge of these repositories. • Lauren Frederick has changed others' files more than any other developer. • Paolo M. Masci has made the most positive impact on these repositories for 4 of 6 architectural impact metrics. • "asherp" has committed the most Security warnings and "centos" the most Potential Bug warnings.

Code

Process

Team

sema

Basic Facts

Item	Amount
Total size	797,733 lines, 4,531 files, and 26,245,718 bytes
Number of Repositories Analyzed	9
Number of Groups Analyzed	None
Version Control System Used	Github Open Source
Did Code Owner confirm that all code has been shared?	No (PoC)
First Commit Analyzed	October 19, 2011 in worldview repository
Last Commit Analyzed	September 03, 2020 in openmct and cumulus repositories
Number of contributors	
All time	311
Current (Last 90 Days)	48

sema

Repository Sizing and Language Composition

- 9 repositories were analyzed, they 7 are currently active. “Currently active” means there has been at least one commit to a repository within the last 90 days.
- The time frame analyzed was October 19, 2011 to the present (Sep 4, 2020).
- ”mct” is the largest repository by both file count and by lines of code. Compared to the second largest repository by file count (openmct), ”mct” has ~83 more files and ~59,000 more lines of code.
- 41% of the source files in these repositories are JavaScript and 36% are Java.
- Strength – the organization uses popular, well-supported programming languages. Updates, security patches, and skilled developers should remain available for the foreseeable future.
- Potential Risk – The Apollo code was written in Assembly. This is a high-risk language: there are few available developers. It is still a language commonly in use, however, and is rated highly by developers.

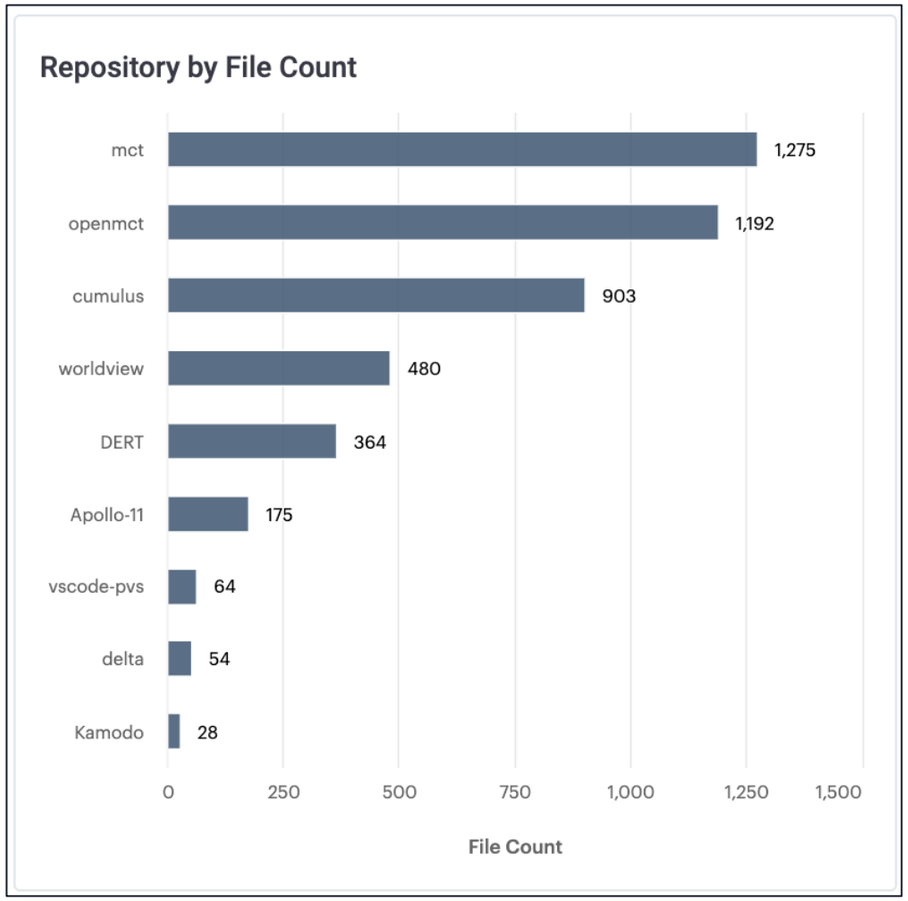
sema

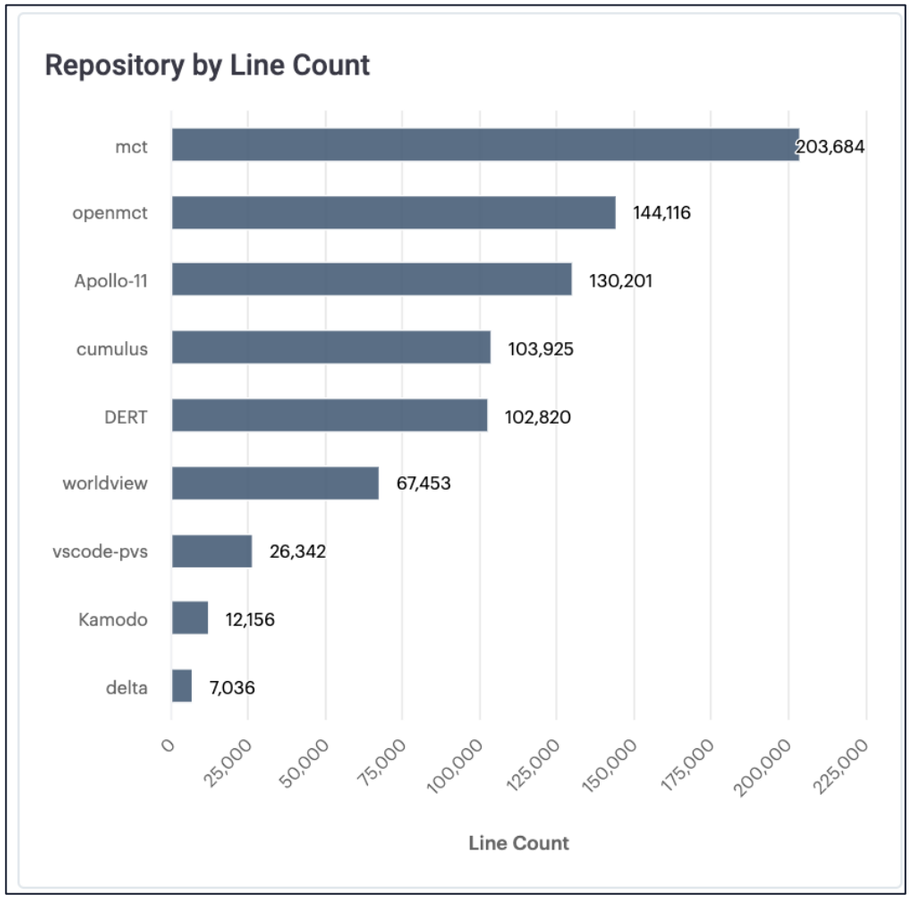
Repository Basic Fact Summary

Repository	First Commit Date	Last Commit Date	Commit Count	File Change Count	File Count
Apollo-11	Apr 03 2014	Sep 02 2020	386	2,599	226
cumulus	Nov 08 2016	Sep 03 2020	10,704	45,931	1,931
delta	Mar 18 2019	Aug 20 2020	514	1,701	81
DERT	Nov 17 2015	Feb 06 2019	429	3,937	543
Kamodo	Oct 17 2018	Aug 31 2020	217	454	99
mct	May 09 2012	May 19 2016	813	5,361	1,747
openmct	Oct 28 2014	Sep 03 2020	5,707	27,795	1,286
vscode-pvs	Mar 18 2019	Aug 30 2020	660	3,759	388
worldview	Oct 19 2011	Sep 02 2020	7,908	46,630	5,013

sema

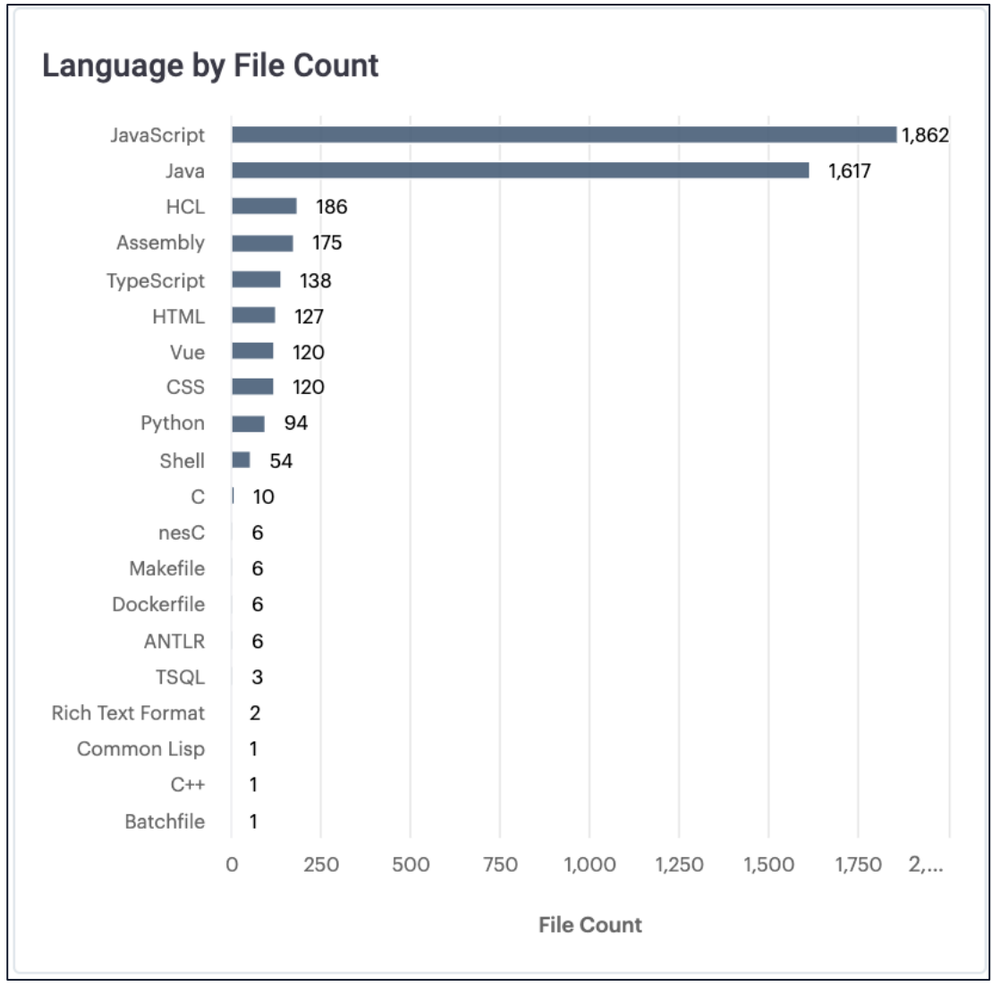
Size by Product- Files and Lines

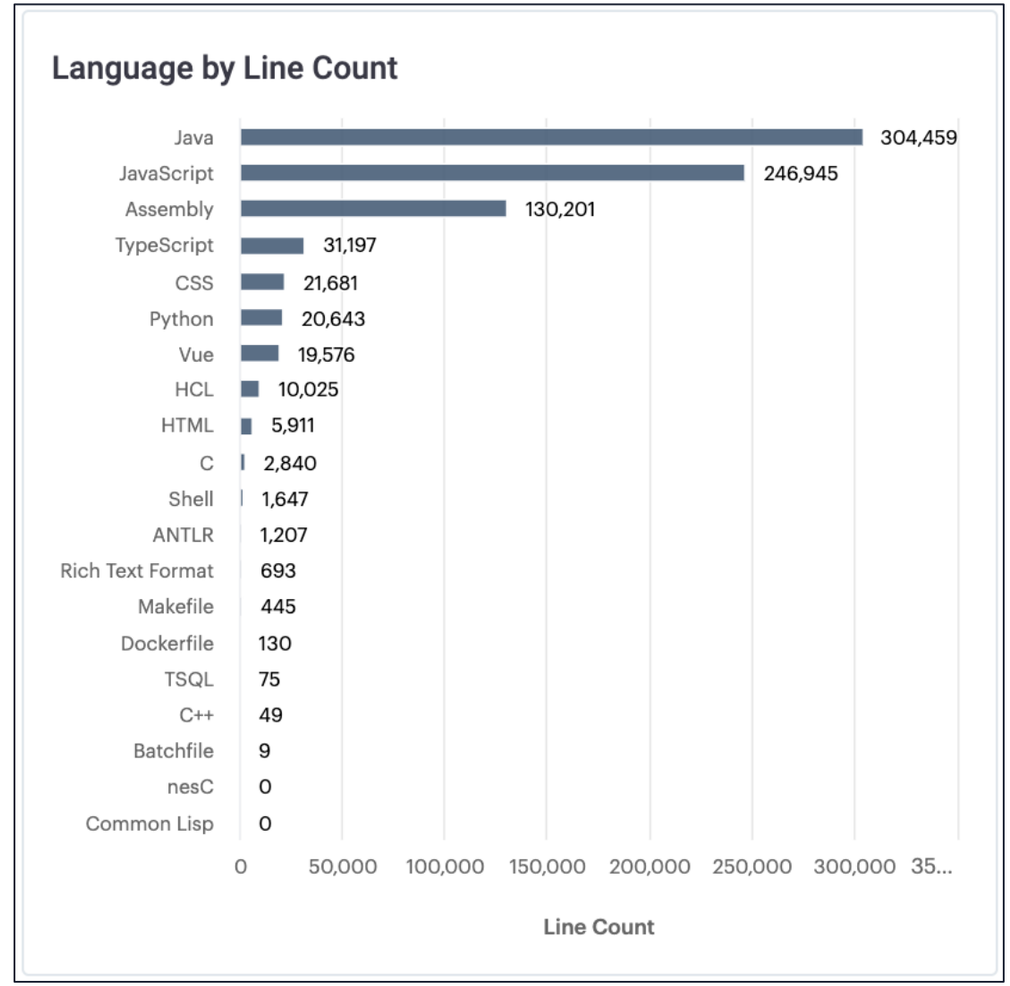




sema

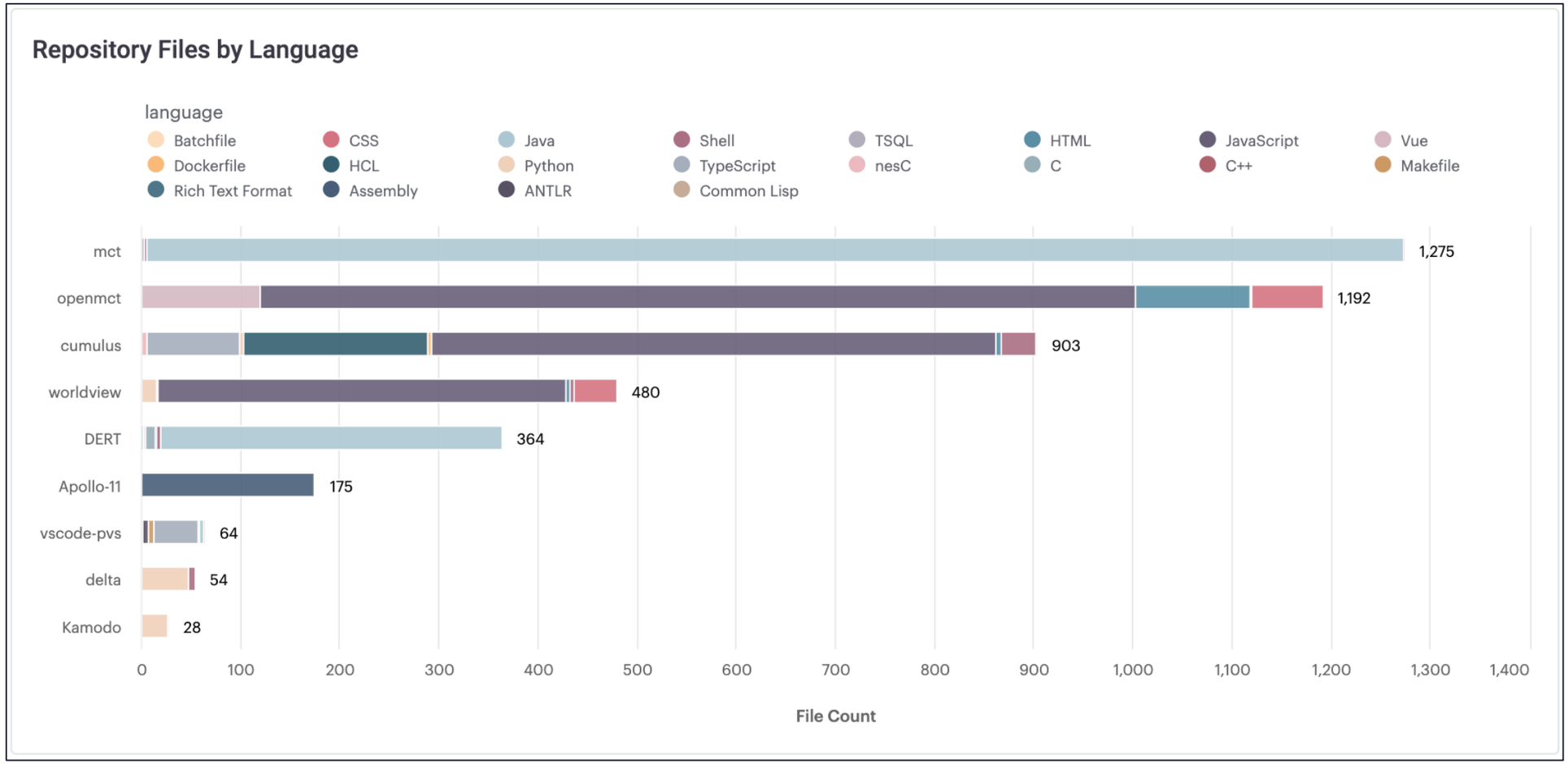
Language Composition for the organization





sema

Language Composition by Repository



sema

Core Technical Debt Sizing and Resolution Cost Estimate- Overview

This section includes four components of technical debt:

- Files with too much complexity
- Duplicated blocks of code
- Unit test coverage
- Substantive line level warnings, covered in more detail in the next section

Then, it estimates the time in person-days to reduce the technical debt at a cost of $500/day based on US averages for benchmarking.

Sema does not recommend reducing technical debt to $0 as the cost does not outweigh the effort.

However, technical debt should be monitored and optimized for continued cost-effective development or maintenance of a code base.

Core Technical Debt – All Code, Third-Party and In-House

ORGANIZATION	COST PER LOC	TECH DEBT TOTAL	CURRENT TEST RAT...	COMPLEXITY DAYS	DUPLICATION DAYS	UNIT TEST DAYS	LINE LEVEL WARNI...	TOTAL DAYS TO FIX
NASA	$3.38	$878,217	48%	12	0	1,492	253	1,756

REPOSITORY	COST PER LOC	TECH DEBT TOTAL	CURRENT TEST RATIO	DUPLICATION DAYS	COMPLEXITY DAYS	UNIT TEST DAYS	LINE LEVEL WARNING DAYS	TOTAL DAYS TO FIX
delta	$8.93	$30,979	12.51%	0	0	57	4	62
Kamodo	$7.38	$60,426	0.00%	0	1	86	34	121
DERT	$7.10	$302,659	3.06%	0	3	537	65	605
mct	$2.98	$242,231	52.70%	0	5	378	101	484
worldview	$2.76	$116,970	16.79%	0	2	225	8	234
openmct	$2.07	$86,673	53.58%	0	0	169	4	173
vscode-pvs	$1.45	$26,935	18.33%	0	0	40	13	54
cumulus	$0.53	$11,344	218.84%	0	0	0	23	23

- The bulk of the core technical debt is in the "delta" repository; the cost per line of code for "DERT", "Kamodo" and "delta" is over $5 while "vscode-pvs" and "cumulus" are under $2.

- Kamodo has little evidence of unit testing within the repository.
  - Discuss: Might testing be performed in a different manner, or are there more tests than were picked up by this scan due to (for example) a different naming convention?
  - Discuss: Assuming the testing results are accurate, do low levels of unit testing make sense for this open source code, as is frequently the case given the community of developers who can contribute to improvements?

sema

Technical Debt Summary – Third-Party vs. In-House Code

ORGANIZATION	VENDOR CATEGORY	COST PER LOC	TECH DEBT TOTAL	CURRENT TEST RATIO	COMPLEXITY DAYS	DUPLICATION DAYS	UNIT TEST DAYS	LINE LEVEL WARNING DAYS	TOTAL DAYS TO FIX
NASA	In-House	$3.41	$871,837	48.74%	12	0	1,482	250	1,744
NASA	Third-Party	$1.37	$5,646	1.36%	0	0	9	3	11

REPOSITORY	VENDOR CATEGORY	COST PER LOC	TECH DEBT TOTAL	CURRENT TEST RATIO	COMPLEXITY DAYS	DUPLICATION DAYS	UNIT TEST DAYS	LINE LEVEL WARNING DAYS	TOTAL DAYS TO FIX
worldview	Third-Party	$10.50	$1,365	0%	0	0	3	0	3
delta	In-House	$8.93	$30,979	13%	0	0	57	4	62
Kamodo	In-House	$7.38	$60,426	0%	1	0	86	34	121
DERT	In-House	$7.10	$302,659	3%	3	0	537	65	605
mct	In-House	$2.98	$242,231	53%	5	0	378	101	484
cumulus	Third-Party	$2.75	$688	0%	0	0	1	0	1
worldview	In-House	$2.75	$115,801	17%	2	0	222	8	232
openmct	In-House	$2.06	$85,983	54%	0	0	167	4	172
vscode-pvs	In-House	$1.52	$22,502	23%	0	0	34	11	45
vscode-pvs	Third-Party	$0.97	$3,594	0%	0	0	5	3	7
cumulus	In-House	$0.53	$11,256	221%	0	0	0	22	23
openmct	Third-Party	$0.00	$0	243%	0	0	0	0	0

- Medium Risk – In-House cost per line of code (i.e. cost excluding third-party code) is $3.41, driven primarily by lack of unit testing.
    - Over 99% of core technical debt comes from in-house code.
- Low Risk – Third-Party cost per line of code is $1.36.
- Discuss: Barring technology or customer concerns, Sema recommends removing third-party code from repositories wherever possible

sema

Line Level Warnings

Sema identifies over 1000 line level warnings across multiple languages, grouped into seven categories: Environment Sensitive, Misleading, Potential Bug, Smell, Stylistic, Security, and Performance.

229,830 instances of line level warnings were identified. Stylistic warnings represent 52% of that number and are not considered technical debt.

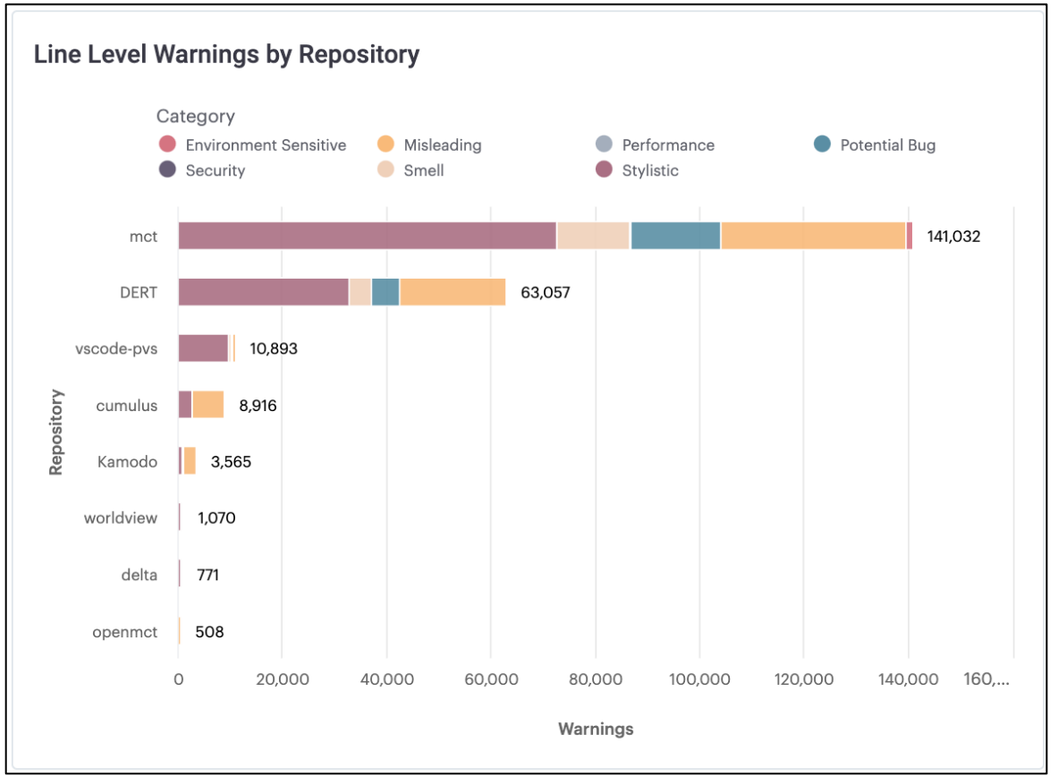
Misleading is the next largest category, representing 29% of line level warnings.

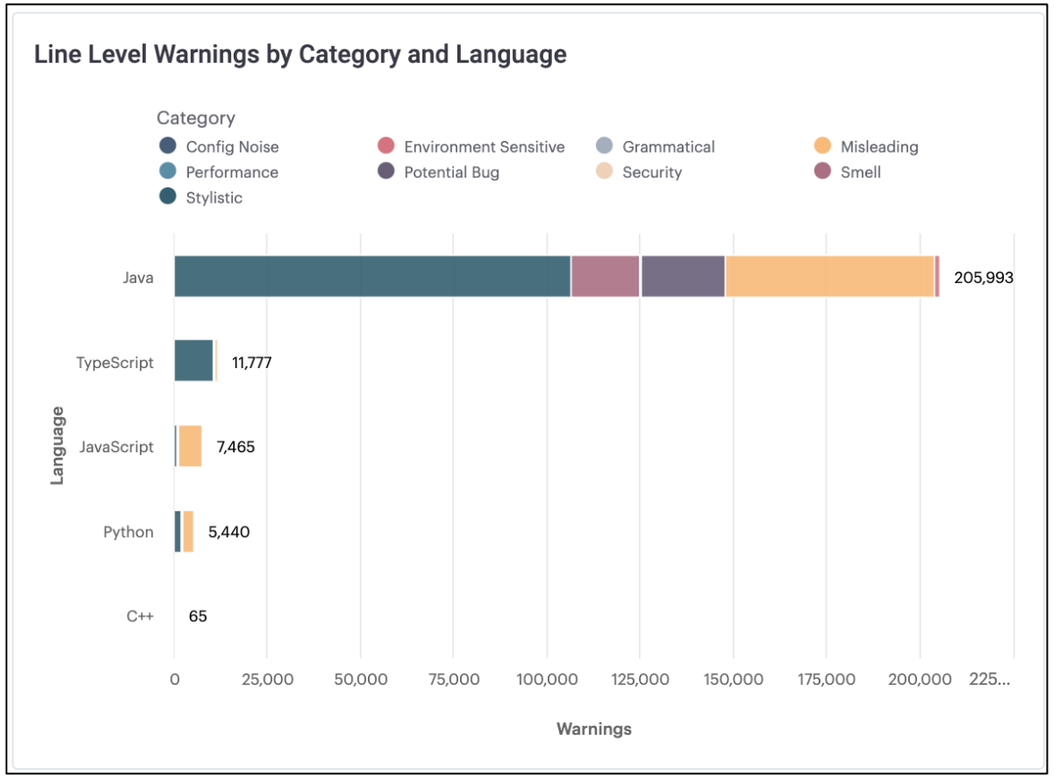
High Risk – Investigate and as needed remediate 45 Security warnings.

Low Risk – Investigate and as needed remediate 23,460 Potential Bug warnings, 66,082 Misleading, 18,896 Smells, and 1,487 Environment Sensitive warnings.

Sema does not recommend investigating Stylistic warnings.

Total Line Level Warnings Across All Repositories

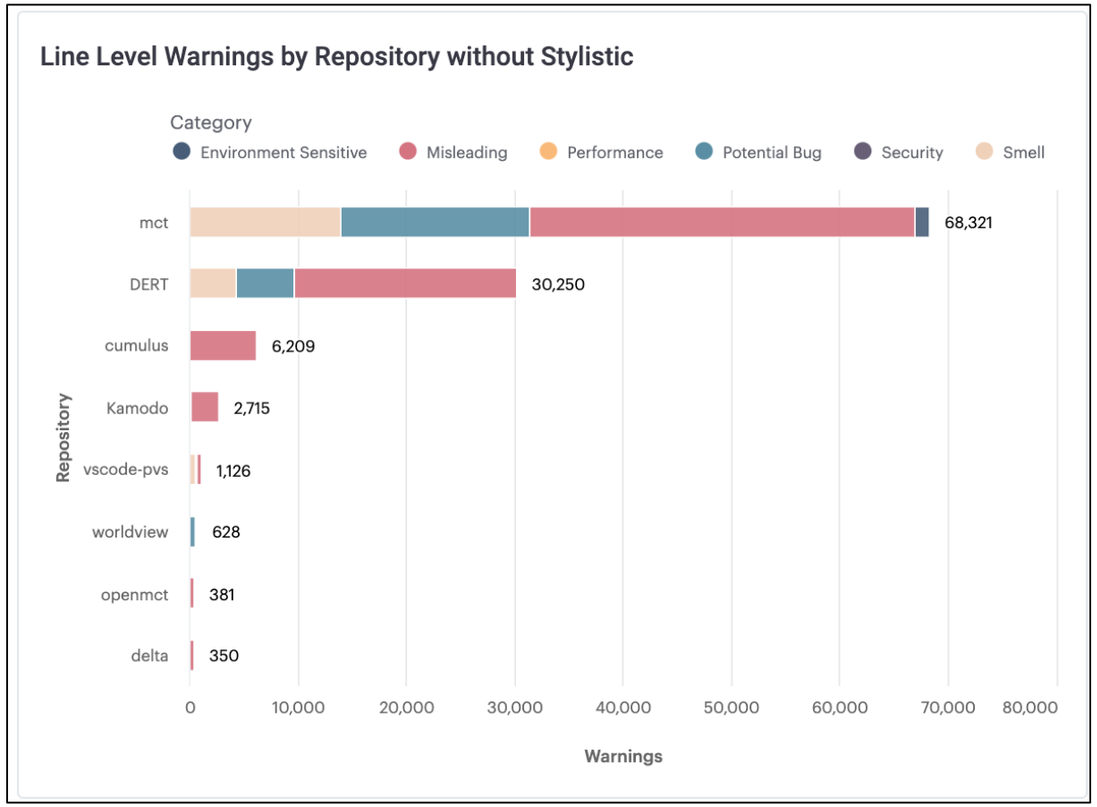


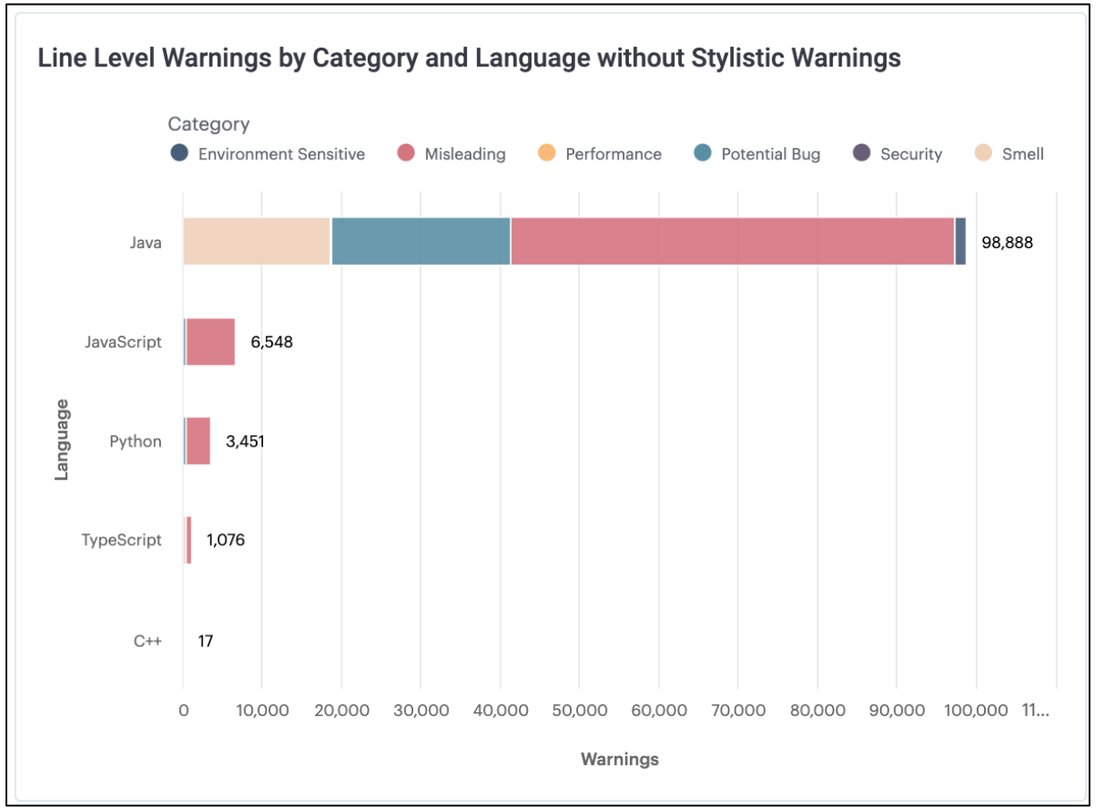


- The "mct" repository accounts for ~61% of all line level warnings.
- Java accounts for ~89% of all line level warnings, TypeScript for ~5%, and JavaScript for ~3%.

sema

Line Level Warnings – Detail Without Stylistic Warnings

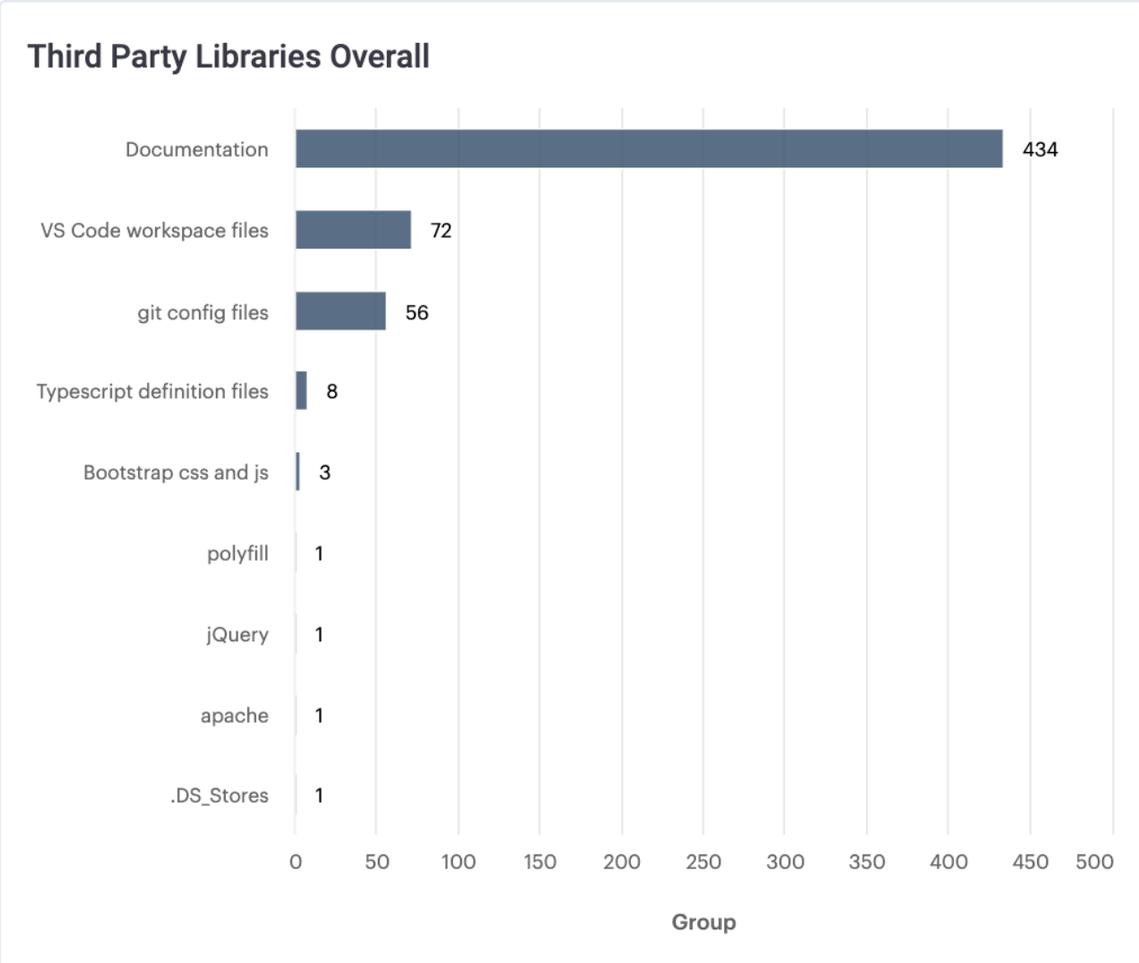


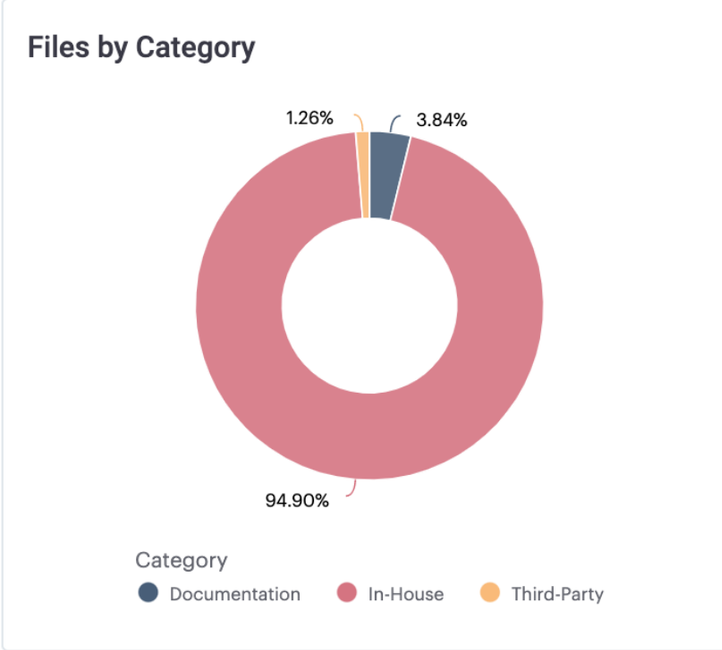


- The "mct" repository accounts for ~62% of non-Stylistic line level warnings.
- Java accounts for ~90% of non-Stylistic line level warnings followed by JavaScript at ~6%.

sema

Third Party Libraries Summary





- Third-Party libraries are appropriate for efficient programming and modern code design. Instead of including a library's source code, Sema recommends the use of dependency management tools to minimize risk and ease maintenance.
- Low Risk – Less than 1.5% of code is from third-party libraries. Besides Documentation, VS Code workspace files is the most prevalent third-party library.
- Discuss: Are there compelling reasons to include third-party source code in the repositories? For example, code that cannot be updated during space transit should indeed include libraries

sema

Distributed repositories and smaller packages

Sema generally recommends that both repositories and individual packages not get too large, for ease of maintenance, adding additional features, and team management.

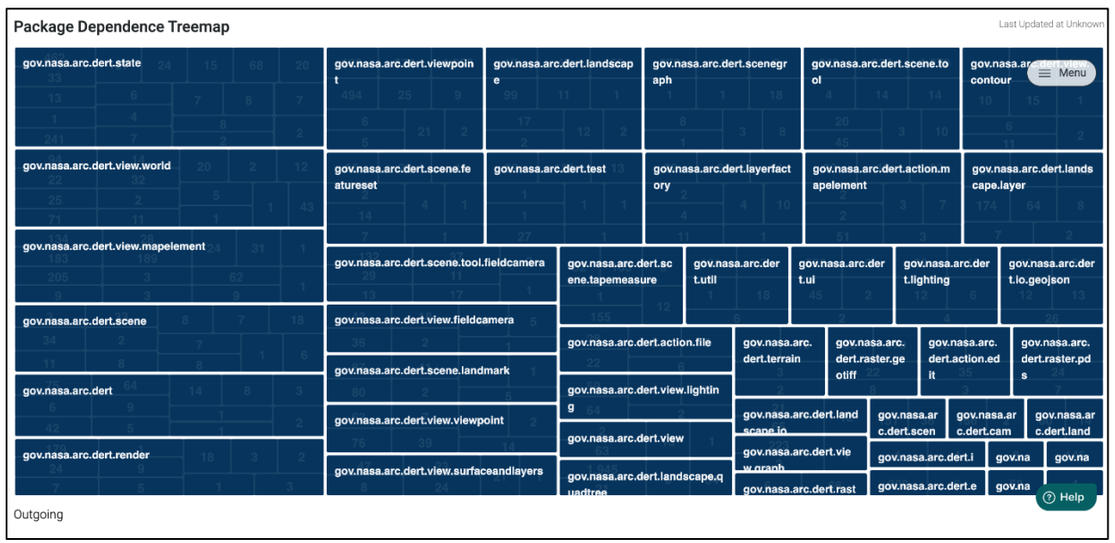
What "too large" means for each organization is highly dependent on individualized circumstances, but Sema recommends a regular review and consideration of the largest repositories and packages.

Strength – delta, DERT, Kamodo, mct were the analyzable repositories have understandable package dependency structures. The more complex repository is shown below as an example.

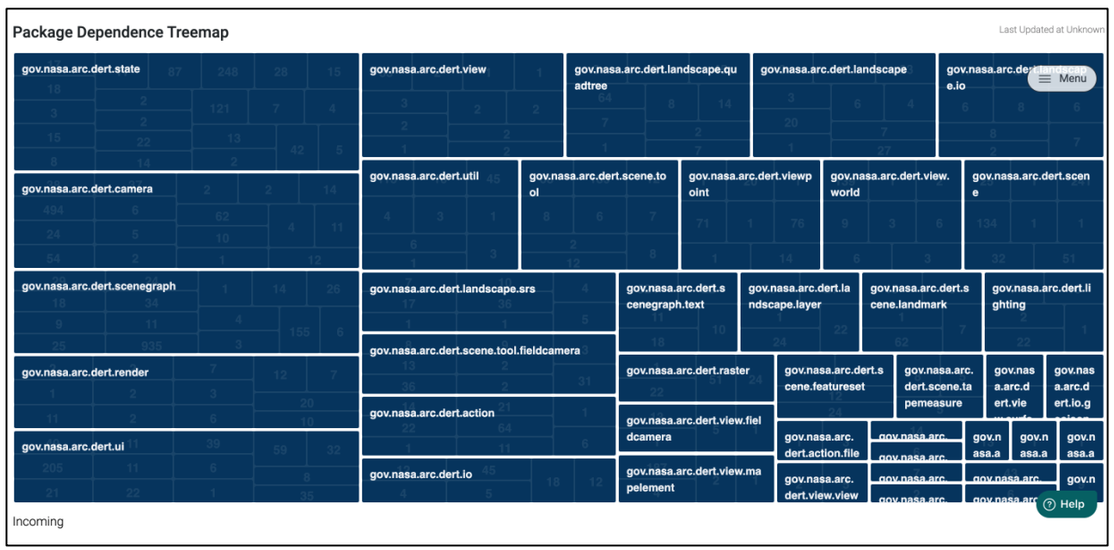
Low Risk – The "mct" repository contains 9 packages with >1000 dependencies, these packages may be candidates to be broken up. The gov.nasa.arc.mct.fastplot.bridge is shown below as an example.

Package Dependencies – DERT

Outgoing



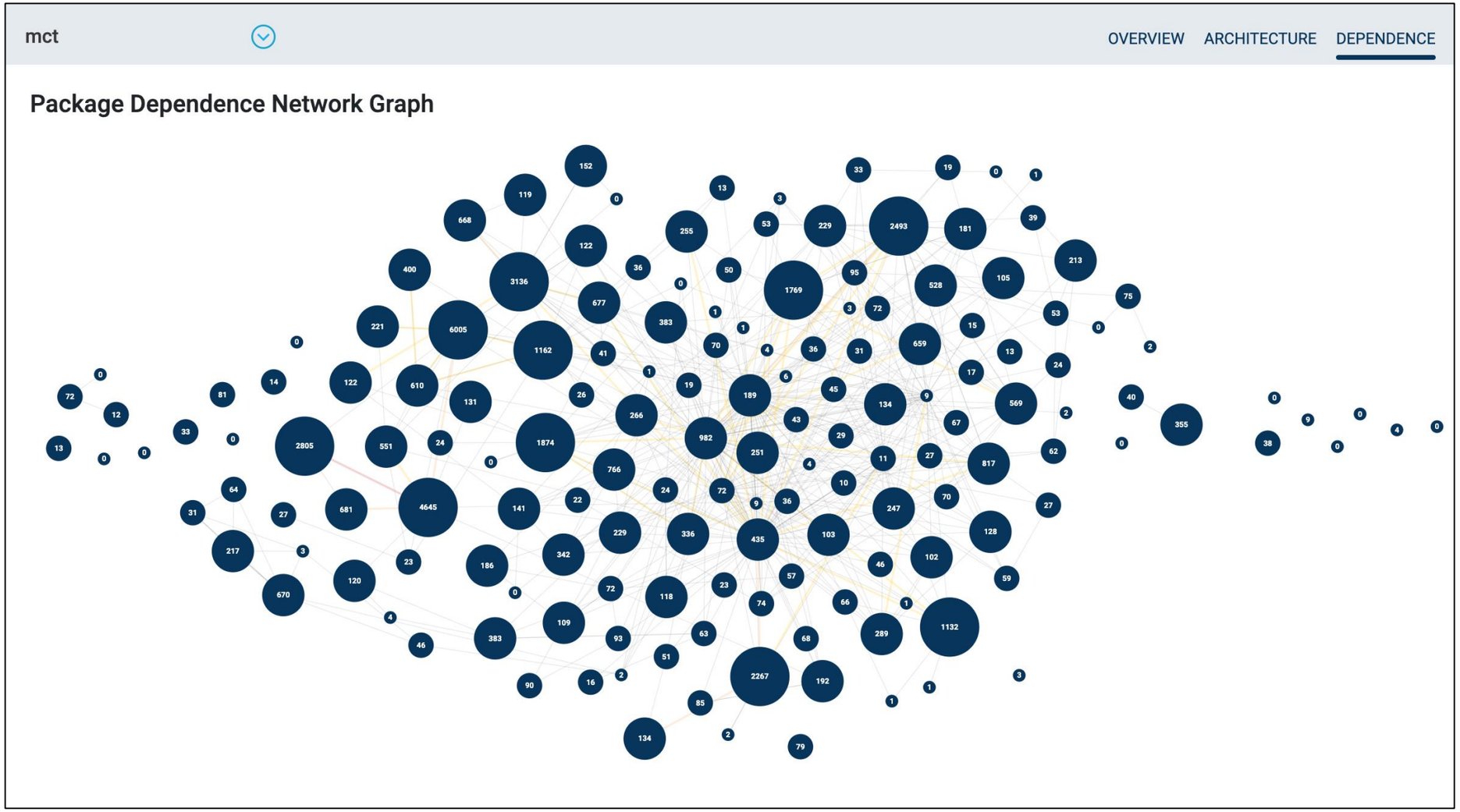
Incoming



- Sema generally recommends that an application should have few classes with high outgoing dependencies, several classes with high incoming, and key classes with middle values that represent the divisions of business logic within the application.
- Low risk– the DERT repository, shown here, as an example does not have a particularly pronounced structure and could be a candidate for architectural improvement

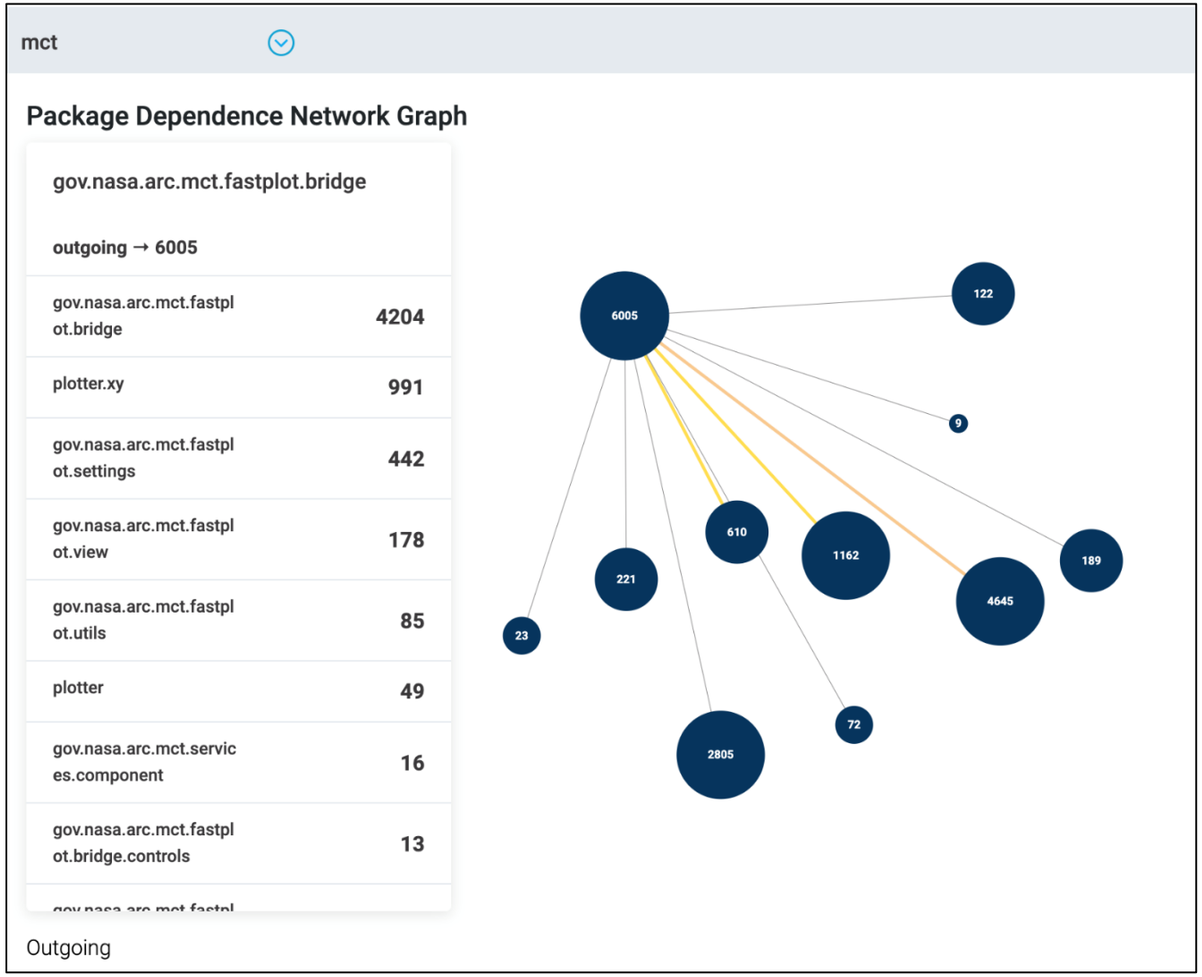
sema

Distributed repositories and smaller packages – mct



sema

Distributed repositories and larger packages - mct



- Low Risk – the "mct" repository has 10 large packages with over 1,000 self-referential or outgoing dependencies. This could be a candidate to be broken up.
- gov.nasa.arc.mct.fastplot.bridge (6005 dependencies) is shown here.
- gov.nasa.arc.mct.fastplot.bridge (4204), plotter.xy (991), gov.nasa.arc.mct.fastplot.settings (442) are some of its dependencies.

sema

Architectural Quality Indicators

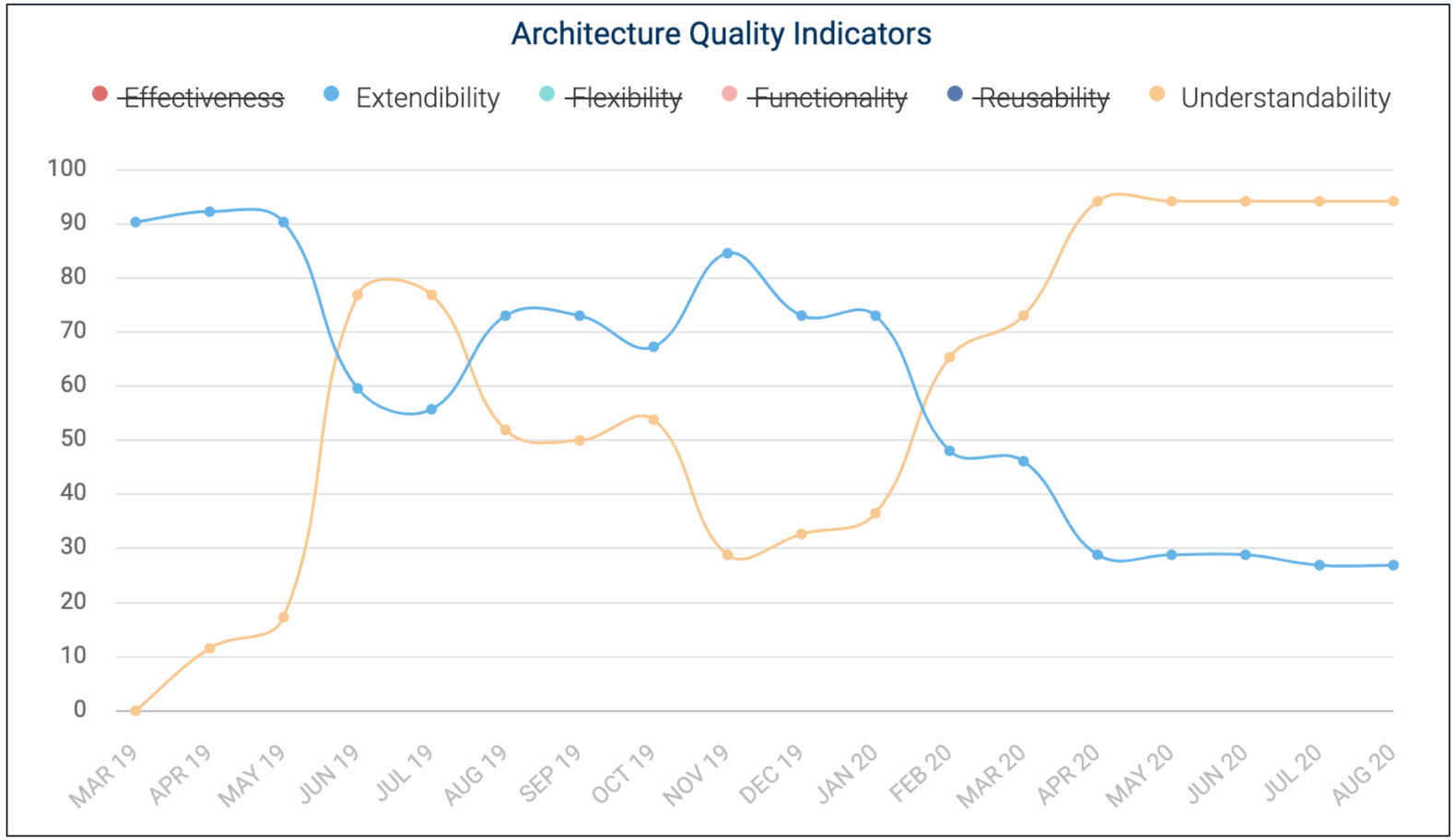
Architectural Quality Indicator	Definition	Computation
Reusability	A design with low coupling and high cohesion is easily reused by other designs.	0.25 * Coupling + 0.25 * Cohesion + 0.5 *Messaging + 0.5 * Design Size
Flexibility	The degree of allowance of changes in the design.	0.25 * Encapsulation - 0.25 * Coupling + 0.5 * Composition + 0.5 * Polymorphism
Understandability	The degree of understanding and the easiness of learning the design implementation details.	0.33 * Abstraction + 0.33 Encapsulation -0.33 * Coupling + 0.33 * Cohesion - 0.33 *Polymorphism - 0.33 * Complexity - 0.33 *Design Size
Functionality	Classes with given functions that are publicly stated in interfaces used by others.	0.12 * Cohesion + 0.22 * Polymorphism +0.22 * Messaging + 0.22 * Design Size + 0.22 * Hierarchies
Extendibility	Measurement of design's allowance to incorporate new functional requirements.	0.5 * Abstraction - 0.5 * Coupling + 0.5 * Inheritance + 0.5 * Polymorphism
Effectiveness	Design efficiency in fulfilling the required functionality.	0.2 * Abstraction + 0.2 Encapsulation + 0.2 * Composition + 0.2 * Inheritance + 0.2 * Polymorphism

Design Quality Indicators

Design Metric	Design Property	Description
Design Size in Classes (DSC)	Design Size	Total number of classes in the design.
Number of Hierarchies (NOH)	Hierarchies	Total number of "root" classes in the design. (count(MaxinheritenceTree (class)=0))
Average Number of Ancestors (ANA)	Abstraction	Average number of classes in the inheritance tree for each class.
Direct Access Metric (DAM)	Encapsulation	Ratio of the number of private and protected attributes to the total number of attributes in the class.
Direct Class Coupling (DCC)	Coupling	Number of other classes a class relates to, either through a shared attribute or a parameter in a method.
Cohesion Among Methods of Class (CAMC)	Cohesion	Measure of how related methods are in a class in terms of used parameters. It can be computed by: 1 - LackOfCohesionOfMethods()
Measure of Aggregation (MOA)	Composition	Count of number of attributes whose type is user defined classes.
Measure of Functional Abstraction (MFA)	Inheritance	Ratio of the number of inherited methods per the total number of methods within a class.
Number of Polymorphic Methods (NOP)	Polymorphism	Any method that can be used by a class and its descendants. Counts of the number of methods in a class excluding private, static, and final ones.
Class Interface Size (CIS)	Messaging	Number of public methods in a class.
Number of Methods (NOM)	Complexity	Number of methods declared in a class.

sema

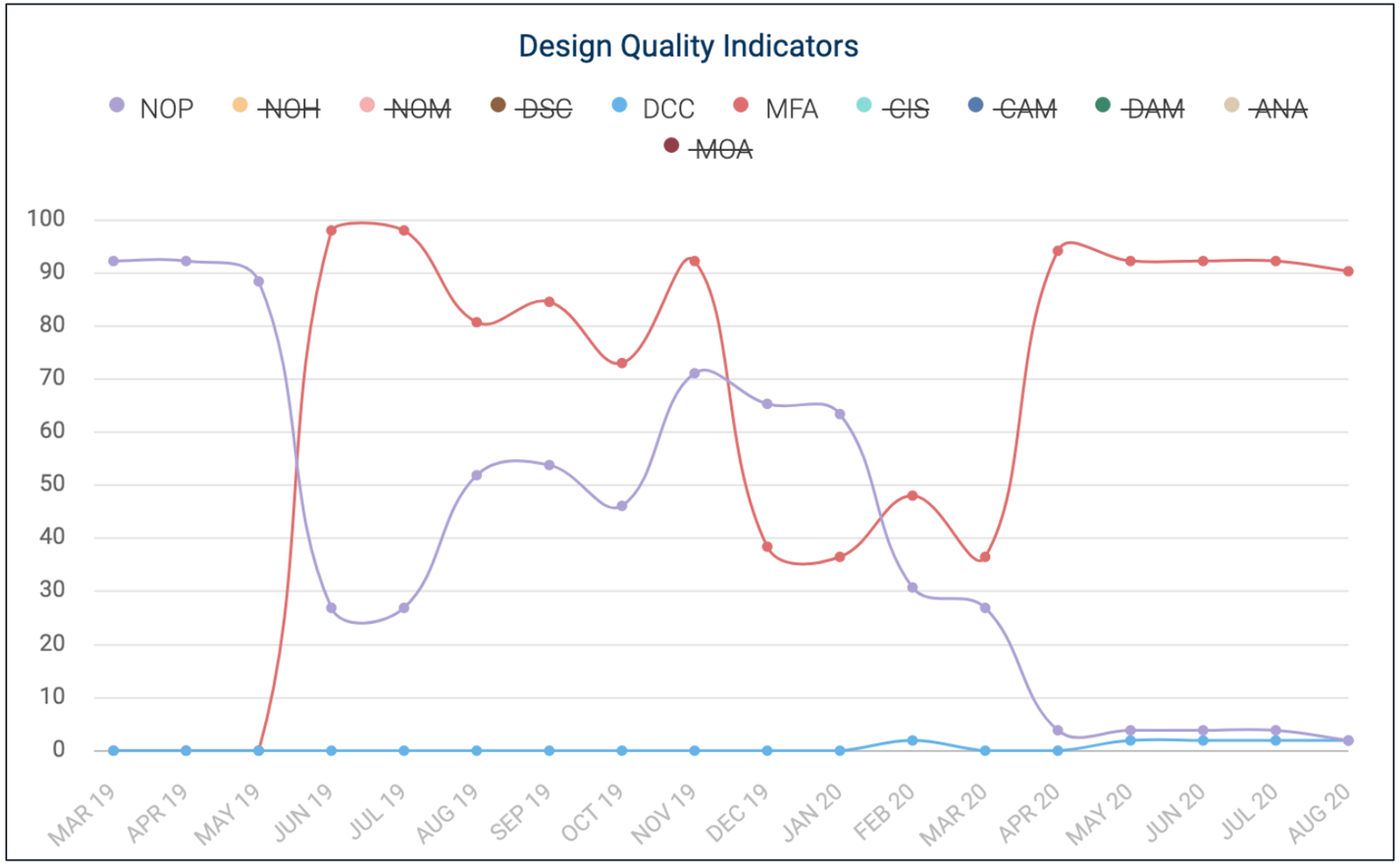
Architectural Quality Indicators – delta



- Architectural quality has fluctuated considerably through the history of this repository. Along with its short history this fluctuation indicates this repository under development
- These two metrics move in opposite directions due to polymorphism. Less polymorphism drives Understandability higher and Extedibility lower.
- Another Design Quality Indicator – Inheritance – positively impacts Understandability but does not impact Extendibility.

sema

Design Quality Indicators – delta



- Polymorphism(NOP) has fluctuated through this development has remained low in the last few months.
- Coupling (DCC) remained consistently low.
- Inheritance (MFA) has fluctuated as well but has remained high in the last few months

sema

Code

Process

Team

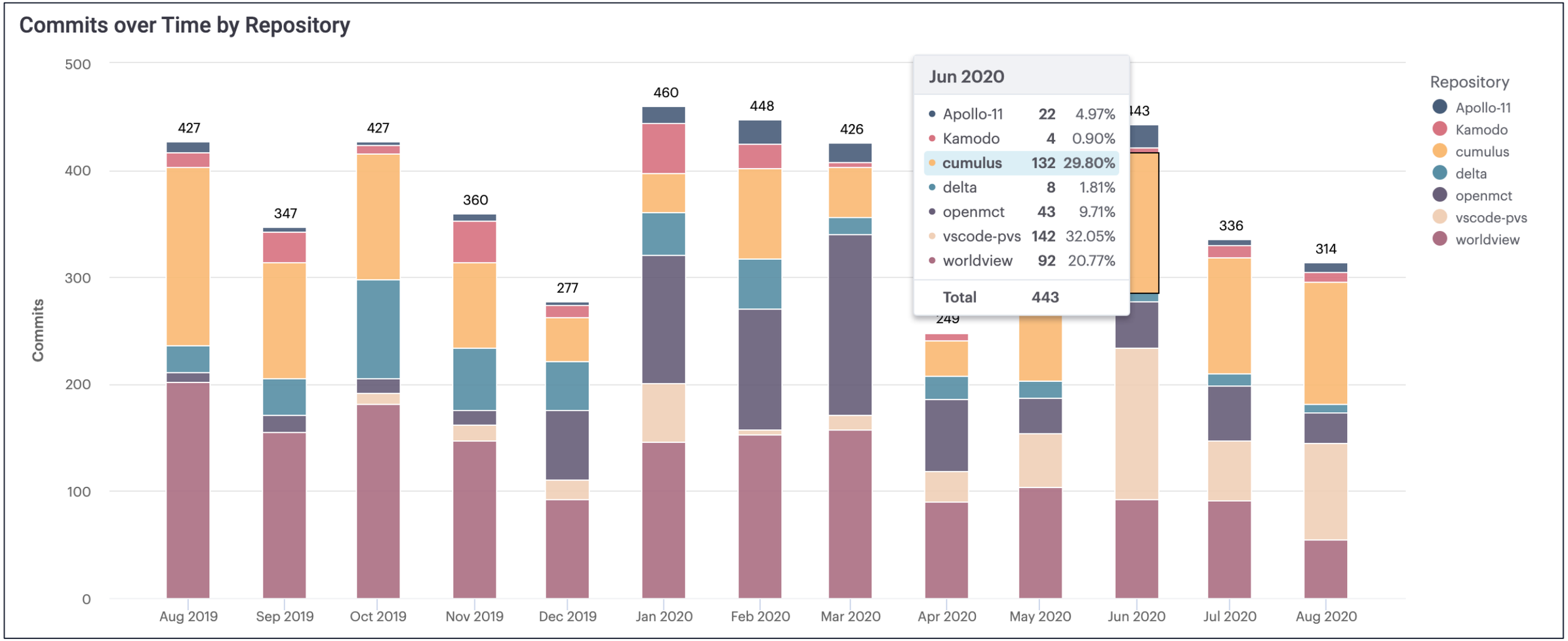
sema

Commit Analysis

Sema recommends setting consistent commit activity goals over time and manage towards them, with respect to code priorities.

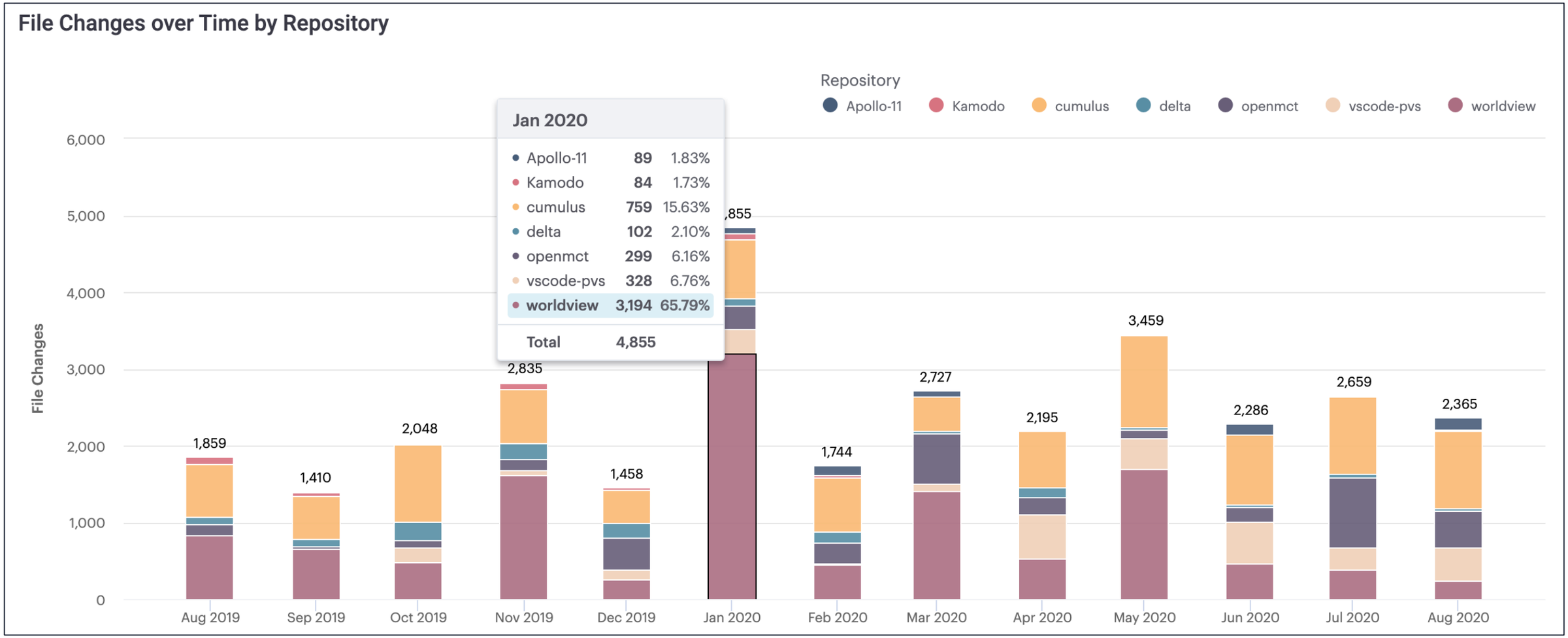
Discuss two particular patterns: decline in December 2019, likely due to holiday breaks, and in April 2019, likely due to COVID disruption.

Commit Analysis – Commits Last Full Year



sema

Commit Analysis – File Changes Last Full Year



sema

Commit Management

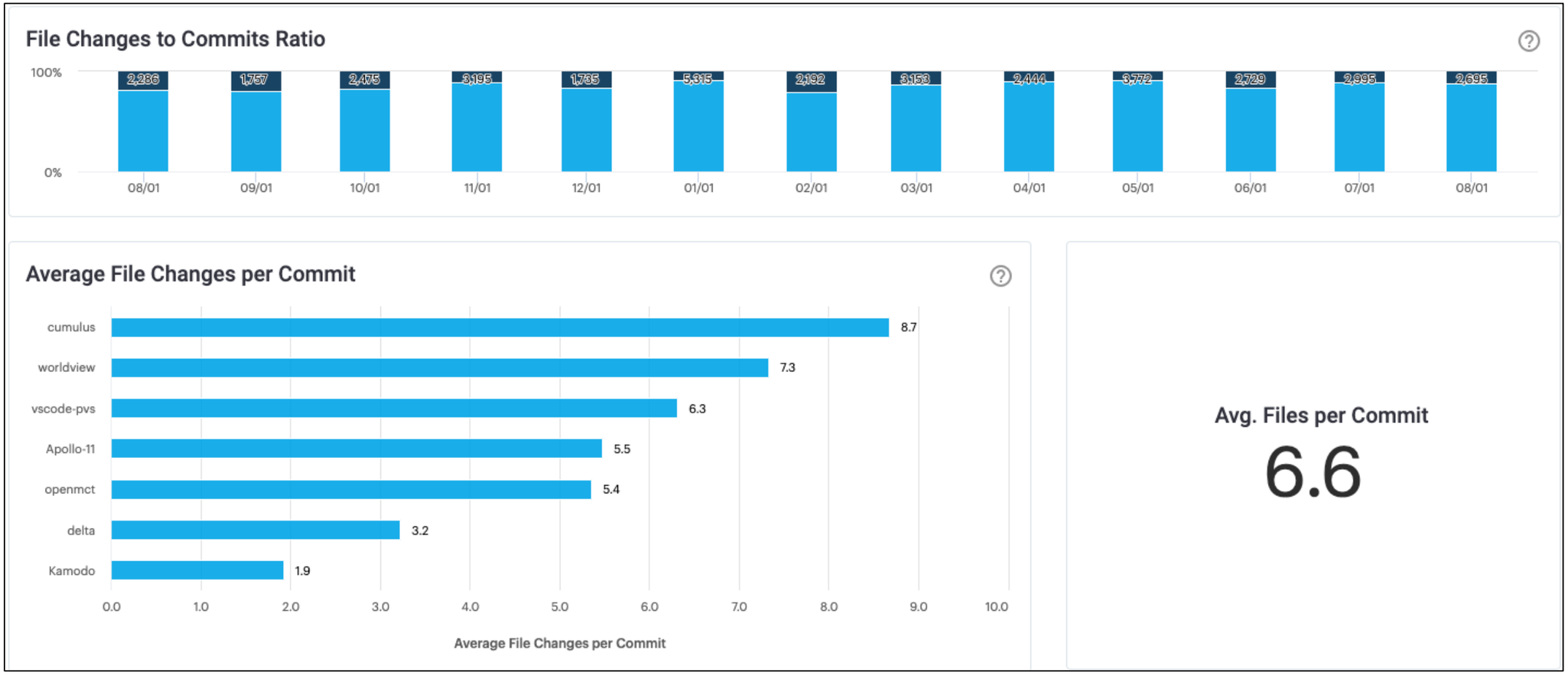
A good process for committing code improves the quality and ease of maintenance of a code base. Sema analyzes an aspect of this, the average number of files per commit.

Sema recommends keeping files per commit low (<5) to make maintenance/future changes easier. Sema considers 10-25 files per commit as a risk, and organizations should look to investigate further. Over 25 files per commit is likely due to administrative changes.

Low Risk – 3 of 7 repositories average more than 5 files per commit over the past year – "cumulus" (8.7 files per commit), worldview (7.3), vs-code-pvs (6.3). 3 of 7 repositories average more than 5 files per commit over the past 90 days (openmct, cumulus, and Apollo-11).

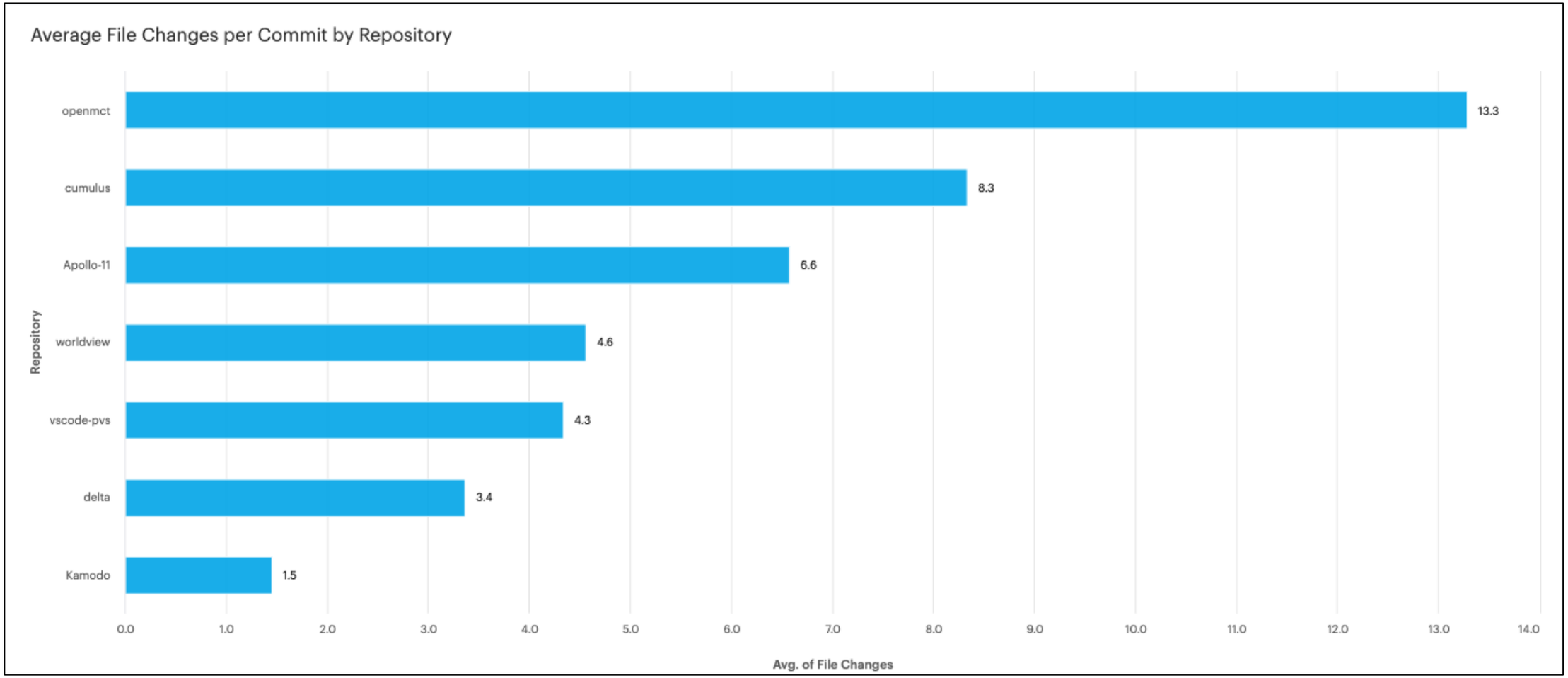
Low Risk – 8 developer averages more than 10 files per commit over the past 90 days (Chuck Daniels, and Joel McKinnon are the top 2 with more than 62 files per commit), 17 average more than 10 over the past year (Chuck Daniels, and Juanisa McCoy are the top 2 with more than 35 files per commit).

File Changes per Commit Ratio – Last Full Year



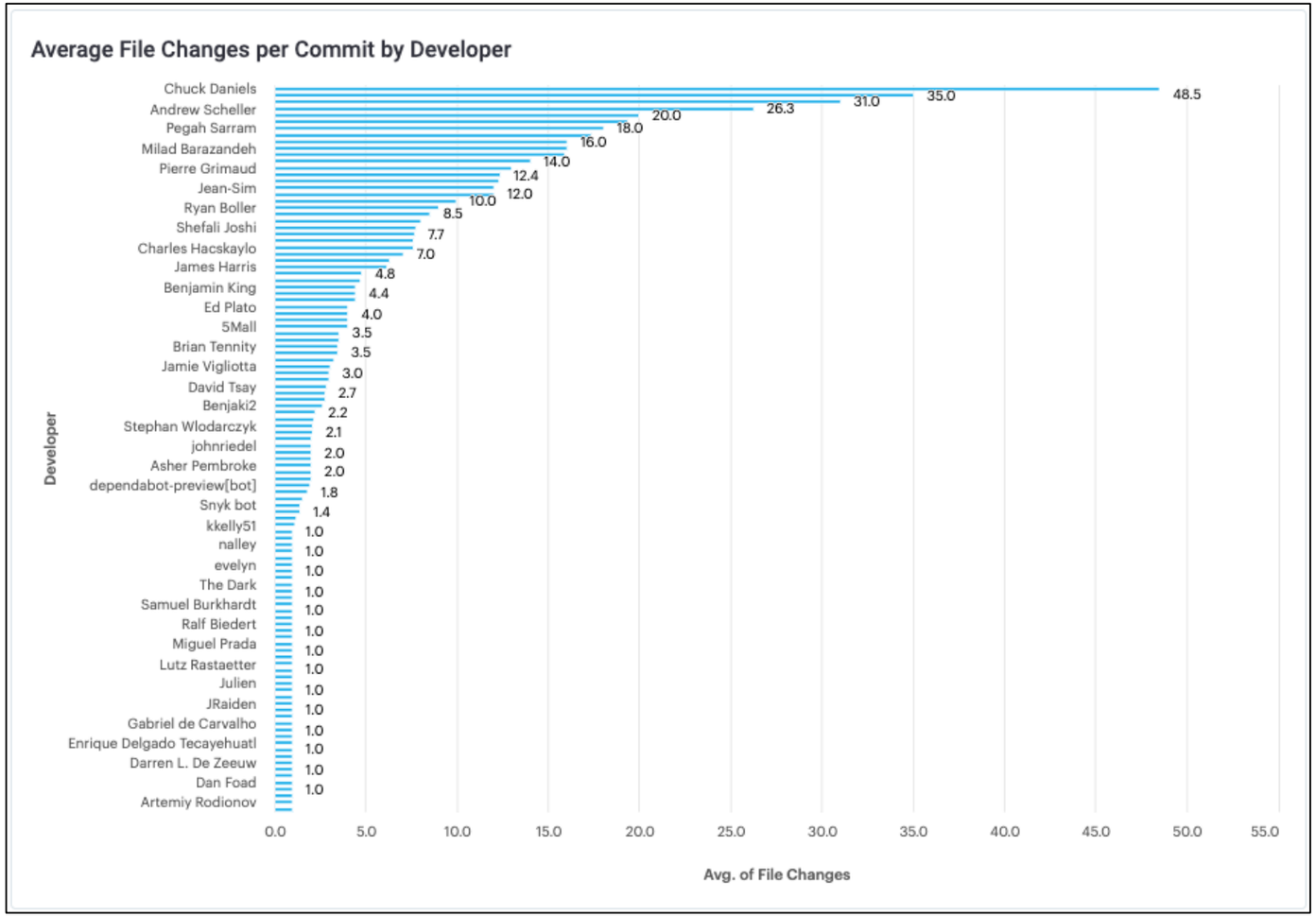
sema

File Changes per Commit – Last 90 Days



sema

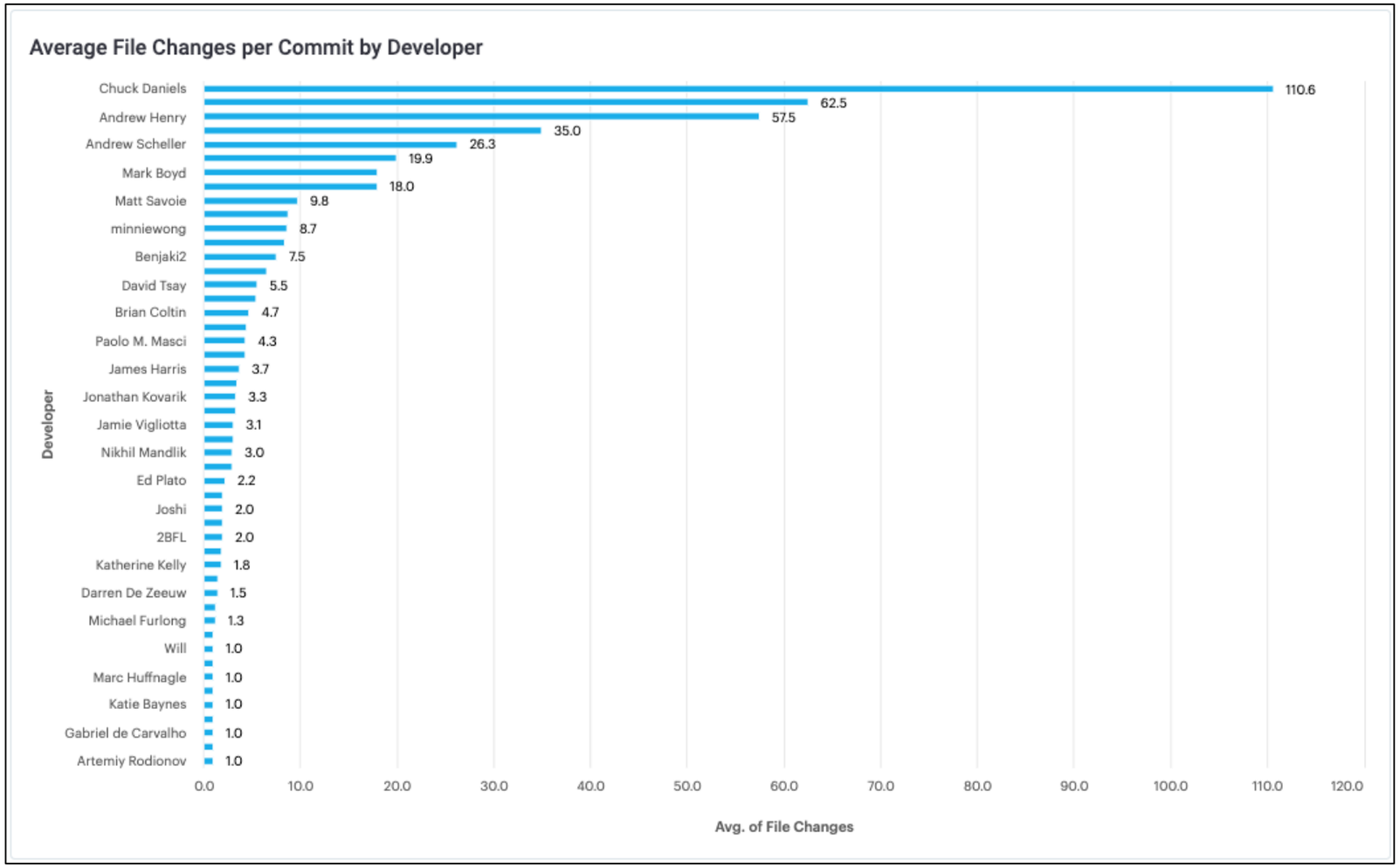
Average File Changes Per Commit by Developer – Last Full Year



- Low Risk- review commit practices for developers with files per commit > 10 - Chuck Daniels average is the top of the File changes with ~48 followed by Juanisa McCoy with 35 – and see if there is a development rationale. If not, coach towards fewer files per commit

sema

Average File Changes per Commit by Developer – Last 90 Days



sema

Ticket Reference and Test Reference Management

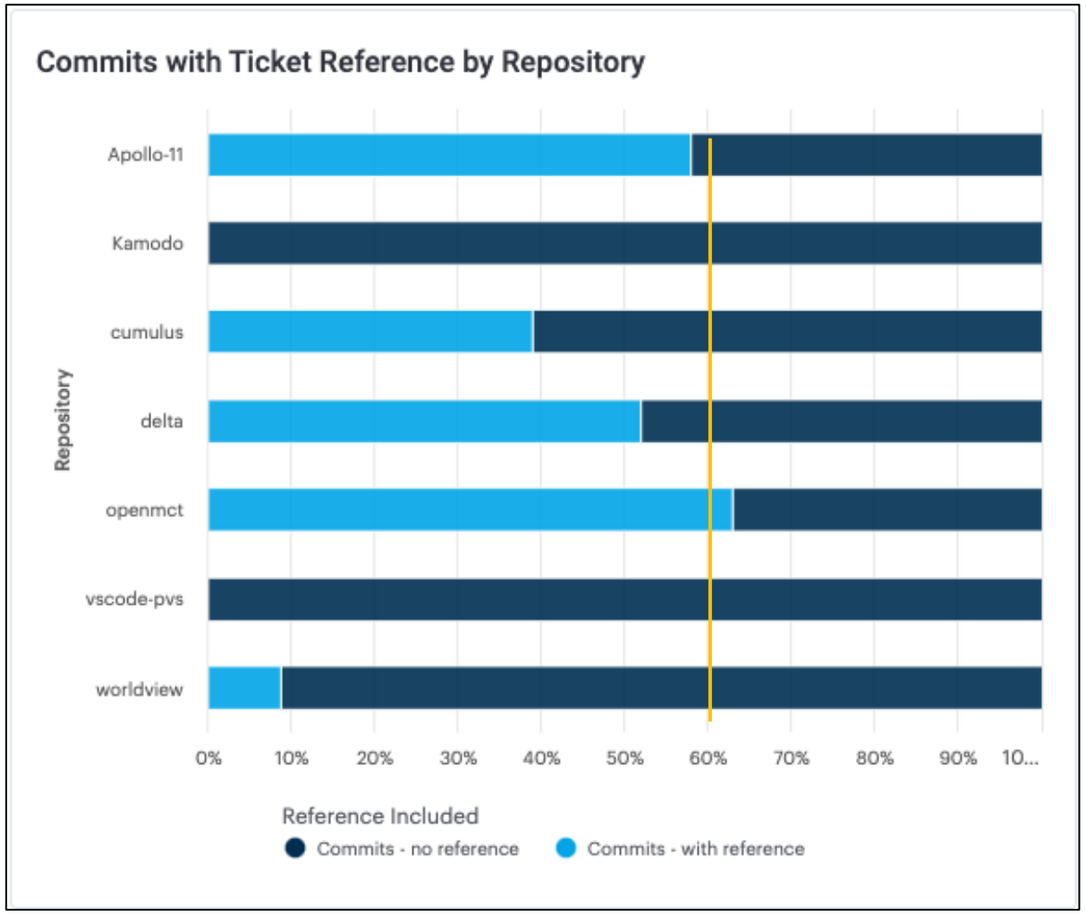
Adherence to a process of committing code based on associated tickets allows for easier troubleshooting and the ability to understand the reason for code changes.

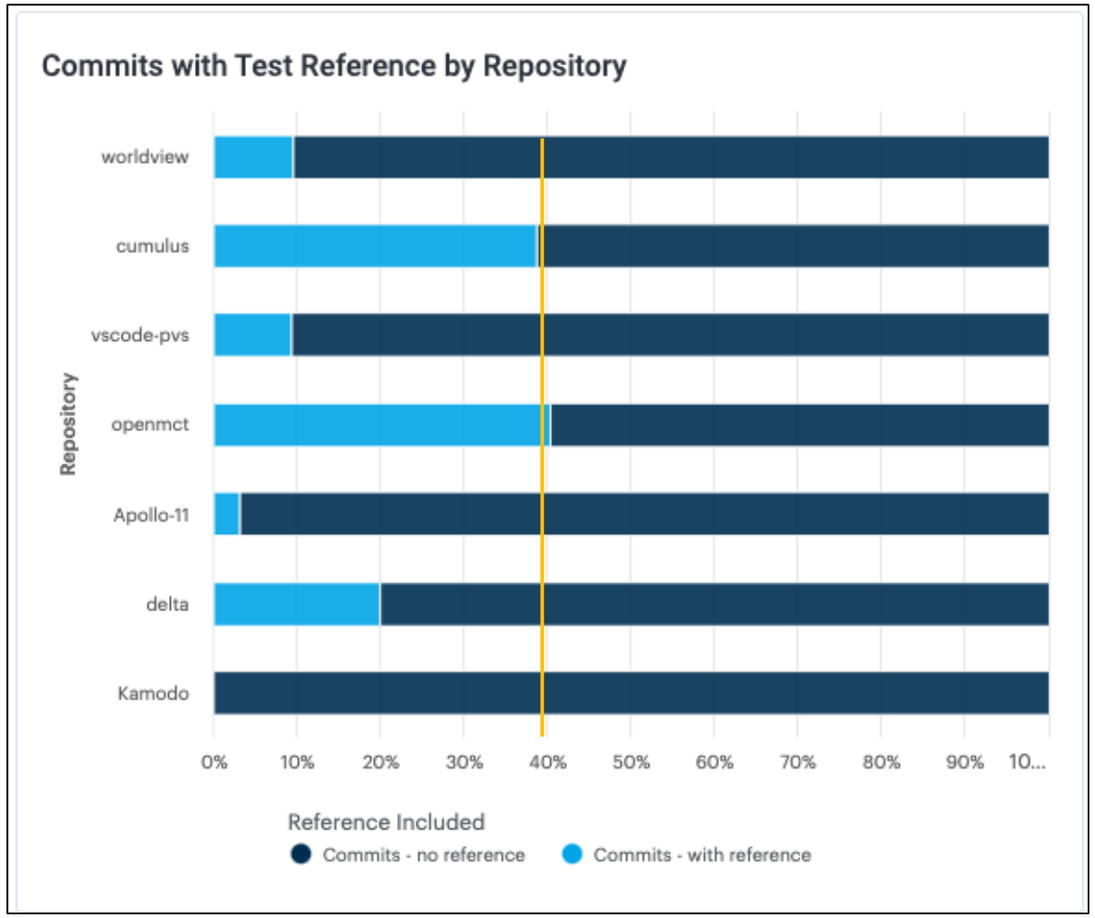
Adding testing as development occurs is the one of the easiest ways to minimize technical debt and drive quality in the code.

Sema recommends that over 60% of commits have ticket references and over 40% of commits have test references so that testing is included as development occurs.

Medium Risk – 1 of 7 repositories meets the recommended ticket and test reference benchmark over the past 90 days (openmct). Are these acceptable levels in light of the code being open source?

Commits with Ticket References and Test Commits – Past 90 Days





sema

Repository Access Management

As a risk avoidance measure, employees and contractors should only be allowed access to the repository with approved accounts.

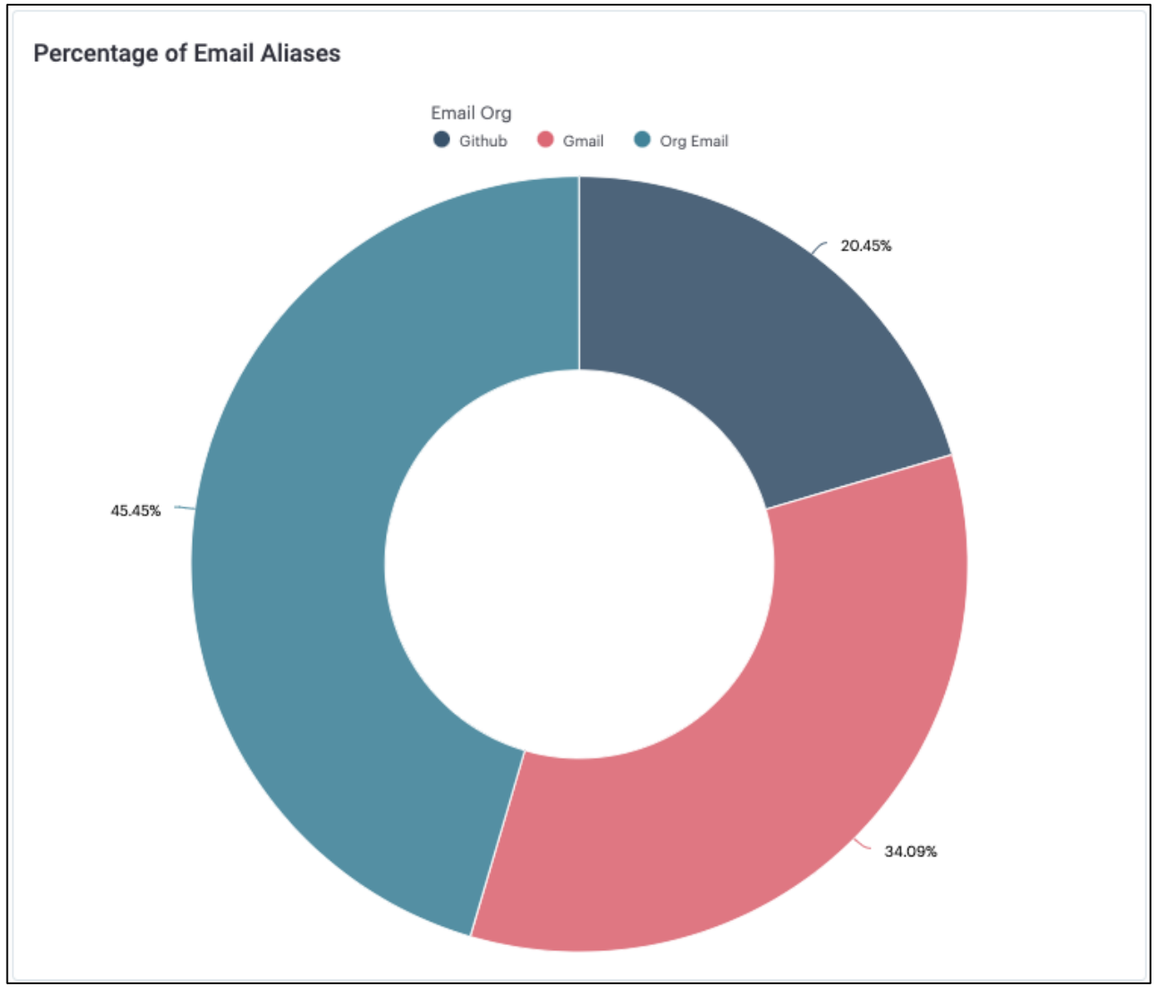
Organization aliases – such as the current company or previous acquisitions – are appropriate.

GitHub aliases are also appropriate. Commits with GitHub in their email address have performed command line comments with their email address set to private.

Generic email addresses such as Gmail, Yahoo, and Microsoft Live should be avoided.

In open source code generic email addresses are expected, they would be problematic in private code.

Email Aliases Used in Commits – Past 90 Days



Observation: ~54% commits in the past 90 days have come from non-organization accounts (29).

In open source code generic email addresses are expected, they would be problematic in private code.

sema

Code

Process

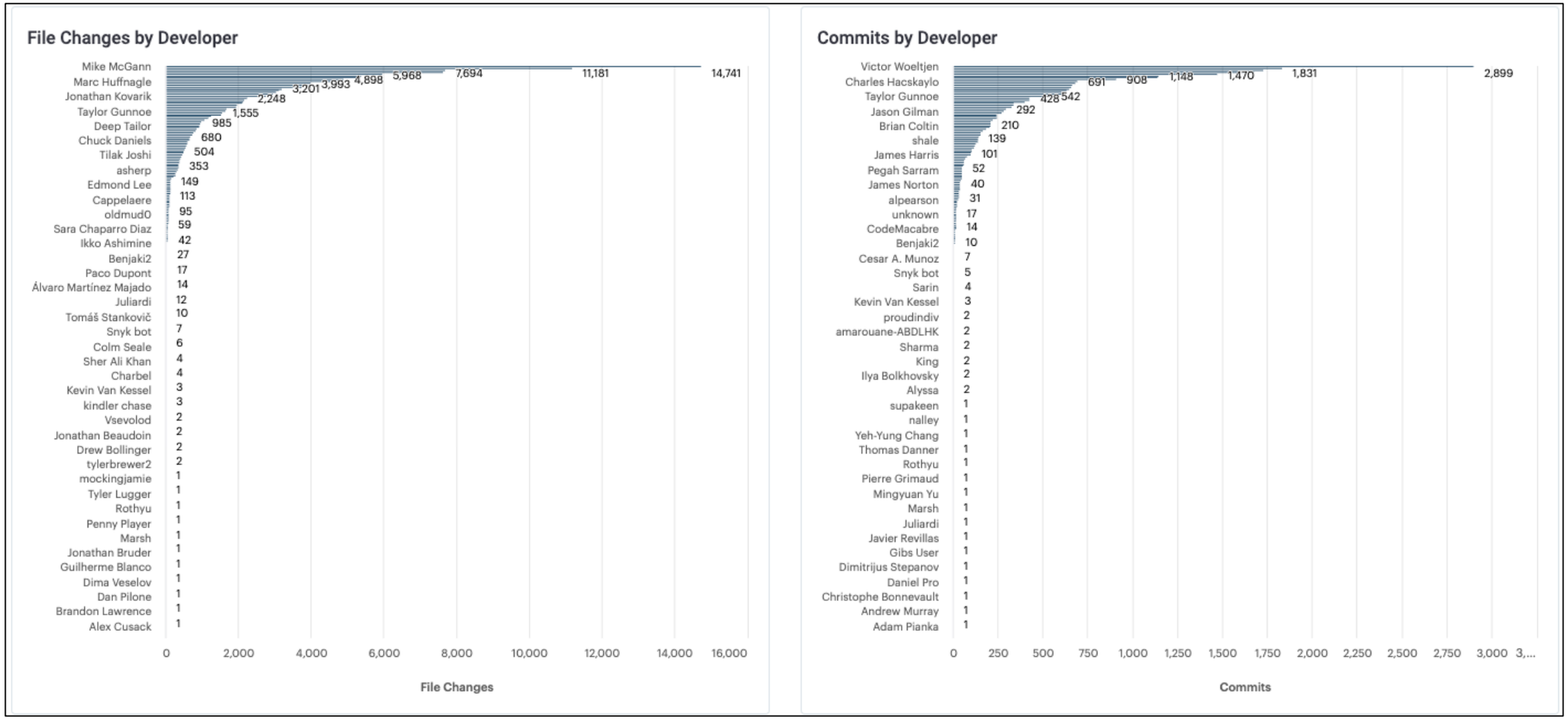
Team

sema

Developer Activity and Expertise

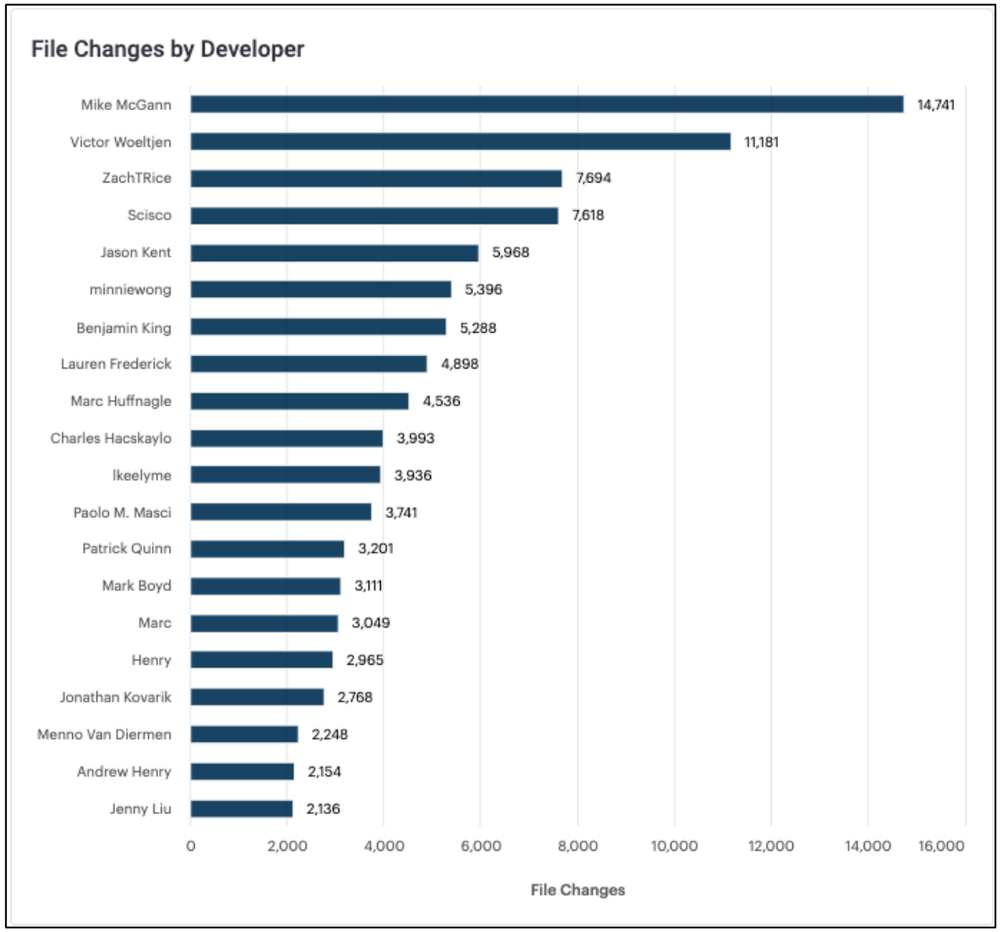
- Sema recommends setting consistent commit activity goals over time and manage towards them, with respect to code priorities.
- Based on current agile methodology, Sema recommends maximizing the number of full stack developers on a team.
- Victor Woeltjen is the top contributor to these repositories in terms of total commits for all time and Paolo M. Masci in the past 90 days.
- Mike McGann is the top contributor to these repositories in terms of file changes, both all time and in the past 90 days.

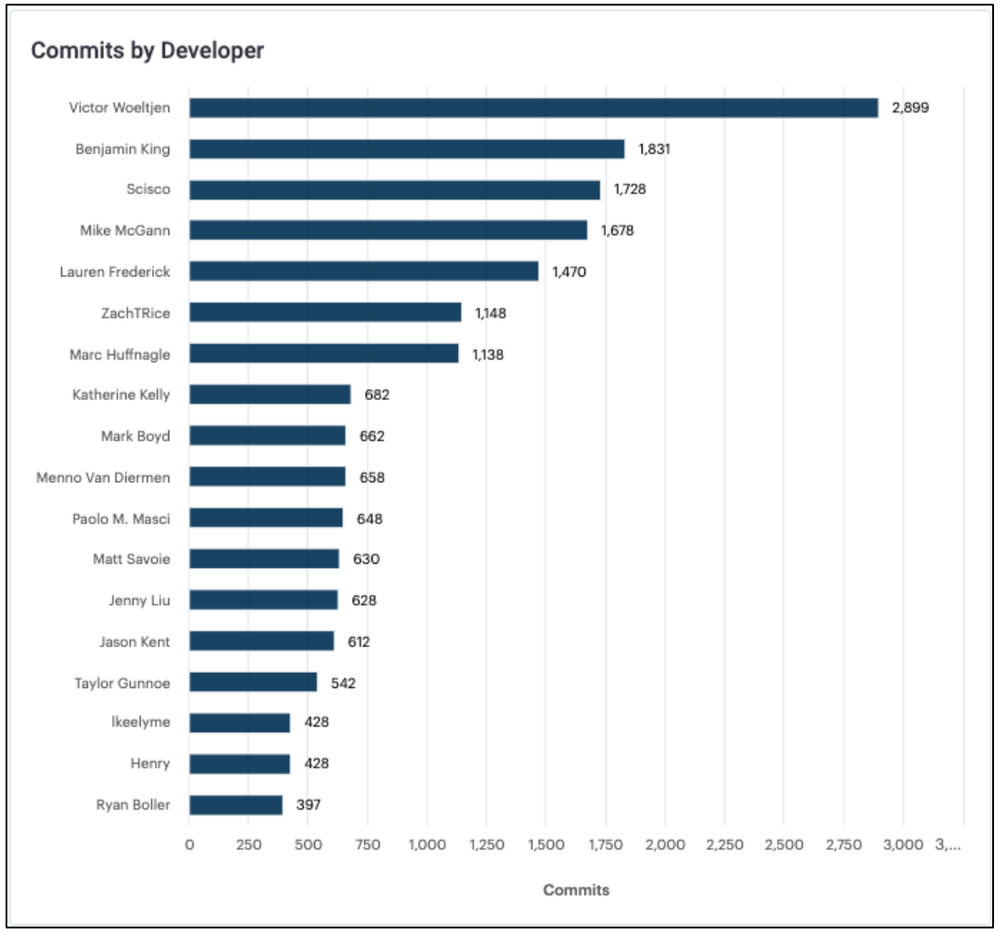
Commits and File Changes by Developer – All Time



sema

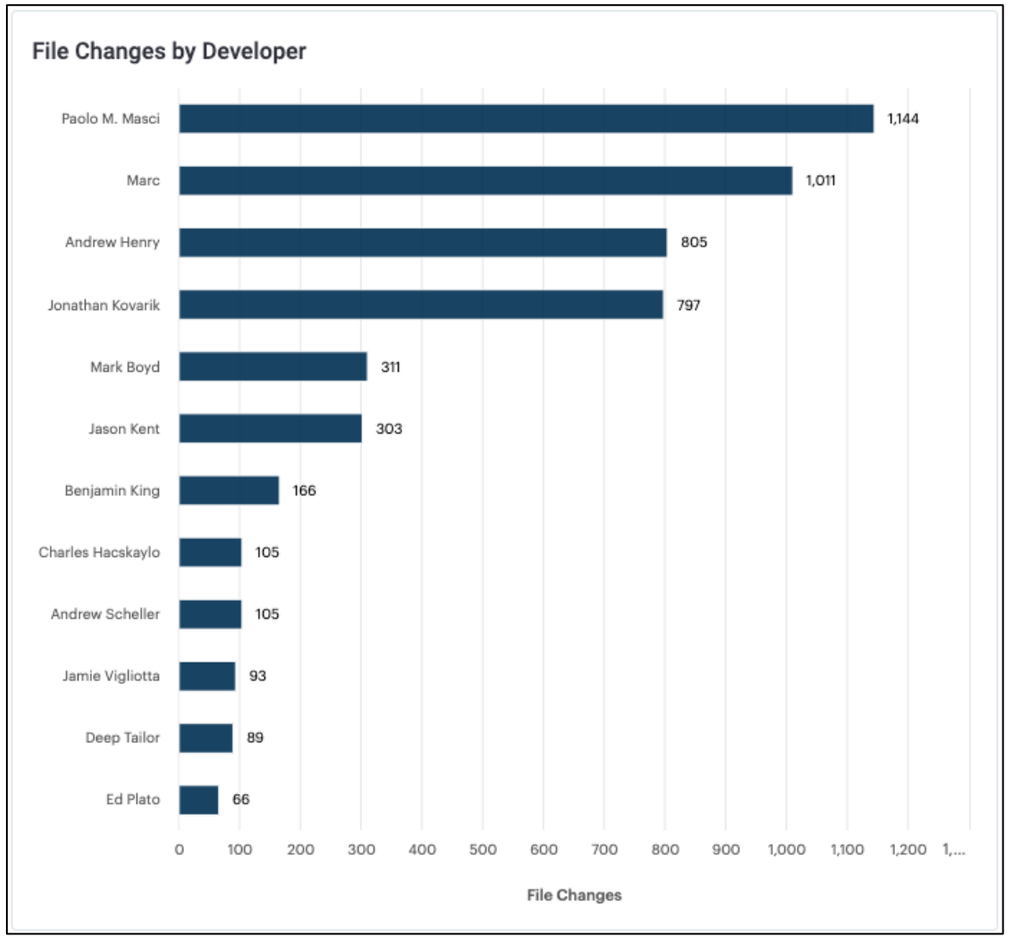
Commits and File Changes by Developer – All Time, Top 20

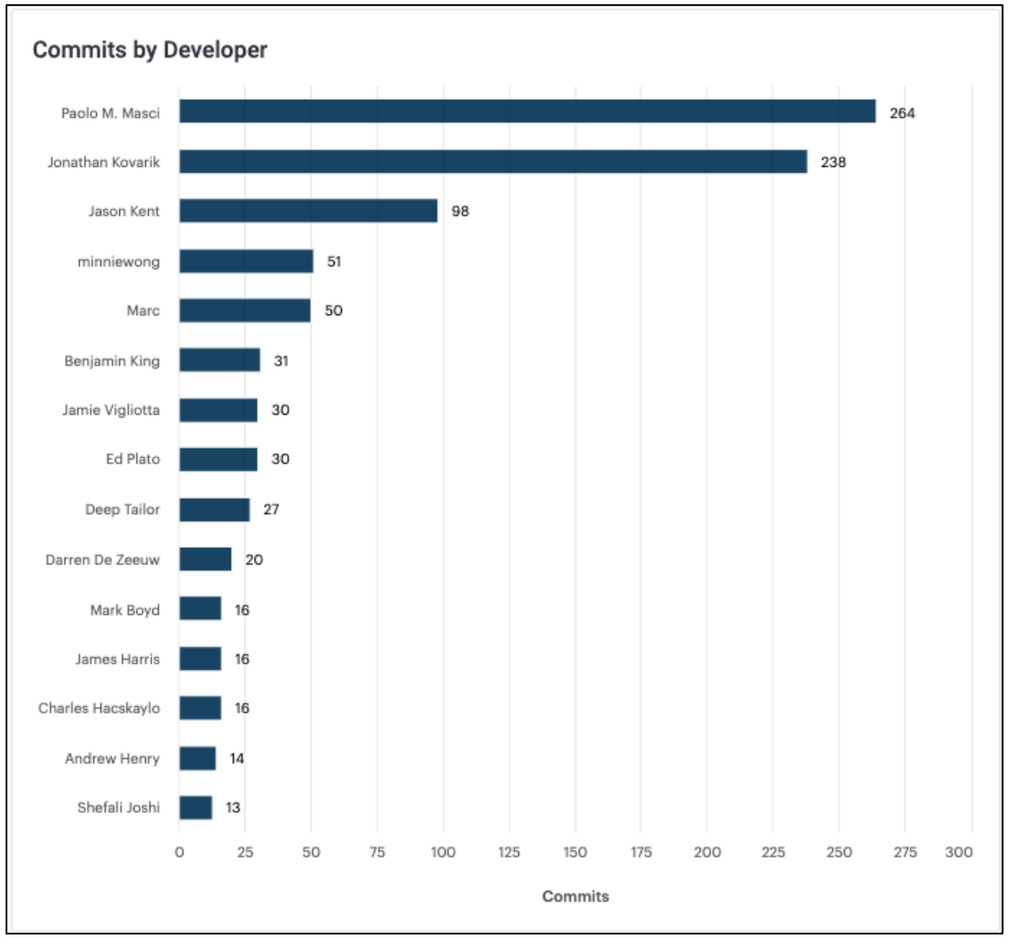




sema

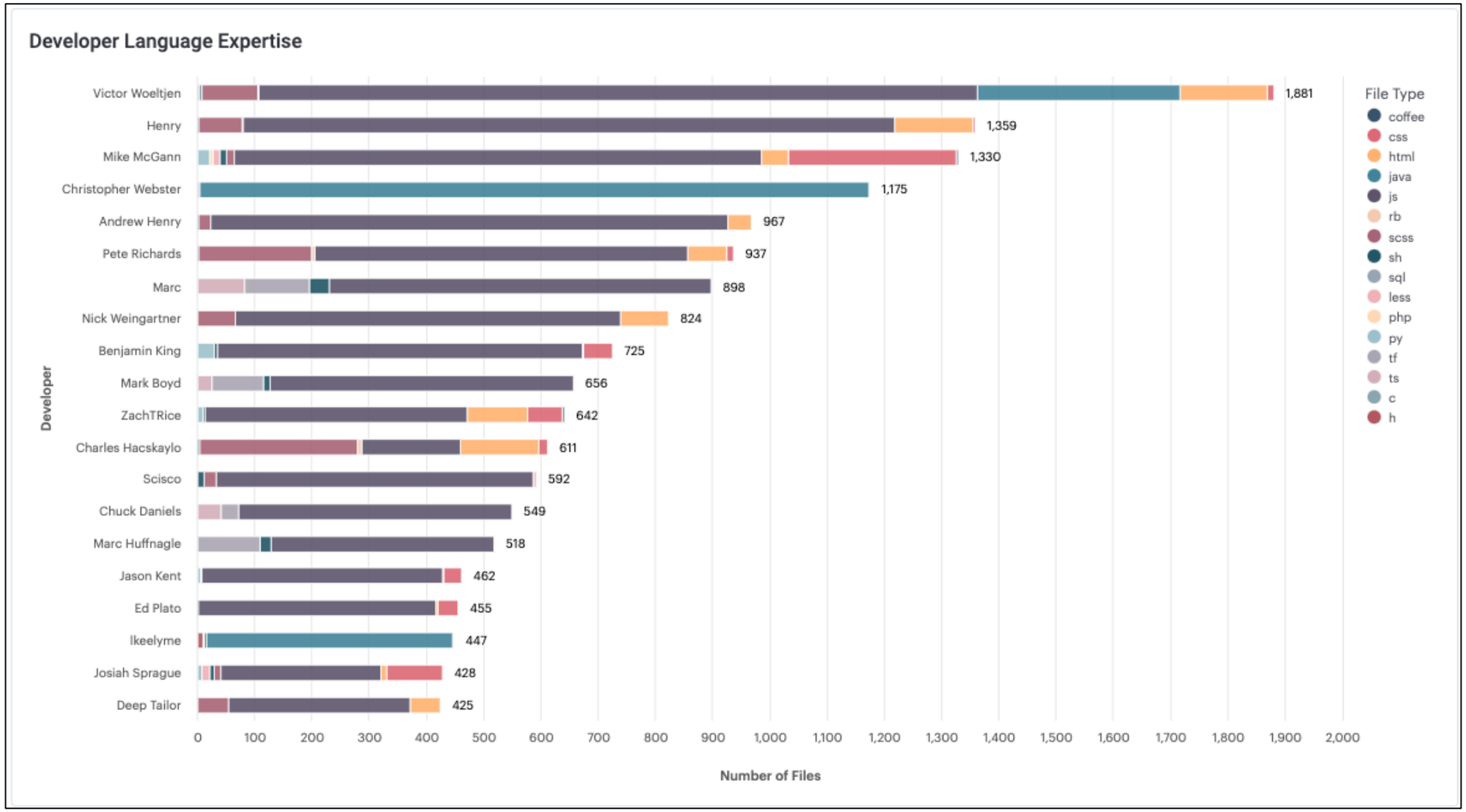
Commits and File Changes per Developer (Top 15) – 90 Days





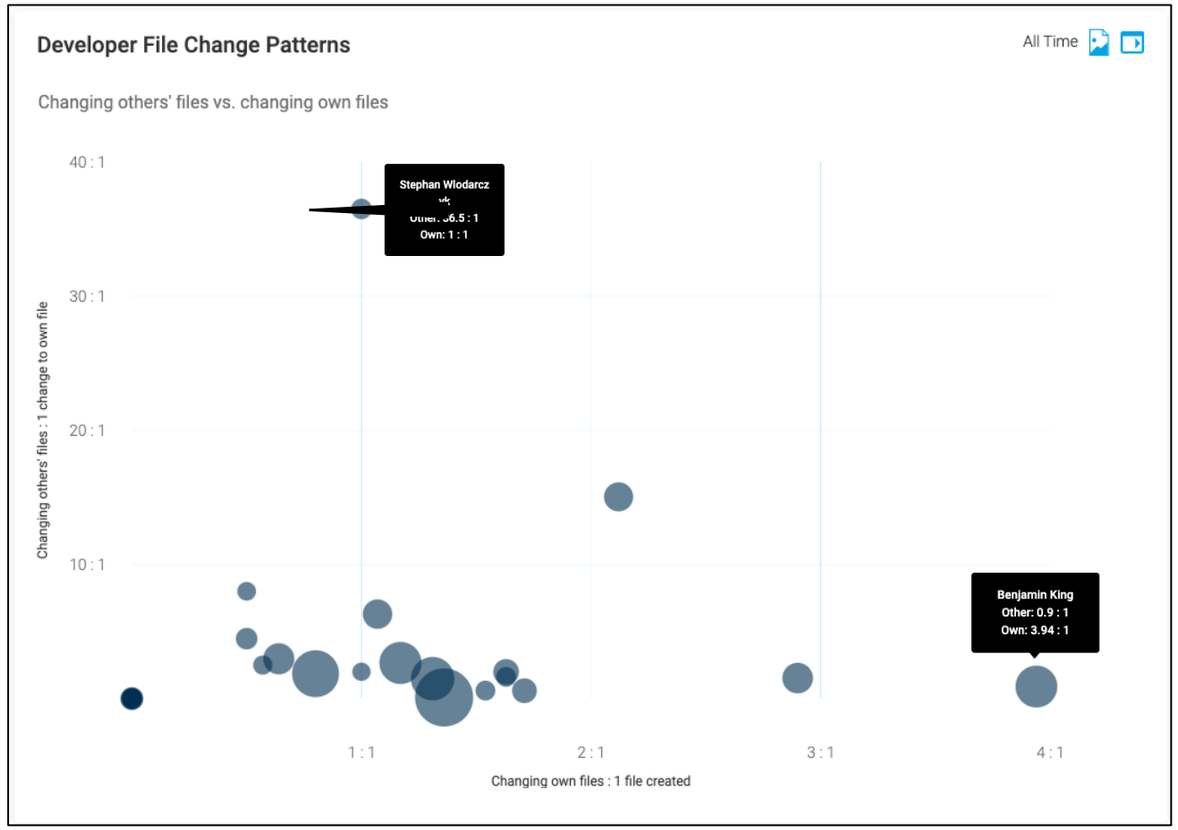
sema

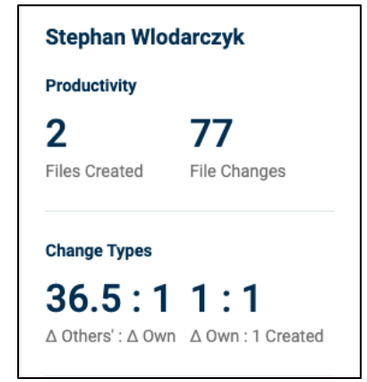
Developer Expertise – All Time (Top 20)

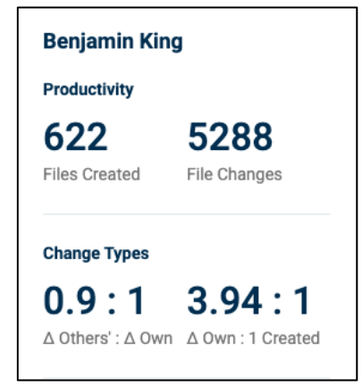


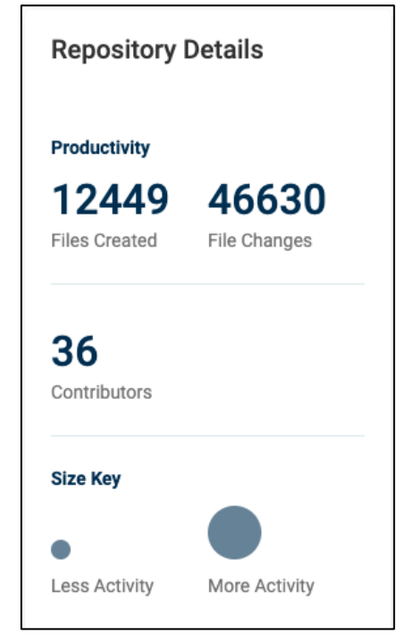
sema

Contribution Pattern – worldview









- Benjamin King is a "Self-Editor" who changes their own code ~3.94x more than creating code.
- Stephan Wlodarcyyk is an "Editor" who changes others code ~36x more than their own code. Is he maintaining code?

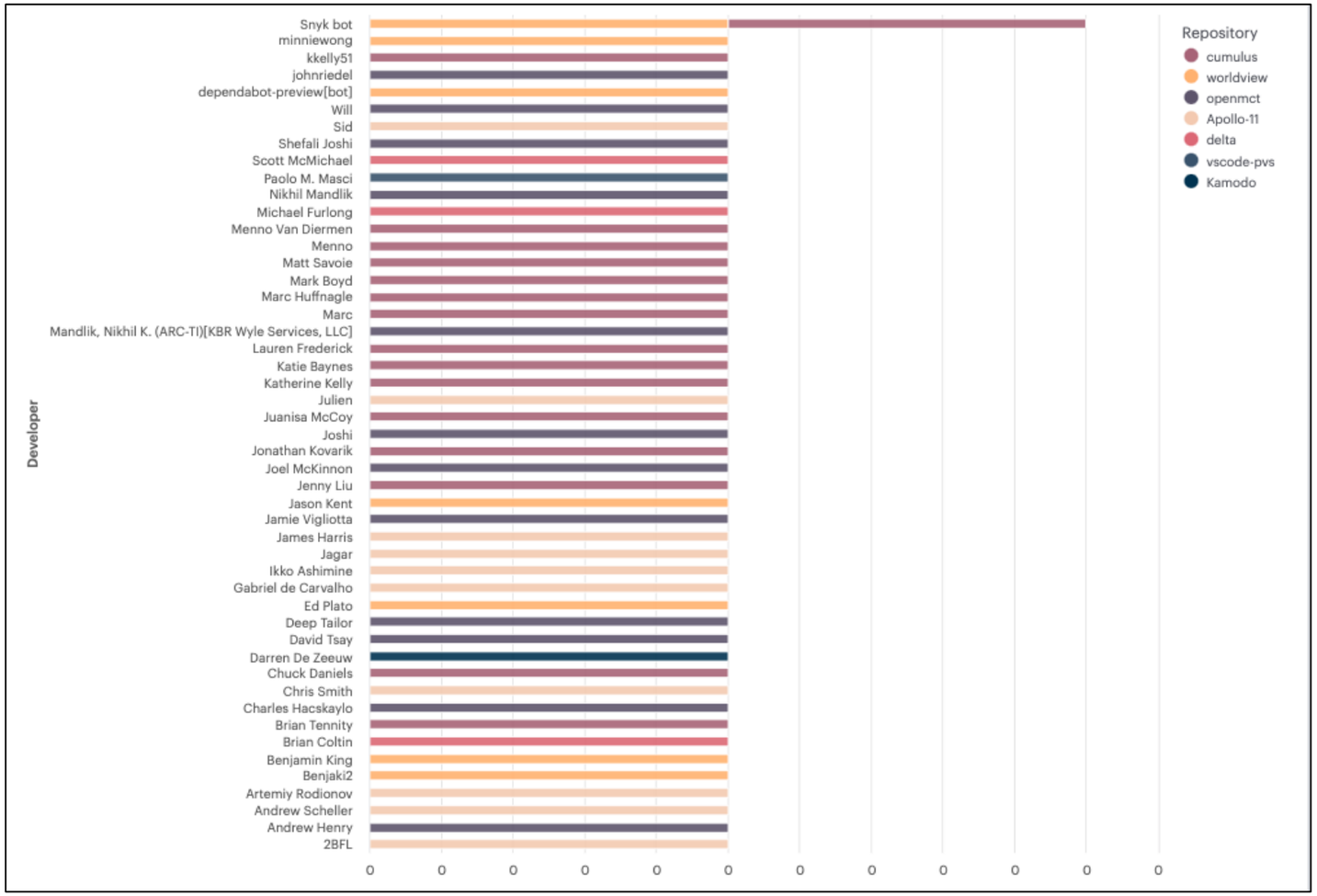
sema

Contribution Detail Top Contributors – worldview

		# Files Created or Changed				File Change Patterns
St...	Committer	Total ↓	Files Created	Change Own	Change Others'	Δ Own : 1 File Created
○	Mike McGann	14741	5927	8071	743	1.36
●	ZachTRice	7694	2333	1873	3488	0.8
●	Jason Kent	5968	1397	1837	2734	1.31
●	minniewong	5396	1023	1194	3179	1.17
●	Benjamin King	5288	622	2452	2214	3.94
○	Taylor Gunnoe	1640	465	296	879	0.64
●	Edward Plato	1520	182	528	810	2.9
○	Josiah Sprague	1251	142	152	957	1.07
●	Ed Plato	1189	34	72	1083	2.12
●	Ryan Boller	595	101	165	329	1.63
○	Tilak Joshi	468	126	215	127	1.71
●	dependabot-preview[bot]	266	0	0	266	
○	Edmond Lee	145	25	0	120	0
○	Beth Timmons	142	38	19	85	0.5
●	Stephan Wlodarczyk	77	2	2	73	1

sema

Contributors to Repositories – top 15 - Last 90 Days



- Snyk bot have commits in 2 different repositories over the past 90 days. 48 other developers have committed to 1 repository over the past 90 days.
- Where applications are written by teams of developers, there would be much more collaboration than is the case here, with distinct teams.

sema

Overview of Developer Coaching Grid: 23 sets of metrics

What we measure

- Contribution: how much work is each developer doing, and how are they spending their time?
- Skill: who writes the most/ least clean code? Who is making the most/ least positive impact on code architecture?

How we calculate

- Time series review of which developers have written which code
- Delta of the quality of the code at each commit compared to previous code
- Rollup into objective and comparative ranking for each developer

sema

Metrics Definitions: 9 for Contribution

Sub-category	Metric name	Metric description
Activity	first_commit_count	Average Count of First Commits to a repository
Activity	changes_to_others_code_count	Average Count of Changes to other developer's
Activity	changes_to_own_code_count	Average Count of Changes to a developer's own code
Activity	changes_to_other_code_perc_of_initial	Average ratio of code other's code changed vs created
Activity	changes_to_own_code_perc_of_initial	Average ratio of own code changed vs created
Activity	changes_to_any_code_perc_of_initial	Average ratio of any code changed vs created
Activity	files_created_count	Average number of files created within a repository
Expertise	total_exp_per_repo	Percent of Expertise for the repositories the Develper has committed to
Expertise	total_exp	Percent of Expertise compare to all repositories

sema

Metrics Definitions: Up to 14 for Skill (language-specific)

Sub-category	Metric name	Metric description
Architectural	Effectiveness	Design efficiency in fulfilling the required functionality
Architectural	Extendibility	Measurement of design's allowance to incorporate new functional requirements
Architectural	Flexibility	The degree of allowance of changes in the design
Architectural	Functionality	Classes with given functions that are publicly stated in interfaces used by others
Architectural	Reusability	A design with low coupling and high cohesion is easily reused by other designs
Architectural	Understandability	The degree of understanding and the easiness of learning the design implementation details
Line Level	Environment Sensitive	Warnings that are relevant depending on whether strict mode is turned on or not. Issues that may affect functionality in the future. As the underlying platform evolves, the use of deprecated features may be risky, given that they may dissappear in the future
Line Level	Misleading	Issues that may mislead developers in such a way as to introduce bugs in the future
Line Level	No matching category	No category assigned. These are less prevalent warnings depening on type of code
Line Level	Performance	Issues which result in less efficient code
Line Level	Potential Bug	Issues which may affect functionality in some circumstances. They are at the very least misleading to developers and so should be fixed as they pose a high risk that they either currently affect functionality or may mislead developers in such a way as to introduce bugs in the future
Line Level	Security	Items which are know security flaws, which may allow various exploits at varying severity levels
Line Level	Smell	Code structure that is abnormal e.g. large or overly complex method or class
Line Level	Stylistic	Code structure that is abnormal at the line level. e.g. brackets on same line as for statement, instead of on the next line

sema

Team Assessment Checklist

- Is individual developer activity monitored?
- Are low activity developers coached on improvement?
- Is type of developer activity monitored (create vs. change), and is coaching provided?
- Is skill of developers measured quantitatively as well as qualitatively?

- Among developers identified as highest contribution (top 10-20%) by Sema's analysis…
  - Do you agree or can you explain? (e.g. administrative changes)
  - Among developers with actual high knowledge, how many are still at COMPANY? If they are no longer there was there a knowledge transfer before they left? What risk mitigation methods do you have in place for protecting current subject matter expertise?

- Among developers identified as highest skill (top 10-20%) by Sema's analysis…
  - Do you agree or can you explain?
  - If you agree and the assessment is accurate, do they still work at COMPANY? Have they been recognized/ compensated?

- Among developers identified as lowest skill and lowest contribution (10-20%) by Sema's analysis…
  - Do you agree or can you explain (e.g. new developers)?
  - If you agree and the assessment is accurate, what is the status of those developers? Have they been coached and or evaluated? Can you provide stats, e.g. "XX of the YY low skill developers have been exited."
  - In particular is there a training program in place for new-to-COMPANY developers (low contribution, varied skill)

Contribution – Top 15 Contributors – All Time

Contribution

CONTRIBUTOR	CHG ANY % INIT	CHG OTHER % OF INIT	CHG OTHER COUNT	CHG OWN % OF INIT	CHG OWN COUNT	FILES CREATED COUNT	FIRST COMMIT COUNT	EXP PER REPO
Mike McGann	8	0	202	92	1,000	1,000	500	1,000
Paolo M. Masci	14	0	25	173	261	129	69	823
lkeelyme	18	3	12	225	278	114	57	608
Victor Woeltjen	48	20	625	349	634	265	131	255
Scisco	104	70	746	429	348	232	58	233
ZachTRice	12	6	947	84	300	394	165	212
Christopher Webster	4	0	35	45	132	15	135	267
Taylor Gunnoe	12	9	239	69	54	78	36	218
Patrick Quinn	13	0	31	157	221	202	65	195
Charles Hacskaylo	135	110	753	320	87	41	17	108
Brian Coltin	36	10	74	321	49	15	7	165
Marc Huffnagle	46	33	885	161	91	80	26	93
Benjamin King	74	37	621	460	214	105	44	81
asherp	13	0	1	163	22	13	6	180
Lauren Frederick	59	45	1,000	186	87	90	38	69

- The top 5 contributors all-time are Mike McGann, Paolo M. Masci, lkeelyme, Victor Woeltjen, and Scisco.
- Mike McGann has created more files, changed the most of their own files, has the most first commits, and scores highest in experience per repository. This indicates that Victor has significant knowledge of these repositories.
- Lauren Frederick has changed others' files more than any other developer.

sema

Skill – Architectural Impact – Top 15 Contributors – All Time

Skill - Architectural Impact

CONTRIBUTOR	EFFECTIVENESS	EXTENDIBILITY	FLEXIBILITY	FUNCTIONALITY	REUSABILITY	UNDERSTANDABILITY
Mike McGann	62	62	62	2	2	97
Paolo M. Masci	100	100	100	8	6	91
Ikeelyme	73	72	72	15	13	84
Victor Woeltjen	0	0	0	100	100	0
Scisco	62	62	62	2	2	97
ZachTRice	62	62	62	4	4	94
Christopher Webster	70	70	70	31	32	65
Taylor Gunnoe	62	62	62	2	2	97
Patrick Quinn	62	62	62	2	2	97
Charles Hacskaylo	62	62	62	2	2	97
Brian Coltin	62	62	61	2	2	97
Marc Huffnagle	62	62	62	2	2	97
Benjamin King	63	63	63	2	2	97
asherp	62	62	62	2	2	97
Lauren Frederick	62	62	62	2	2	97

- Paolo M. Masci has made the most positive impact on these repositories for 4 of 6 architectural impact metrics.
- Sema's default assumption is that all architectural outliers are high skill contributors, if this were to be used for evaluation purposes. Note: all metrics should be TUNED for NASA's specific case before using them in this manner.

sema

Skill – Line Level Warnings – Top 15 Contributors – All Time

Skill - Line Level Warnings

CONTRIBUTOR	ENVIRONMENT_SENSITIVE	MISLEADING	PERFORMANCE	POTENTIAL_BUG	SECURITY	SMELL	STYLISTIC	SUBSTANTIVE SUBTOTAL
Mike McGann	100	100	100	97	100	100	99	596
Paolo M. Masci	100	100	100	100	100	100	100	600
lkeelyme	100	96	98	27	80	0	82	400
Victor Woeltjen	99	100	0	98	93	98	99	488
Scisco	100	97	100	100	100	100	99	597
ZachTRice	100	99	100	100	100	100	97	599
Christopher Webster	99	100	96	99	93	99	100	587
Taylor Gunnoe	100	100	100	100	100	100	67	600
Patrick Quinn	100	96	100	100	100	100	84	596
Charles Hacskaylo	100	99	100	100	100	100	100	599
Brian Coltin	100	100	100	100	100	100	99	600
Marc Huffnagle	100	99	100	100	100	100	100	599
Benjamin King	100	98	100	95	100	100	96	593
asherp	100	100	100	100	0	100	100	500
Lauren Frederick	100	99	100	100	100	100	100	599

- "asherp" has committed the most Security warnings and "centos" the most Potential Bug warnings (not shown on this chart). Sema recommends reviewing these warnings, remediating them as needed, and coaching the team.

sema

Summary of Skill and Contribution – top 60

Summary of Skill and Contribution

CONTRIBUTOR	SKILL	CONTRIBUTION
Mike McGann	61	100
Paolo M. Masci	100	73
Ikeelyme	17	55
Victor Woeltjen	4	34
Scisco	61	31
ZachTRice	62	29
Christopher Webster	76	24
Taylor Gunnoe	53	21
Patrick Quinn	57	20
Charles Hacskaylo	62	18
Brian Coltin	62	17
Marc Huffnagle	62	16
Benjamin King	61	16
asherp	35	16
Lauren Frederick	62	15
Jonathan Kovarik	62	15
Jenny Liu	62	15
James Harris	63	14
Jason Gilman	63	14
Marc	62	14

Summary of Skill and Contribution

CONTRIBUTOR	SKILL	CONTRIBUTION
Pete Richards	62	14
Jason Kent	60	13
Mark Boyd	62	12
Josiah Sprague	35	10
Henry	62	9
minniewong	61	9
Dan Berrios	43	9
Chuck Daniels	62	8
Matt Savoie	62	8
Menno Van Diermen	61	8
Ed Plato	57	7
Andrew Henry	62	6
Katherine Kelly	62	6
Ryan Boller	63	5
Pat Cappelaere	62	5
Darren De Zeeuw	62	5
Aimee Barciauskas	62	5
Joel McKinnon	60	5
Seth Vincent	59	5
Cappelaere	63	4

Summary of Skill and Contribution

CONTRIBUTOR	SKILL	CONTRIBUTION
shale	63	4
Joshi	62	4
Menno	62	4
Jacob Campbell	62	4
Edward Plato	59	4
Scott McMichael	58	4
P. Michael Furlong	53	4
Daniel Pacak	19	4
Chris Garry	63	3
James Norton	63	3
Ulysses Dotson	62	3
David Tsay	62	3
Shivam Dave	62	3
mhuffnagle	62	3
Shefali Joshi	62	3
Deep Tailor	62	3
alpearson	61	3
Michael Furlong	60	3
Nick Weingartner	63	2
David Hudson	63	2

sema