Client
US Government

Date
Fall 2020

Sema Capabilities Demonstration on Open Source DOD/ IC/ USG Code

sema

Overview and Summary

- Sema is a software company based in Baltimore, MD- AFWERX Phase I recipient, backed by Gates/ Bezos/ Zuckerberg, serving the Fortune 500 and the world's top investors
- We built software that evaluates the quality of code and the quality of developers through 1100 tunable metrics with < 1 hour of setup
- We analyzed 6 open source applications (5,179,231 lines) from USAF / DOD and analyzed them through Sema Code Quality Platform
- Our functionality can serve National Security via:

    1: Evaluate existing software systems to maintain or rebuild

    2: Manage code quality for Mission goals while code is written

    3: Support and coach developers to improve their craft

    4: Assess third party development shops to support procurement

    5: Forensics

- Appendix

What we did

- Found 23 open source applications from 13 Agencies created by the Federal government (Sema works on open source and private code), including 6 from DOD/ IC
- Analyzed these applications through Sema Code Quality Platform
- Picked examples across these applications to illustrate use cases we have heard from 250+ Government stakeholders

We looked at 23 applications from 13 agencies

Agency	Count of Applications
CDC	1
CFPB	1
Department of Agriculture	1
Department of Commerce	1
Department of Defense	3
Department of Homeland Security	1
GSA	2
GSA (18F)	3
NASA	2
NGA	5
NSA	1
Treasury	1
US Customs	1
Grand Total	**23**

DOD / IC Scope Detail: 6 applications

Replicator This project is used to sync all container images from the DoD Centralized Artifacts Repository to a container registry.

VAT (vat-frontend, vat-backend) https://repo1.dsop.io/dsop/iron-bank/vat/vat-frontend

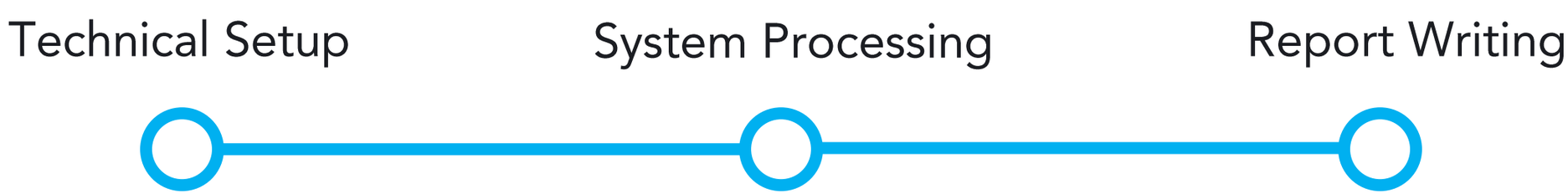
Aws-infrastructure: The Platform One project is designed to provide a UPI/IPI focused automated deployment of a Kubernetes distribution that can be rolled out to any DoD organization.

Dccscr: This repository is used to store Dockerfiles and associated checksums and various documentation. The source code repository is centrally hosted so hardeners can store their code and leverage a CI/CD pipeline.

Opensphere-desktop: application used to visualize temporal/geospatial data. The application represents data using a three-dimensional model of the earth, and has the ability to handle large volumes of features and tiles

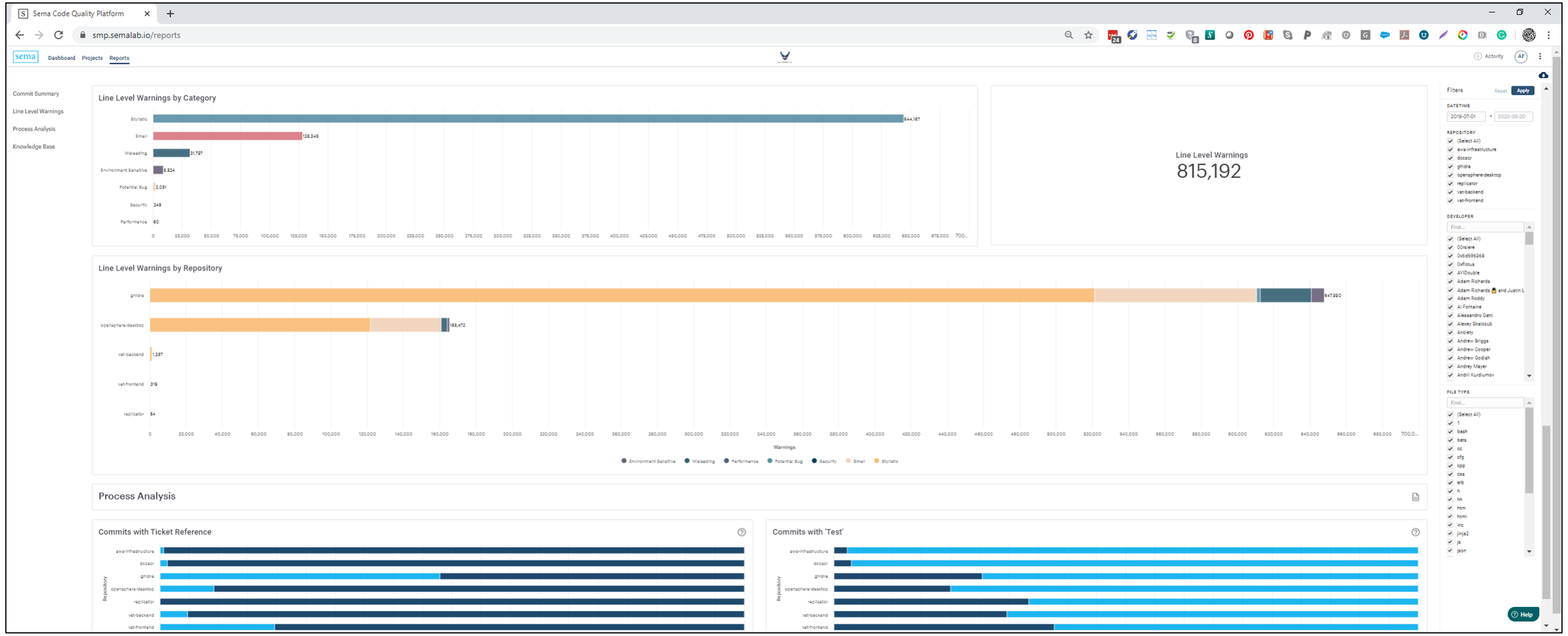
Ghidra: software reverse engineering (SRE) framework developed by NSA's Research Directorate for NSA's cybersecurity mission. It helps analyze malicious code and malware like viruses

Timeline for this Demonstration



sema

Sample Output from Sema Code Quality Platform



sema

Executive Summary- Code

Strength | Medium Risk | Low Risk | High Risk

Element of Code quality	Discussion
Size	• Analyzed 6 applications with 7 total repos, 5,179,231 lines, 23,044 files, 379,915,404 bytes, and 207 all time developers
Language composition	• Strength - Iron Bank repositories are written in modern and widely accessible languages (typescript, python, javascript) • Discuss - ghidra contains code written in 19 different languages - is this in line with expectations and support capabilities?
Core Technical Debt	• Discuss/ Medium risk- Cost per Line of Code for the 6 repos in total is at $4.74 - >$5 is high risk. 94% of tech debt due to apparent lack of unit testing- deeper understanding of testing methods required. Assuming unit test coverage can be low given these are open source repositories- With a unit test standard of 10%, the technical debt would be $.24/ line of code, which is excellent • Strength – excellent (low) technical debt in Iron Bank repositories • Low Risk - duplication and overly complex code in opensphere-desktop and ghidra
Line Level Warnings	• Medium Risk - Investigate and as needed remediate 60 Performance, 248 Security, and 2,051 Potential Bugs warnings • Low Risk - Investigate and as needed remediate 128,345 Smells, 8,524 Environment Sensitive, and 31,797 Misleading warnings
Third party code	• Strength - 0.1% of code is from third party libraries - react is the most prevalent vendor
Package dependencies	• Strength - ghidra and opensphere-desktop follow modern architectural practices for package dependencies

Executive Summary – Code, Process and Team

Strength | Medium Risk | Low Risk | High Risk

Element of Code quality	Discussion
Distributed repositories and smaller packages	• Strength- 2 of the 2 repositories with supported languages have understandable package dependency structures. • Low Risk- 2 of 2 repositories analyzed have packages with >1000 dependencies.
Commit Analysis	• Discuss - 3 large file change events which appear to be imports or large updates to code • Discuss - opensphere-desktop seems to have very little activity in terms of both file changes and commits since February 2019. • Discuss - pair programming
Commit Management	• Strength- 4 of the 7 repositories have more than 10 files per commit, due to one-off administrative changes discussed above. Excluding these one-off events, files per commit are in the appropriate range. • Low Risk- review commit practices for developers with 10-25 files per commit and see if there is a development rationale. If not, coach towards fewer files per commit
Ticket Referencing	• Low risk - All products have a ticket reference rate below 60%. Is this in line with each organization's processes?
Test Referencing	• Investigate - 5 of 7 products have fewer than 40% of their commits with test files in the last year. See discussion above about testing
Developer Contribution and Skill	• Developer contribution is provided for each developer, and Developer skill where Sema has language coverage • Once it is tuned for specific needs, it can be used to help achieve desired organizational goals • These metrics can also be used for the Procurement and Core Forensics use cases

- **Code**
- Process
- Team
- Appendix

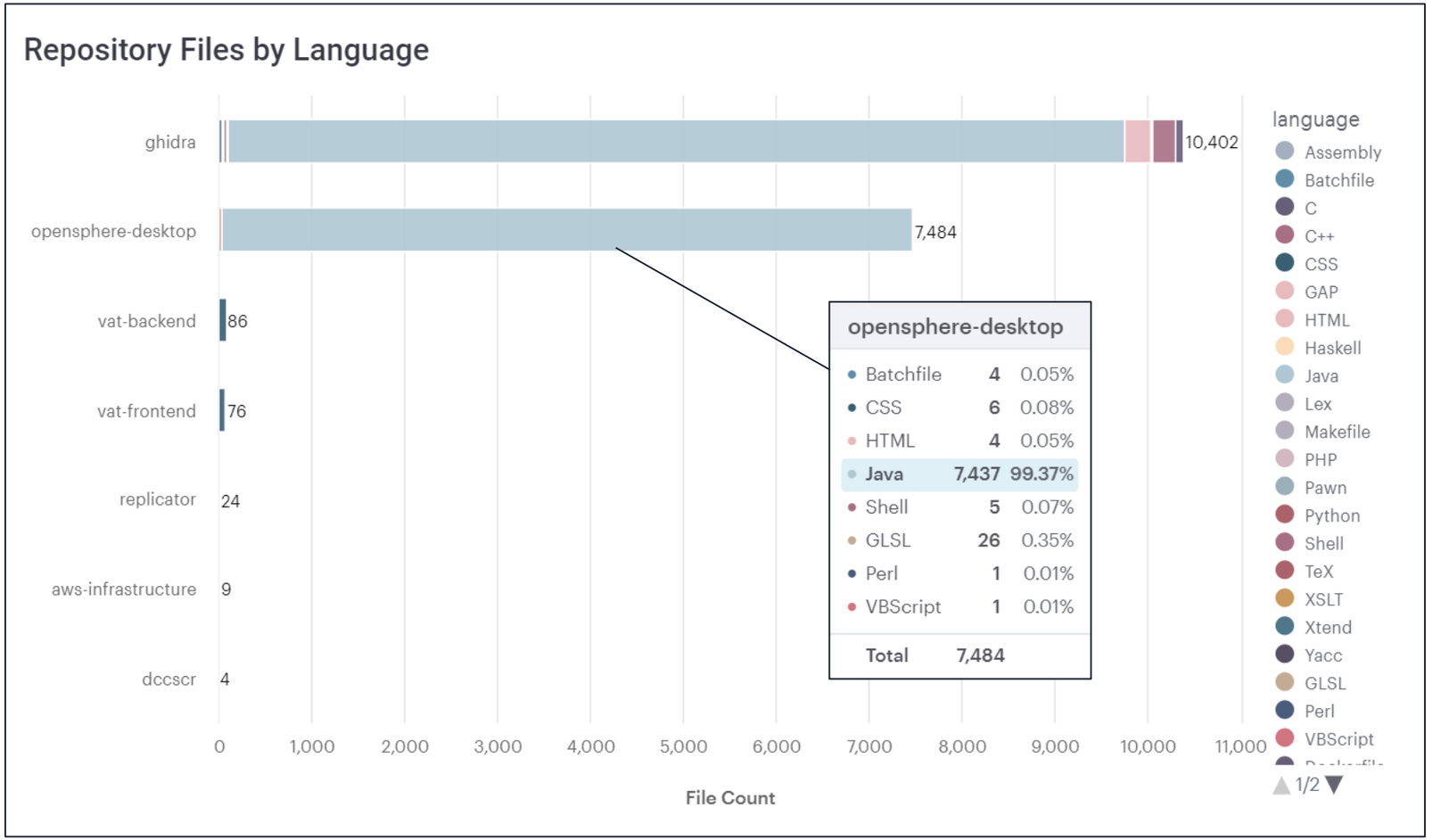
Basic Facts

Item	Amount
Total size	5,179,231 lines, 23,044 files, and 379,915,404 bytes
Number of Products - Analyzed	6
Number of Repositories - Analyzed	7
First Commit Analyzed	January 25, 2018 in opensphere-desktop
Last Commit Analyzed	May 20, 2020 in ghidra
Number of contributors	
All time	207
Current (Last 90 Days)	62

Language Composition

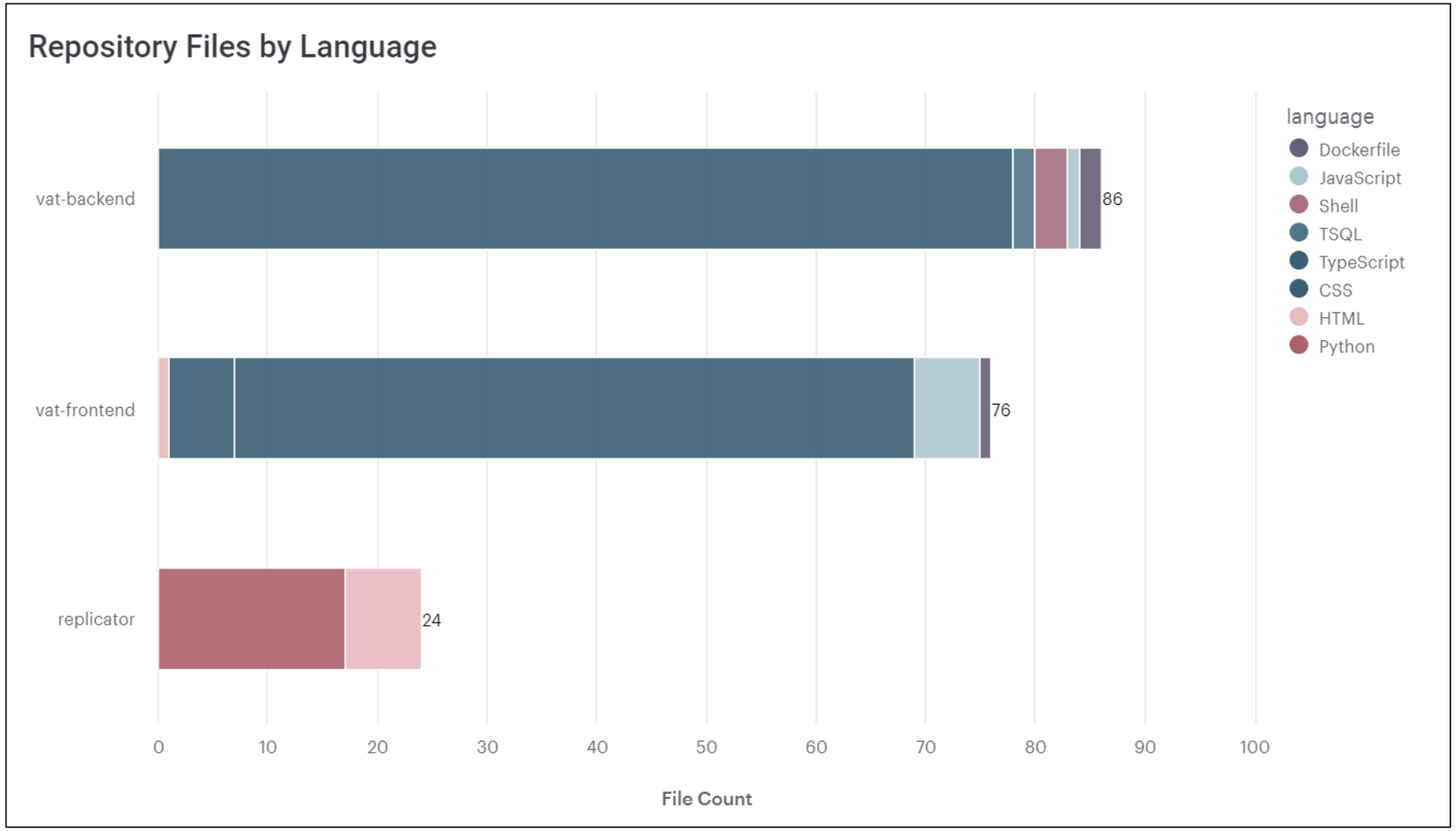
- vat-frontend and vat-backend are written in typescript
- replicator is primarily python
- RepositoryA and RepositoryB are java
- Aws-infrastructure and dccscr both consist primarily of Dockerfiles with shell scripts
- **Strength** – CompanyA repositories are written in modern and widely accessible languages (typescript, python, javascript)
- **Discuss** - RepositoryA contains code written in 19 different languages - is this in line with expectations and support capabilities? How do different USAF / DOD programs think about managing language diversity?

Language Composition per Repository



sema

Detail - Language Composition per 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 Summary, untuned…

Technical Debt for Organization

ORGANIZATION	COST PER LOC	TECH DEBT TOTAL	COMPLEXITY DAYS	DUPLICATION DAYS	UNIT TEST DAYS	LINE LEVEL WARNI...	TOTAL DAYS TO FIX
DepartmentOfDefense	$4.74	$7,682,945	104	359	14,575	328	15,366

REPOSITORY	COST PER LOC	TECH DEBT TOTAL	DUPLICATION DAYS	COMPLEXITY DAYS	UNIT TEST DAYS	LINE LEVEL WARNING DAYS	TOTAL DAYS TO FIX
opensphere-desktop	$5.24	$2,988,404	178	25	5,450	323.55	5,977
ghidra	$4.50	$4,691,587	181	79	9,123	0	9,383
vat-backend	$0.92	$2,535	0	0	2	3.375	5
replicator	$0.66	$263	0	0	0	0.3375	1
vat-frontend	$0.04	$156	0	0	0	0.3125	0

- Medium risk - Cost per Line of Code for applications as a whole is $4.76
- **Strength** – excellent (low) technical debt in Iron Bank repositories- small repos but on a per line basis the core technical debt is low
- **Low Risk** - duplication and overly complex code in opensphere-desktop and ghidra
- 94% of the core technical debt in RepositoryC is caused by an apparent lack of unit testing.
    - **Discuss** - Might testing be carried out a different way than unit tests, or in a way that was not initially identified through Sema solution?
    - **Hypothesis** - these open source repositories are self-correcting and USAF/ DOD's desired unit test levels should be 10-20% for these applications

Core Technical Debt Summary, with unit testing standard at 10%

Technical Debt for Organization

ORGANIZATION	COST PER LOC	TECH DEBT TOTAL	COMPLEXITY DAYS	DUPLICATION DAYS	UNIT TEST DAYS	LINE LEVEL WARNI...	TOTAL DAYS TO FIX
DepartmentOfDefense	$0.24	$395,683	104	359	0	328	791

REPOSITORY	COST PER LOC	TECH DEBT TOTAL	DUPLICATION DAYS	COMPLEXITY DAYS	UNIT TEST DAYS	LINE LEVEL WARNIN...	TOTAL DAYS TO FIX
vat-backend	$0.65	$1,787	0	0	0	3.4	4
opensphere-desktop	$0.46	$263,400	178	25	0	323.6	527
replicator	$0.43	$169	0	0	0	0.3	0
ghidra	$0.12	$130,172	181	79	0	0.0	260
vat-frontend	$0.04	$156	0	0	0	0.3	0

- With a unit test standard of 10%, the technical debt would be $.24/ line of code, which is excellent

At least 10 applications are obvious candidates to maintain

Technical Debt by Repository

REPOSITORY	DUPLICATE_BLOCKS_DAYS	LINE_LEVEL_WARNINGS_DAYS	TEST_LOC_RATIO_DAYS	COMPLEXITY_DAYS	TOTAL DAYS TO FIX	TECH DEBT TOTAL	COST PER LOC
GTAS	8,084	96	3,145	5	11,329	$5,664,533	$60.32
ADSM	0	33	355	2	389	$194,472	$11.79
Dshell	2	2	147	0	152	$75,821	$11.74
data.gov	0	2	895	2	899	$449,258	$11.20
lemongraph	0	7	157	1	165	$82,544	$11.00
pshtt	0	1	19	0	20	$10,210	$8.15
Epi-Info-Community...	1,485	0	4,793	25	6,303	$3,151,613	$5.87
opensphere-desktop	0	898	5,446	25	6,370	$3,184,790	$5.59
calc	7	10	218	1	236	$117,842	$5.49
ghidra	0	1,401	9,775	80	11,255	$5,627,315	$5.40
cfgov-refresh	66	20	322	3	410	$205,241	$4.25
mrgeo	28	67	461	4	560	$280,027	$4.24
janusgraph	4	132	316	3	455	$227,668	$4.06
anet	2	74	289	2	366	$183,141	$3.96
dol-whd-14c	7	6	254	1	268	$134,117	$3.93
send	0	0	59	0	59	$29,596	$3.70
credhub	0	77	0	0	78	$38,859	$3.27
elasticgeo	0	19	0	0	20	$9,963	$2.07
data-act-broker-bac...	9	93	5	2	108	$53,910	$1.95
openmct	0	0	131	1	132	$65,933	$1.81

- Code with <$5 of tech debt per line Sema benchmarking: <$2 is good, $2-5 is investigate, >$5 is at risk
- There are XX applications that have <$5 of technical debt per line, including openmct and data-act-broker-backend which have particularly low technical debt

sema

RepositoryB could be a candidate for rebuilding

Technical Debt by Repository

REPOSITORY	DUPLICATE_BLOCKS_DAYS	LINE_LEVEL_WARNINGS_DAYS	TEST_LOC_RATIO_DAYS	COMPLEXITY_DAYS	TOTAL DAYS TO FIX	TECH DEBT TOTAL	COST PER LOC
GTAS	8,084	96	3,145	5	11,329	$5,664,533	$60.32
ADSM	0	33	355	2	389	$194,472	$11.79
Dshell	2	2	147	0	152	$75,821	$11.74
data.gov	0	2	895	2	899	$449,258	$11.20
lemongraph	0	7	157	1	165	$82,544	$11.00
pshtt	0	1	19	0	20	$10,210	$8.15
Epi-Info-Community...	1,485	0	4,793	25	6,303	$3,151,613	$5.87
opensphere-desktop	0	898	5,446	25	6,370	$3,184,790	$5.59
calc	7	10	218	1	236	$117,842	$5.49
ghidra	0	1,401	9,775	80	11,255	$5,627,315	$5.40
cfgov-refresh	66	20	322	3	410	$205,241	$4.25
mrgeo	28	67	461	4	560	$280,027	$4.24
janusgraph	4	132	316	3	455	$227,668	$4.06
anet	2	74	289	2	366	$183,141	$3.96
dol-whd-14c	7	6	254	1	268	$134,117	$3.93
send	0	0	59	0	59	$29,596	$3.70
credhub	0	77	0	0	78	$38,859	$3.27
elasticgeo	0	19	0	0	20	$9,963	$2.07
data-act-broker-bac...	9	93	5	2	108	$53,910	$1.95
openmct	0	0	131	1	132	$65,933	$1.81

- Sema benchmarking: <$2 is good, $2-5 is investigate, >$5 is at risk
- At $60 per line of code, GTAS is at risk
- It would take $5.7M to clean up the technical debt of GTAS, in particular to remove duplicate code and add testing

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

815,192 instances of line-level warnings were identified.

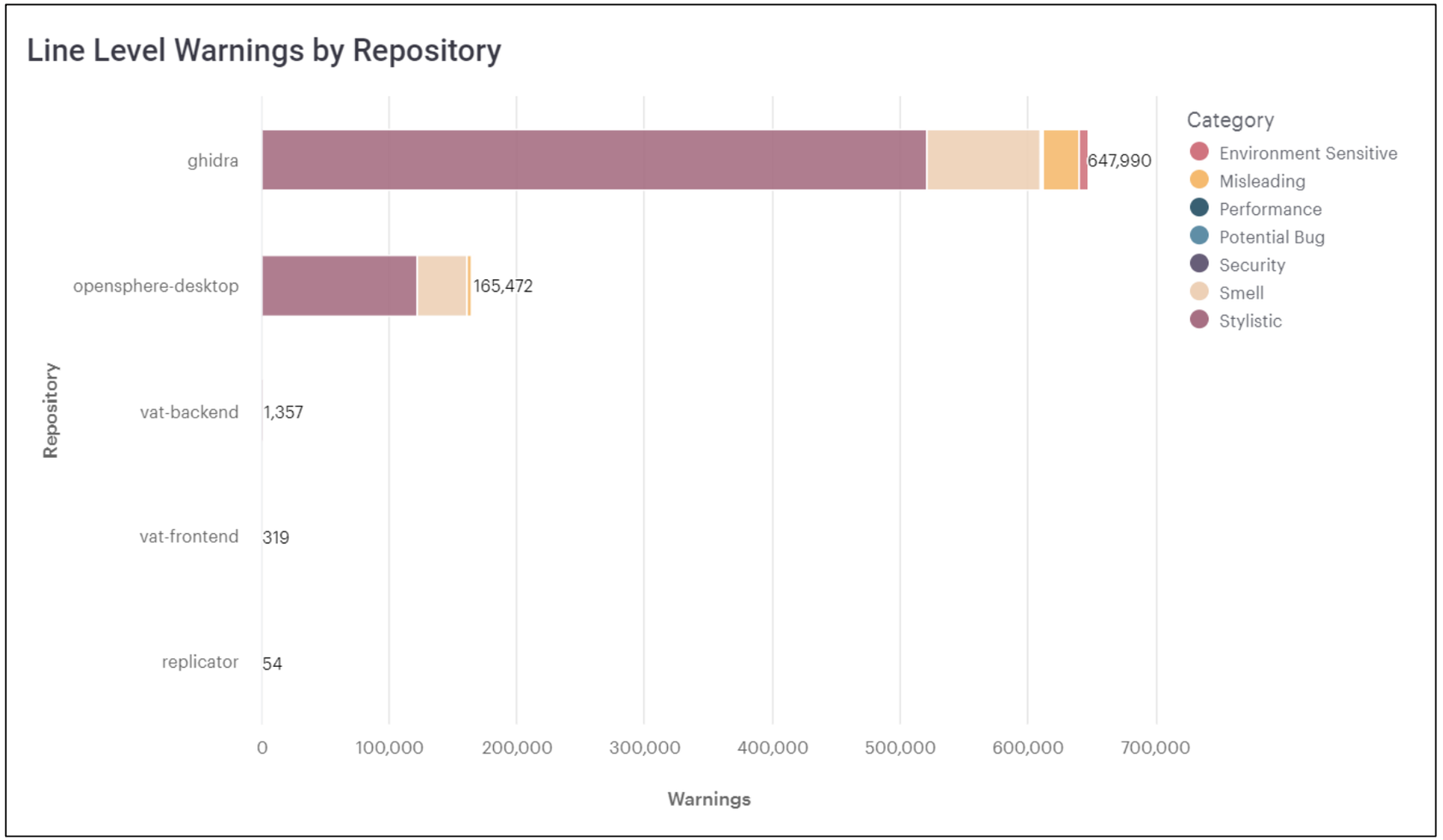
Stylistic warnings are 79% of the line-level warnings and are not considered technical debt.
Smells is the next highest at 15.9% of the line-level warnings.

Medium Risk - Investigate and as needed remediate 60 Performance, 248 Security, and 2,051 Potential Bugs warnings

Low Risk - Investigate and as needed remediate 128,345 Smells, 8,524 Environment Sensitive, and 31,797 Misleading warnings

Not recommended to investigate Stylistic warnings.

Line Level - All Repositories



- RepositoyB has 79% of the line level warnings and should be investigated

sema

These GTAS files are the ones to fix first

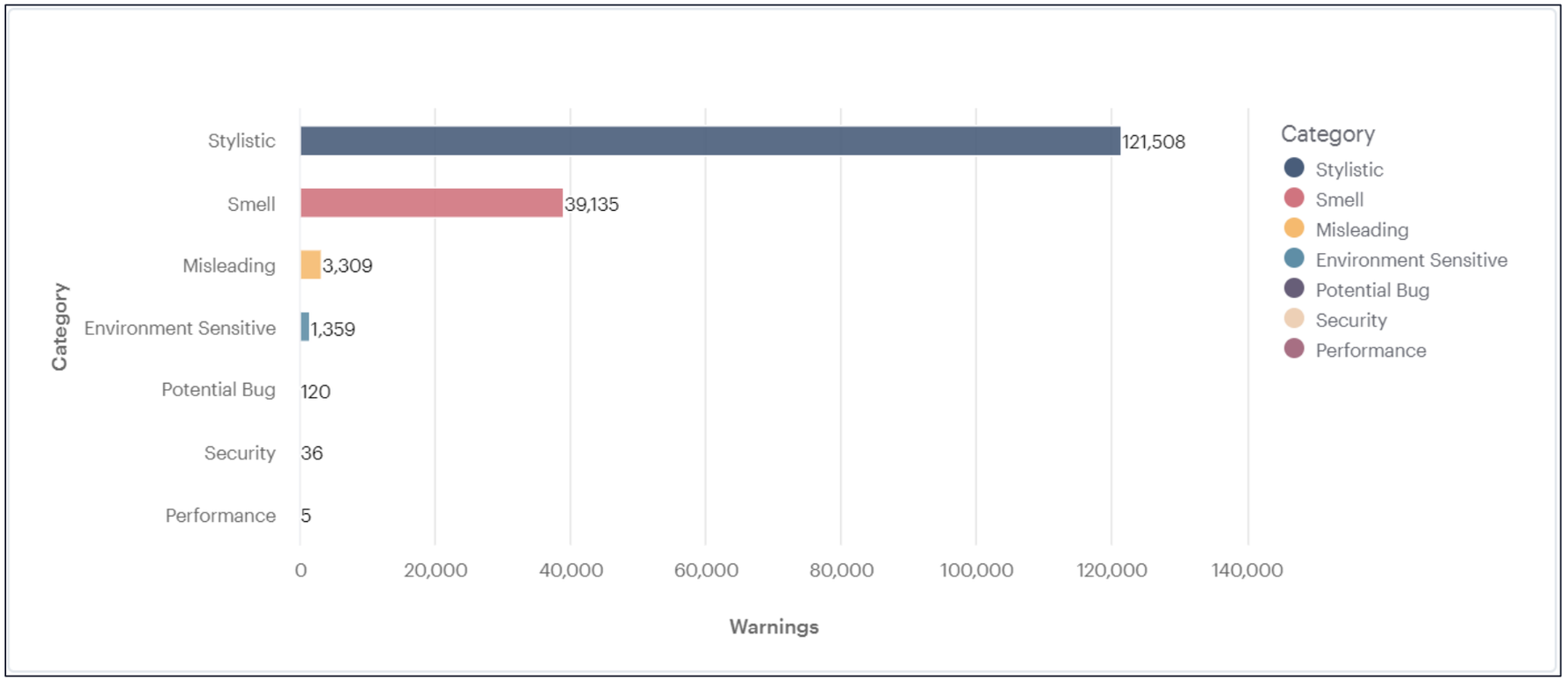
Tech Debt Indicators

	File >		Tech Debt Indicators				
Status	Path	Lines of Code	Complexity	Duplication	Unit Test Lines	Warnings Count	Tech Debt ($)
○	gtas-parent/gtas-webapp/src/main/webapp/dist/js/bower.components.min.js	43	16740	43022	0	0	2016703
○	gtas-parent/gtas-webapp/src/main/webapp/dist/js/bower.components.js	19956	16054.4	42992	0	0	2015297
○	gtas-parent/gtas-commons/src/main/java/gov/gtas/services/EventReportPdfTemplateService.java	663	64.8	9	0	515	3688
○	gtas-parent/gtas-rulesvc/src/test/java/gov/gtas/svc/UdrServiceIT.java	397	18.7	8	397	524	3650
●	gtas-parent/gtas-parsers/src/main/java/gov/gtas/parsers/pnrgov/PnrGovParser.java	988	303.6	0	0	574	3634
○	gtas-parent/gtas-commons/src/main/java/gov/gtas/querybuilder/JPQLGenerator.java	430	126	4	0	370	2547
○	gtas-parent/gtas-loader/src/main/java/gov/gtas/services/PnrMessageService.java	470	90	1	0	376	2444
○	gtas-parent/gtas-commons/src/main/java/gov/gtas/repository/DataManagementRepositoryImpl....	494	46.8	0	0	381	2381
○	gtas-parent/gtas-loader/src/main/java/gov/gtas/services/LoaderUtils.java	436	115.2	1	0	341	2225

- These are the top ten files with highest technical debt
- Teams can use this as a “punch list” to know what to fix first

sema

Detail example: RepositoryB has over XX,XXX substantive warnings



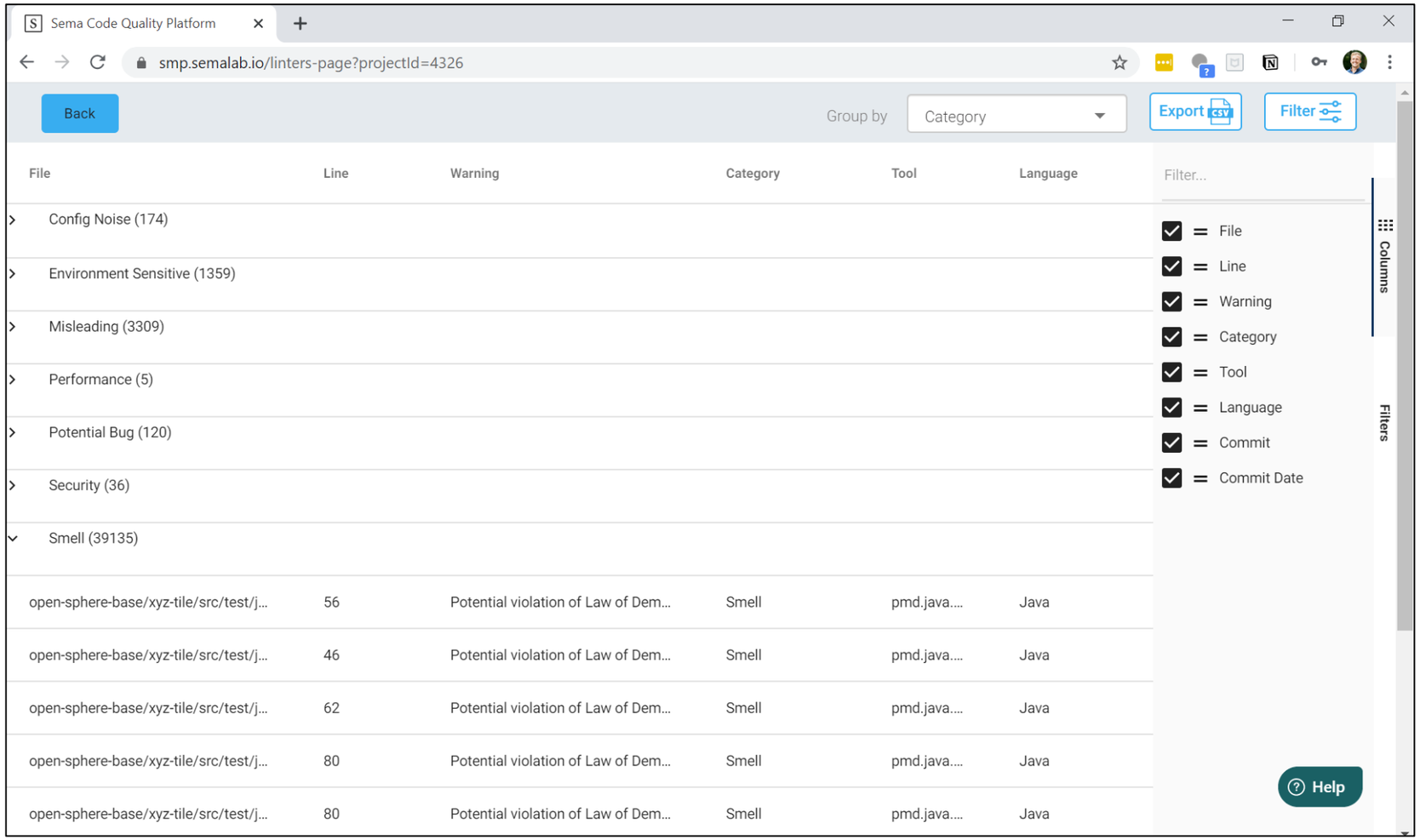
Line Level Warnings

165K

- Stylistic warnings are considered “house style,” not technical debt
- Excluding Stylistic, Code “Smells” are the next largest category of line warnings

sema

Detail: RepositoryB Smells File Location



sema

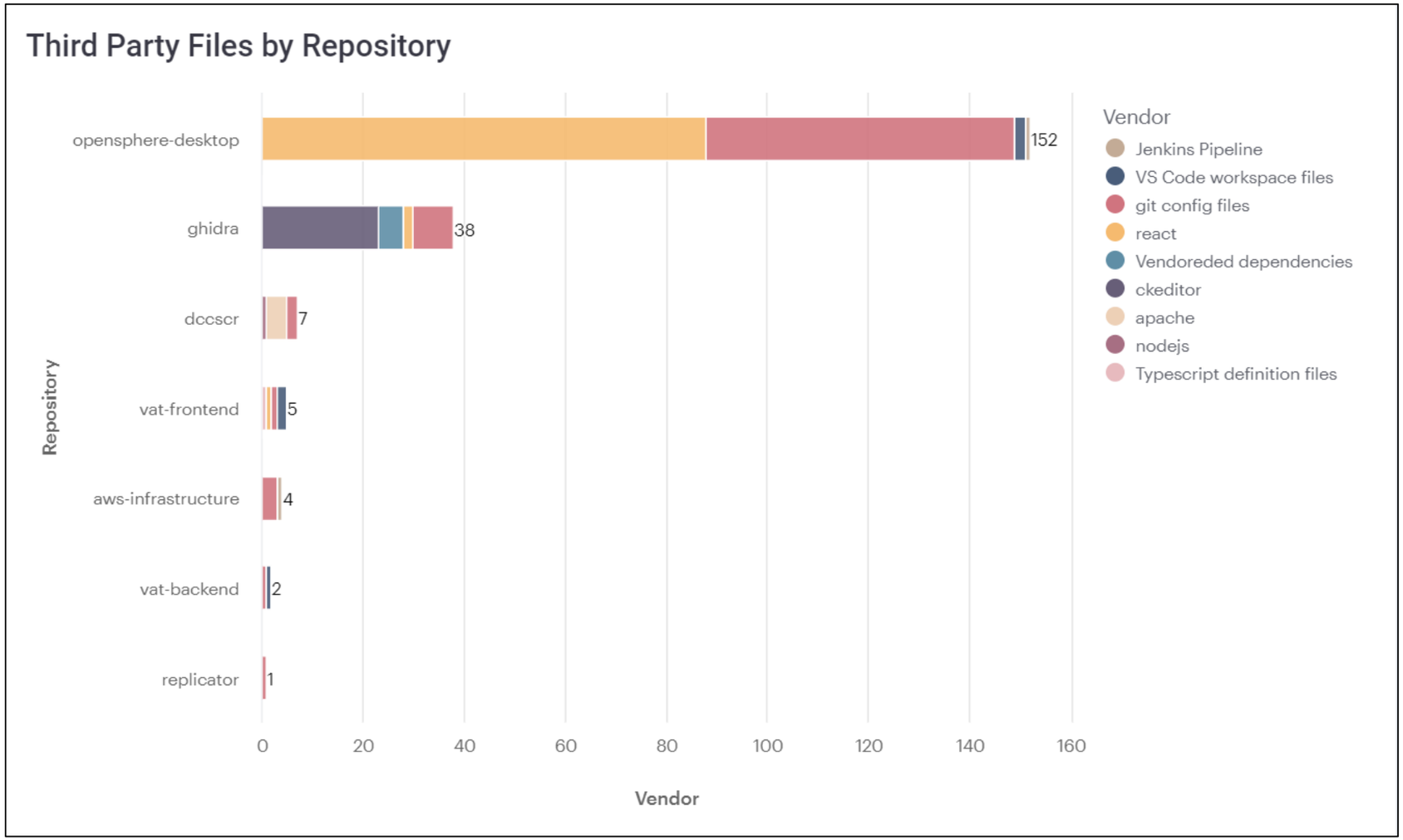
Third Party Libraries

Third party libraries are appropriate for efficient programming, modern code design. Sema recommends however that dependency management tools be put in place to minimize risk and ease maintenance of code, rather than including the code itself

Strength - 0.1% of code is from third party libraries. React is the most prevalent vendor

Discuss - are there substantive reasons to have included some or all of this third party code? If not, recommend removing and using a dependency management tool instead. A substantive reason for current practice, or use of manager, which would make this a “Strength.”

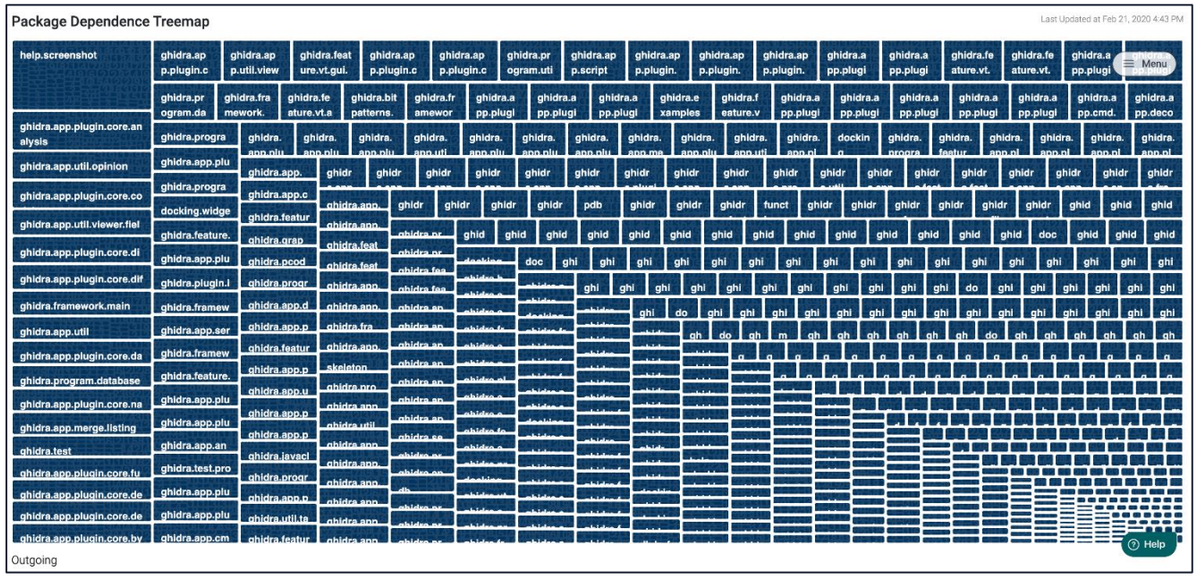
Third Party Libraries



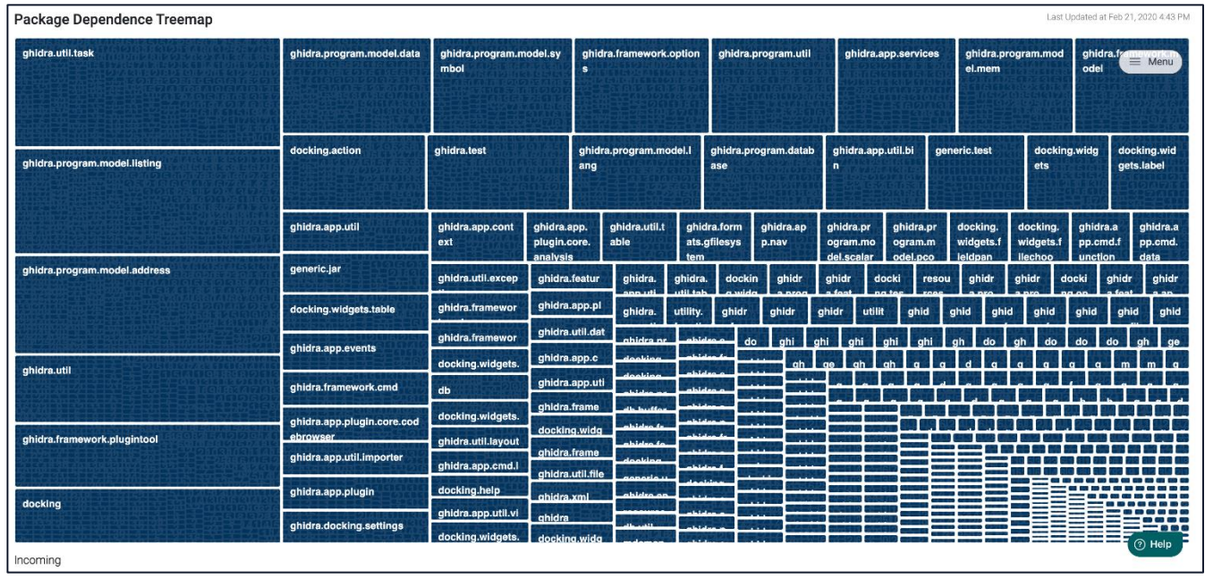
sema

Package Dependencies - RepositoryA

Outgoing



Incoming



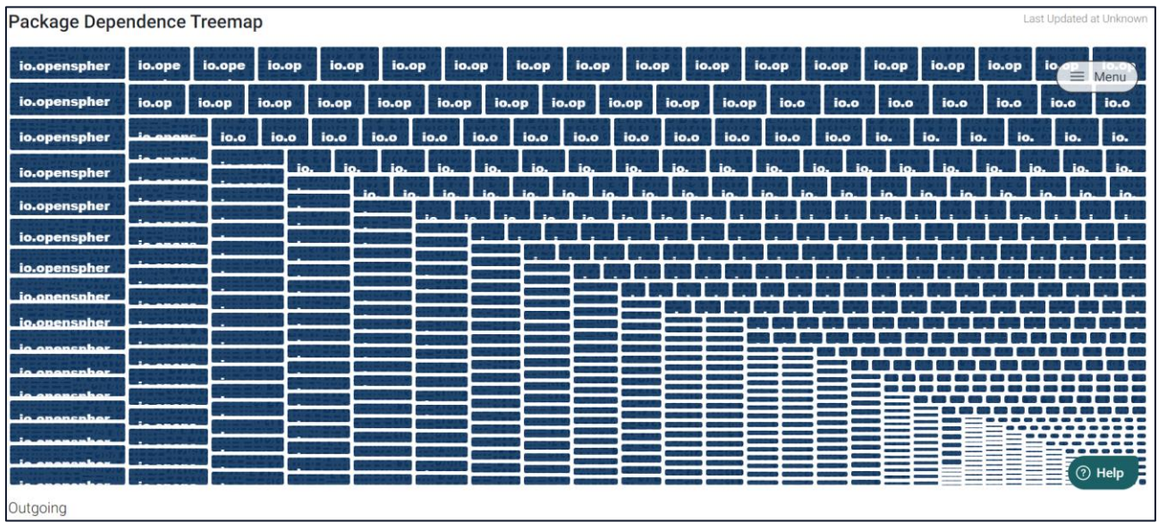
- Sema generally recommends that an application should have few classes with high outgoing dependencies, several classes with high incoming.

- **Strength** - More concentrated incoming dependencies suggest specialized classes performing functions in many parts of the system- a good practice

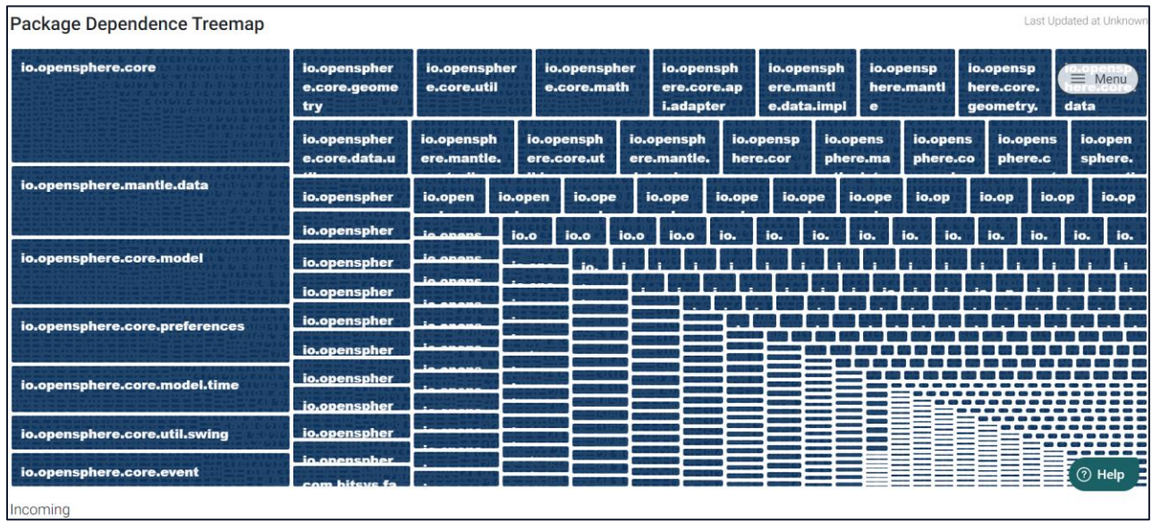
sema

Package Dependencies - opensphere-desktop

Outgoing



Incoming



- Sema generally recommends that an application should have few classes with high outgoing dependencies, several classes with high incoming.
- **Strength** - More concentrated incoming dependencies suggest specialized classes performing functions in many parts of the system- a good practice

sema

Distributed repositories and smaller packages

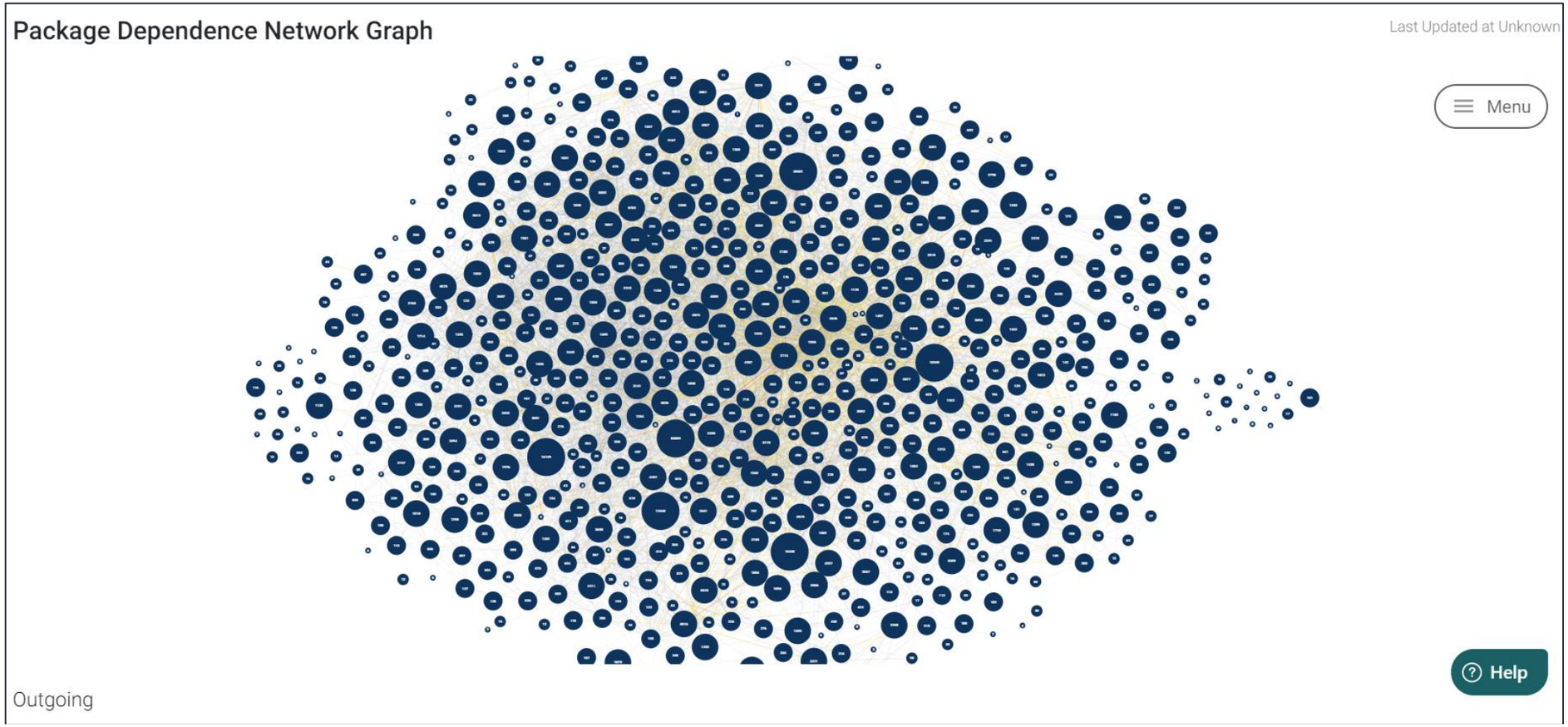
Sema generally recommends that both repositories and individual packages not get too large, for ease of maintenance, adding additional features, and team management.

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.

Low risk - ghidra and opensphere-desktop are candidates to be broken up into multiple repositories

Low risk - ghidra has packages with over 10,000 dependencies that are candidates to be broken up as well. For example app.merge.listing has 58,009 outgoing or self-referential dependencies

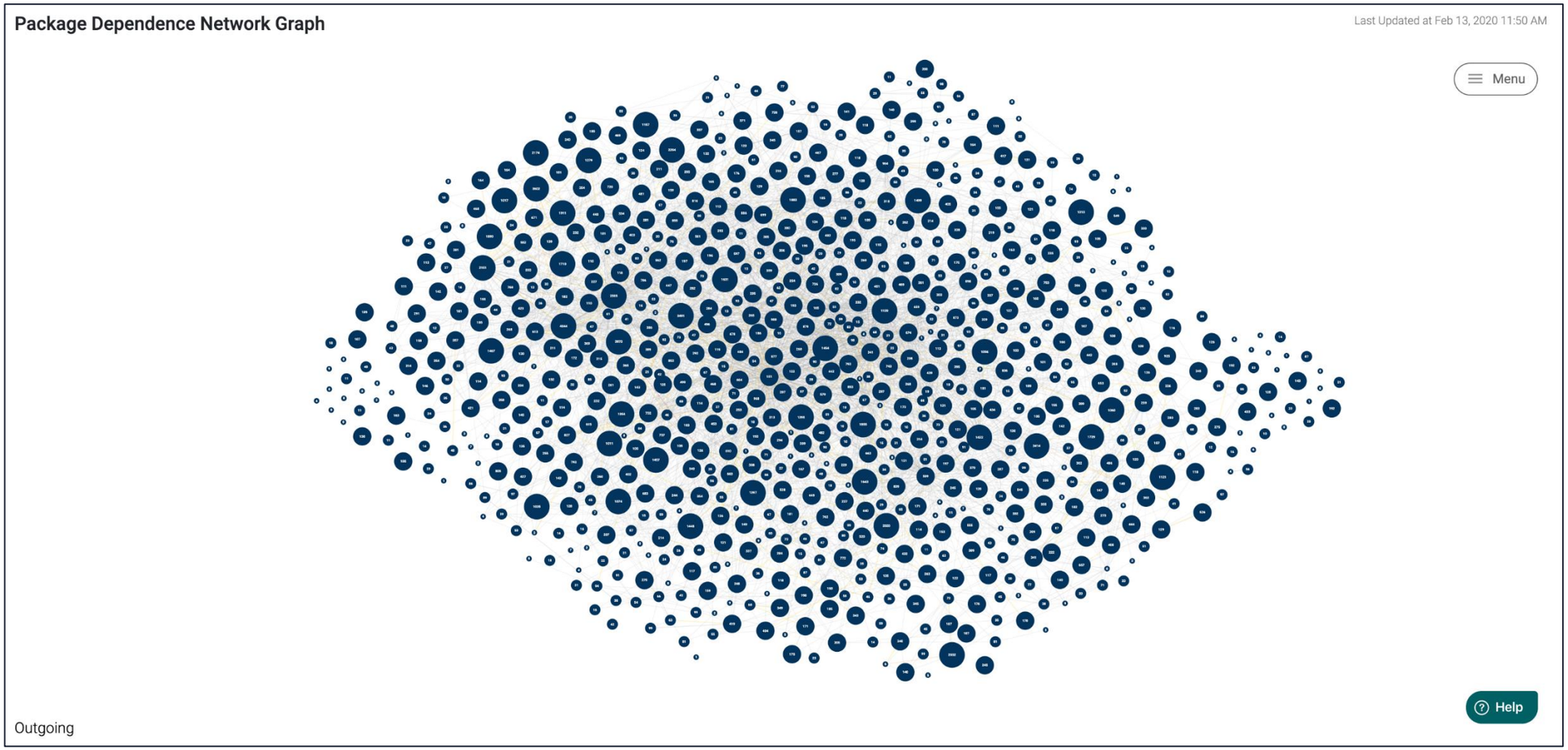
RepositoryA is a candidate to split into multiple repositories



- A large application in one repository can be hard to maintain or add functionality
- This code is well structured as a unit, but is a candidate for breaking into microservices

sema

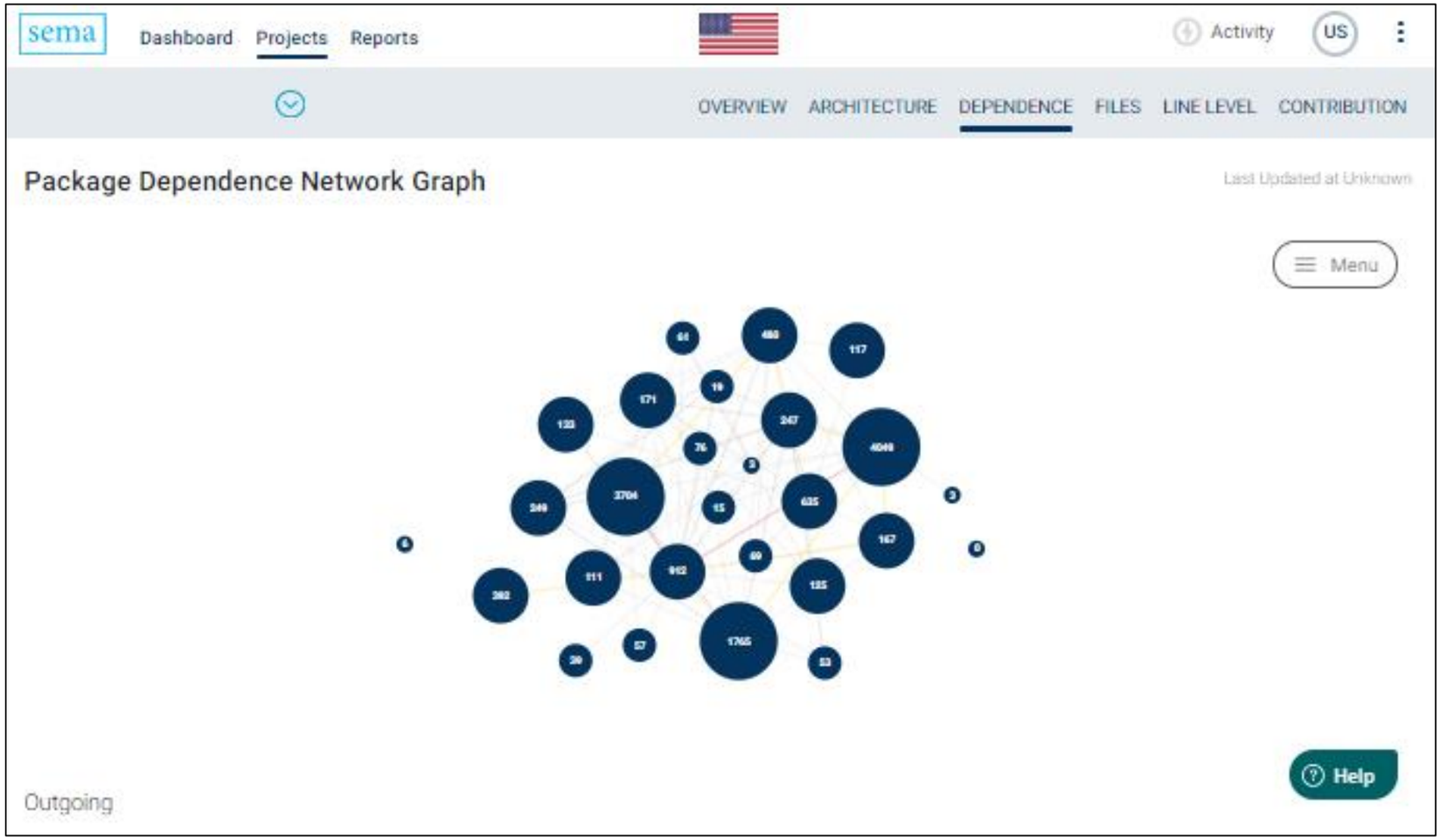
RepositoryA is also a candidate to split into multiple repositories



- A large application in one repository can be hard to maintain or add functionality
- This code is well structured as a unit, but is a candidate for breaking into microservices

sema

By contrast, here is another DOD application that is not a candidate for breaking up into multiple repositories- ANET

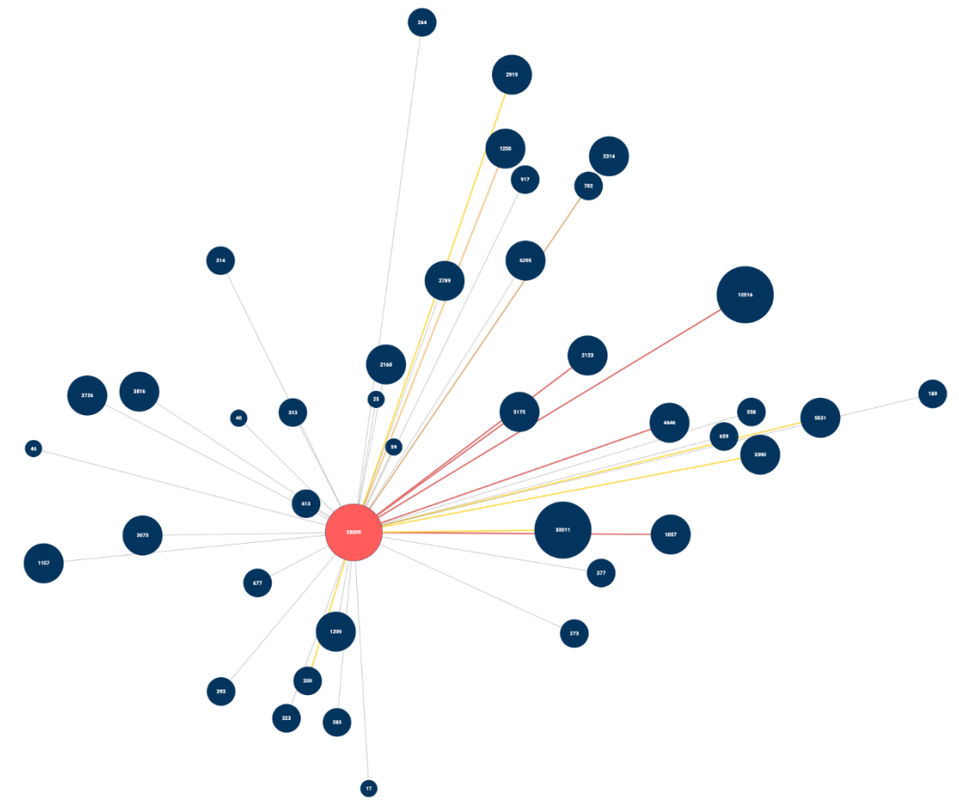


sema

Smaller packages: here is a large ghidra package to be investigated

Package Dependence Network Graph

ghidra.app.merge.listing	
outgoing → 58009	
ghidra.app.merge.listing	22596
ghidra.program.database	10698
ghidra.program.model.listing	9278
ghidra.program.model.symbol	8298
ghidra.program.model.address	3327
ghidra.program.model.data	1069
ghidra.util.task	970
ghidra.program.model.util	508
ghidra.program.util	290
ghidra.program.model.lang	247
ghidra.app.merge.tool	166



Outgoing

- Low Risk- 2 of 2 repositories analyzed have packages with >1000 dependencies.
- For example, this ghidra class should be reviewed to see if it should be broken up into smaller more specialized classes (>1000 dependencies and especially >10,000)

sema

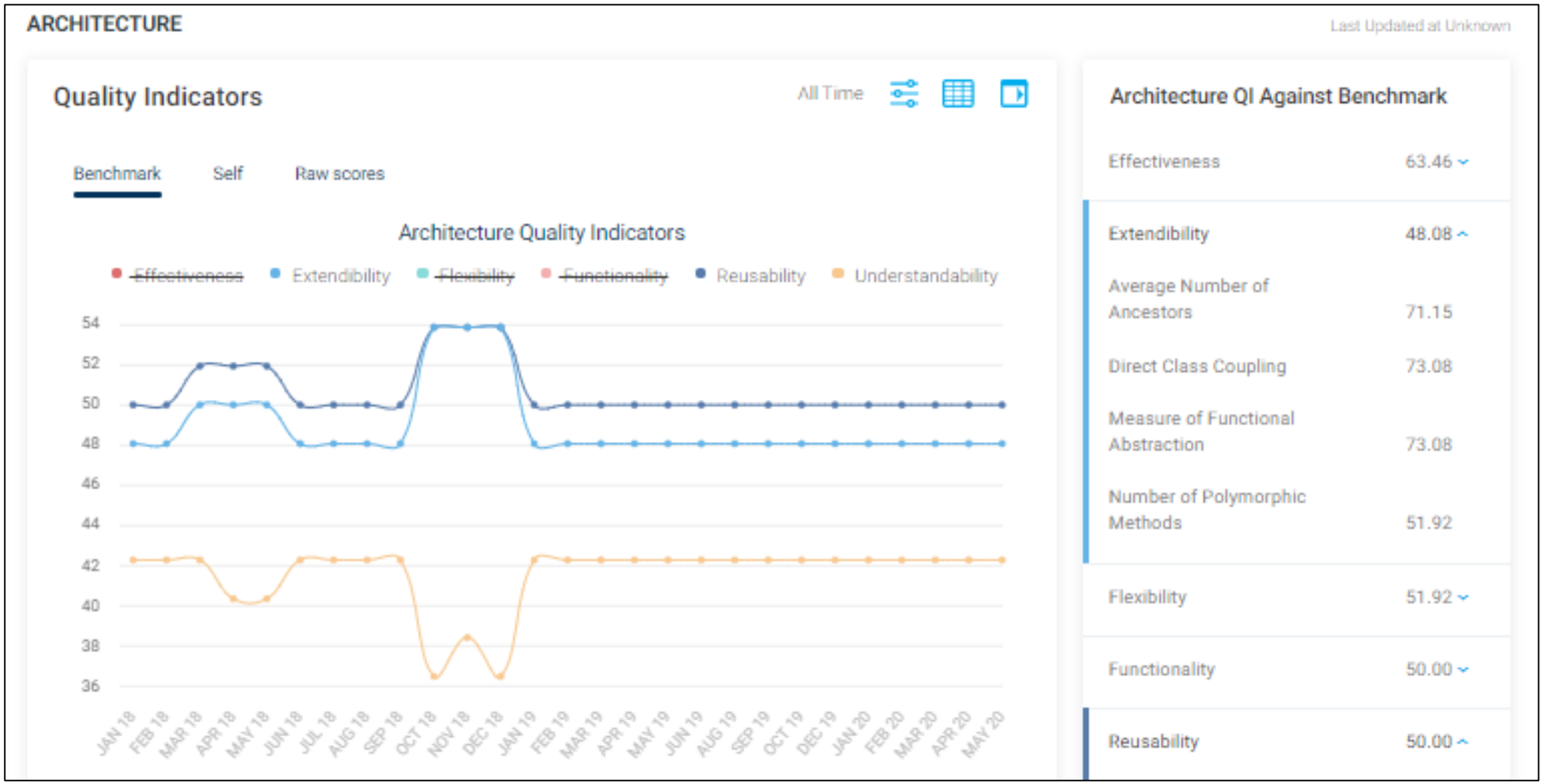
Architectural Quality Indicators

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.

Repository A became briefly more extendible in 2018…



- Extendibility:
  - Measurement of design's allowance to incorporate new functional requirements.
  - Calculation: 0.5 * Abstraction - 0.5 * Coupling + 0.5 * Inheritance + 0.5 * Polymorphism
- **Discuss** – what were architectural assumptions underlying this temporary change?

sema

… driven by an increase in polymorphic methods (NOP)

sema

- Code
- **Process**
- Team
- Appendix

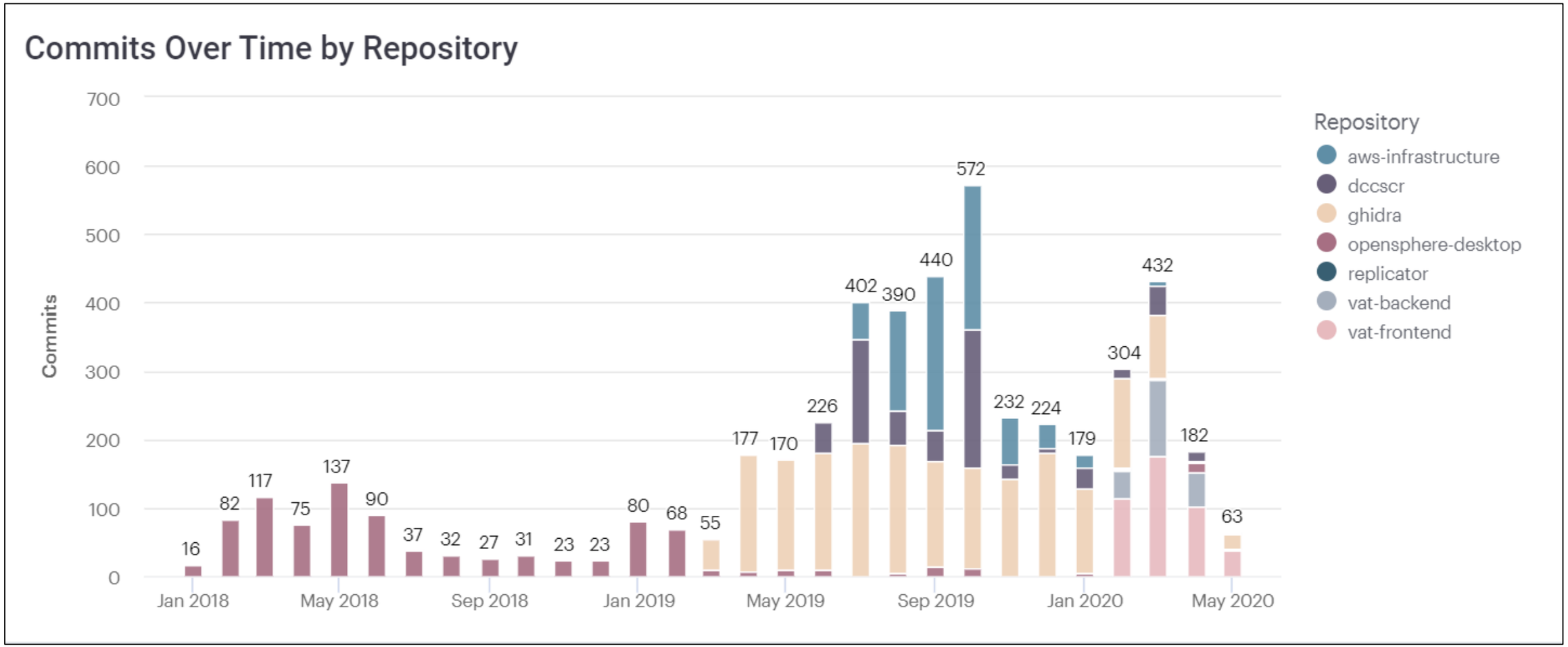
Commit Analysis

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

Discuss - opensphere-desktop seems to have very little activity in terms of both file changes and commits after February 2019

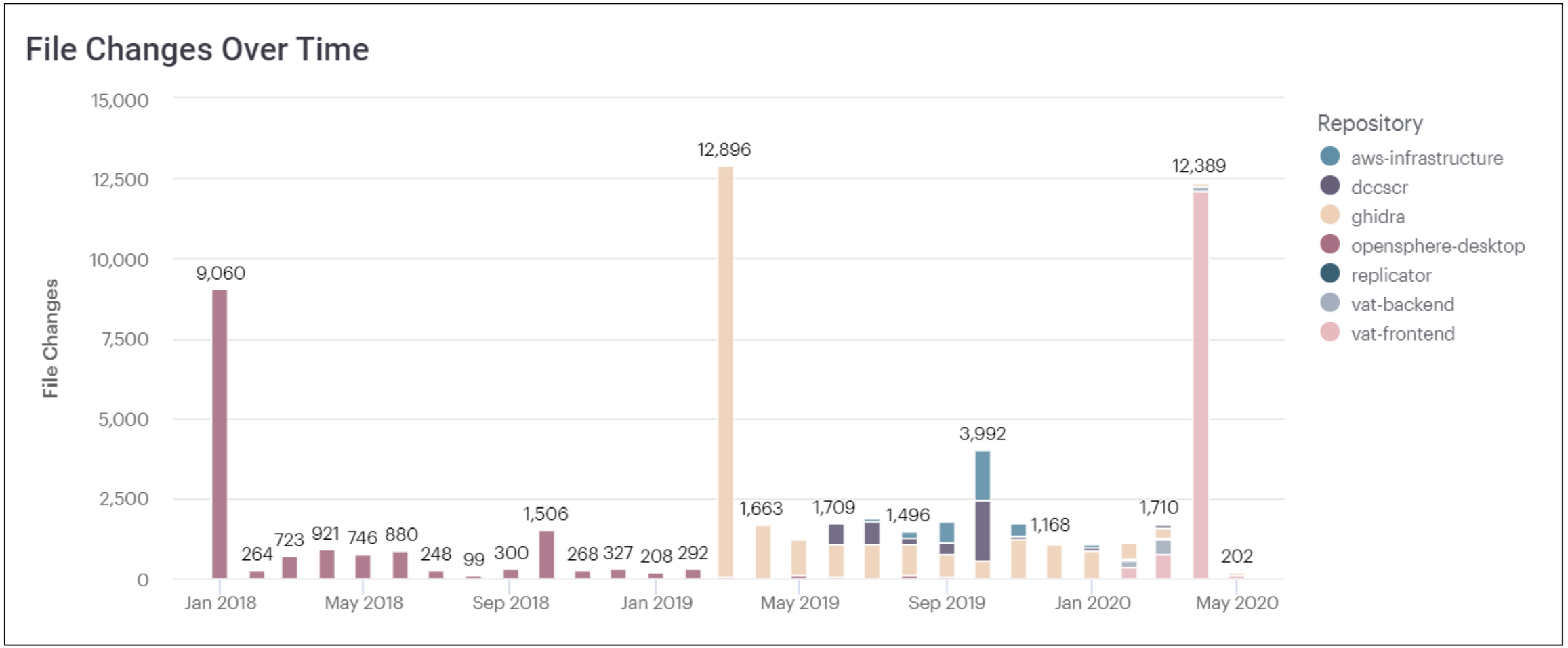
Information - 3 large file change events are evident, which appear to be imports from other storage locations or mass updates

Commit Analysis - Commits Over Time



sema

Commit Analysis - Commits Over Time



sema

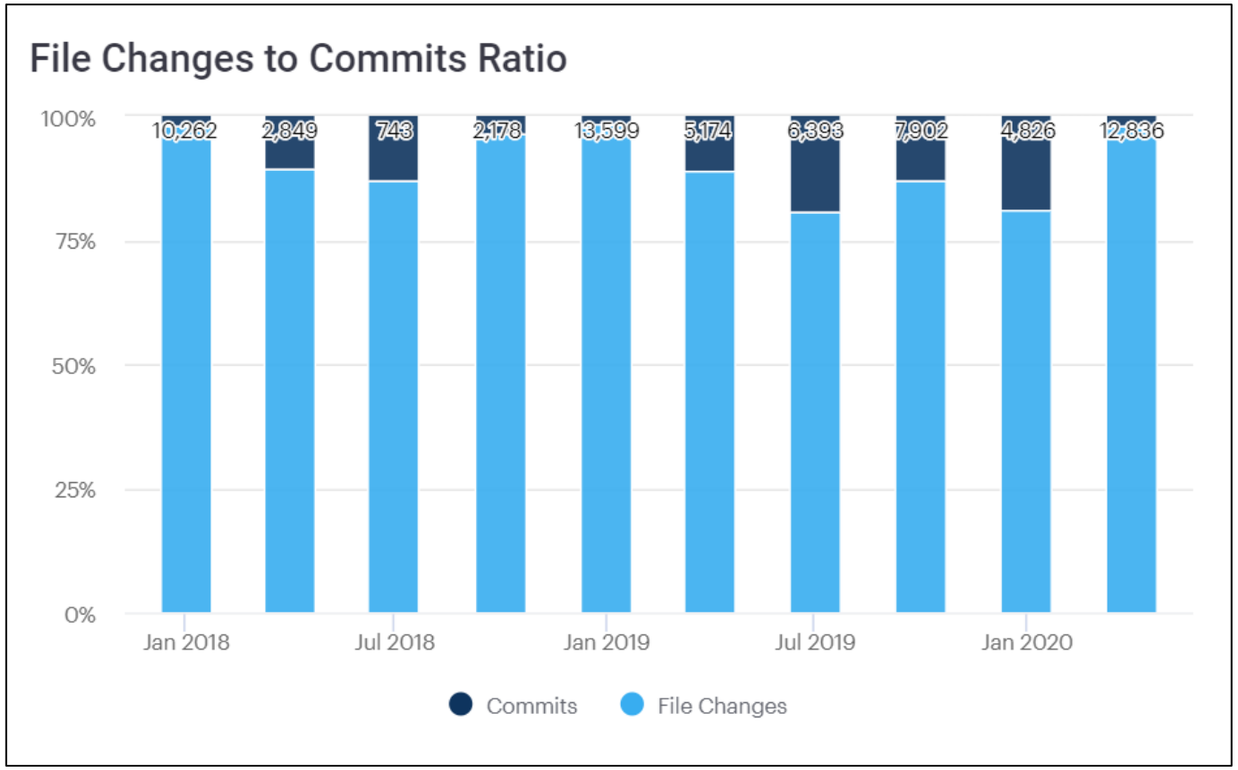
Commit Management

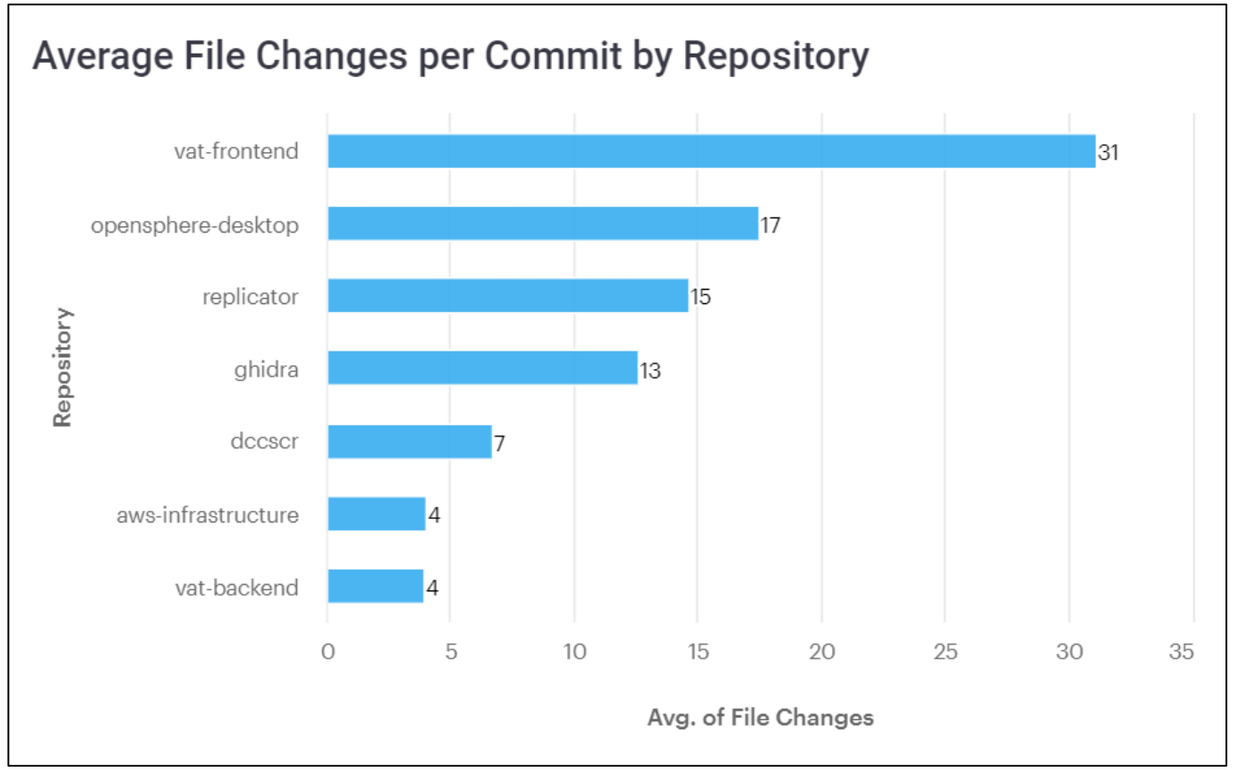
A good process for committing code improves the quality and ease of maintenance of a code base. Sema analyzes an aspect of this, the average number of files per commit.

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.

Strength- 4 of the 7 repositories have more than 10 files per commit, due to one-off administrative changes discussed above. Excluding these one-off events, files per commit are in the appropriate range.

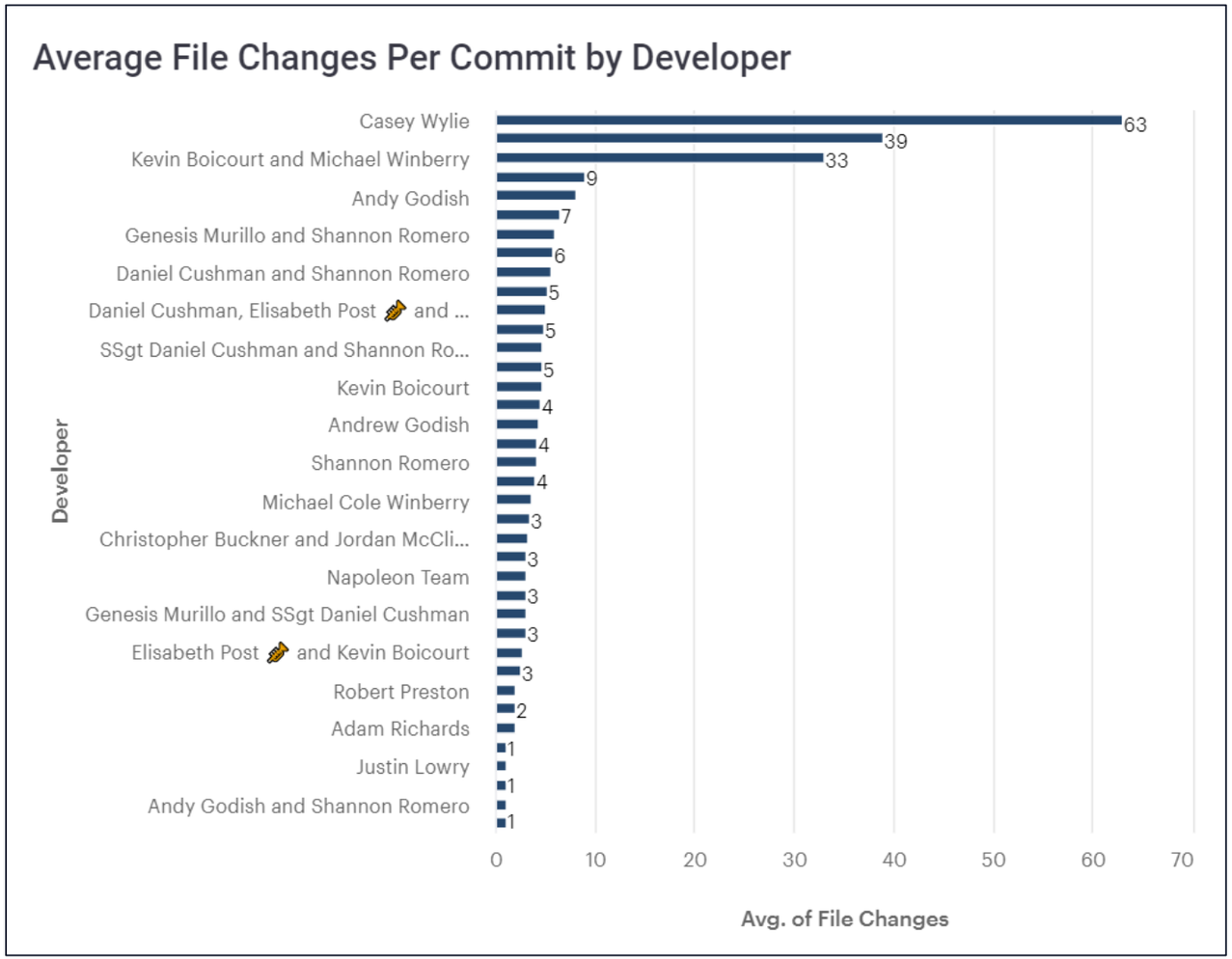
File Changes per Commit Ratio





sema

Average File Changes per Commit Per Developer - Iron Bank



- 3 developers have above 25 average files per commit - these are likely due to administrative tasks
- Strength - no developers with commit ratio between 10-25 files/commit indicating process discipline to submit fewer files/commit excluding administrative commits

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.

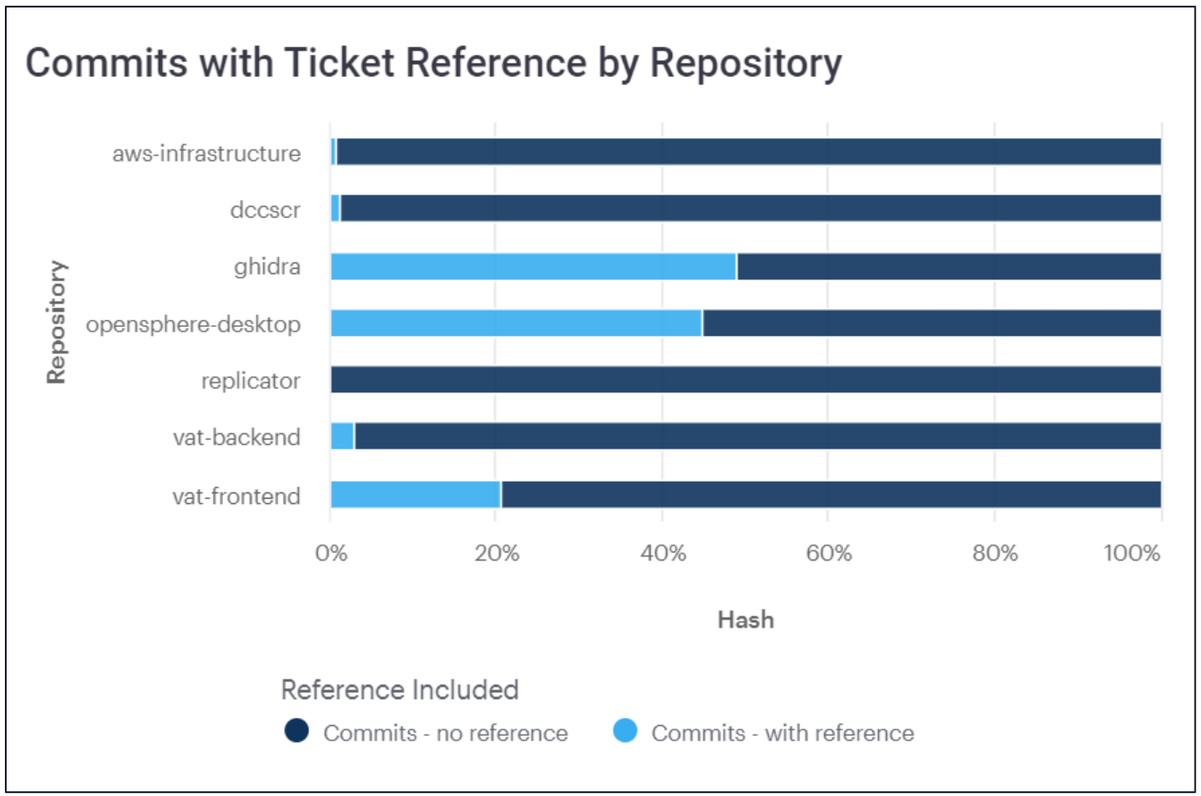
Adding testing as development occurs is the one of the easiest ways to minimize technical debt and drive quality in the code.

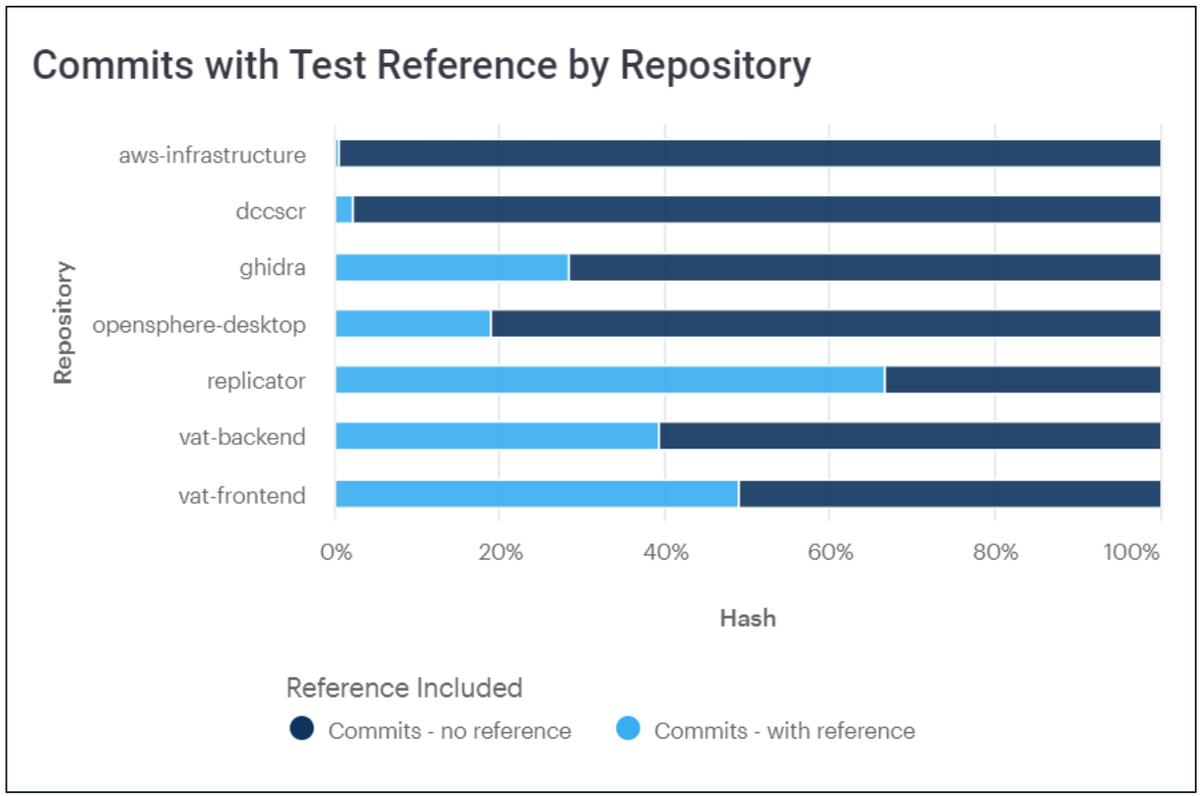
Sema recommends that over 60% of commits have ticket references and over 40% of commits have test references so that testing is included as development occurs.

Low risk - All products have a ticket reference below 60% in the last year. Is this in line with process goals of each organization for open-source code? For internal code?

Investigate - 5 of 7 products have fewer than 40% of their commits with test files in the last year. See discussion above about testing

Commits with Tickets Reference and Test Commits - Last Full Year





sema

- Code
- Process
- **Team**
- Appendix

Questionnaire for building a Developer Coaching and Evaluation plan based on quantitative and qualitative data

- Is individual developer activity measured?
- 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 (usually no; expected… but now you will be able to say that you do) as well as qualitatively?
- Among developers identified as highest knowledge (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 with the organization? 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, what is the status of those developers? Do they still work at the organization? Have they been recognized/ compensated?
- Among developers identified as lowest skill (10-20%) by Sema's analysis…
  - Do you agree or can you explain?
  - 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-the organization developers (low knowledge)

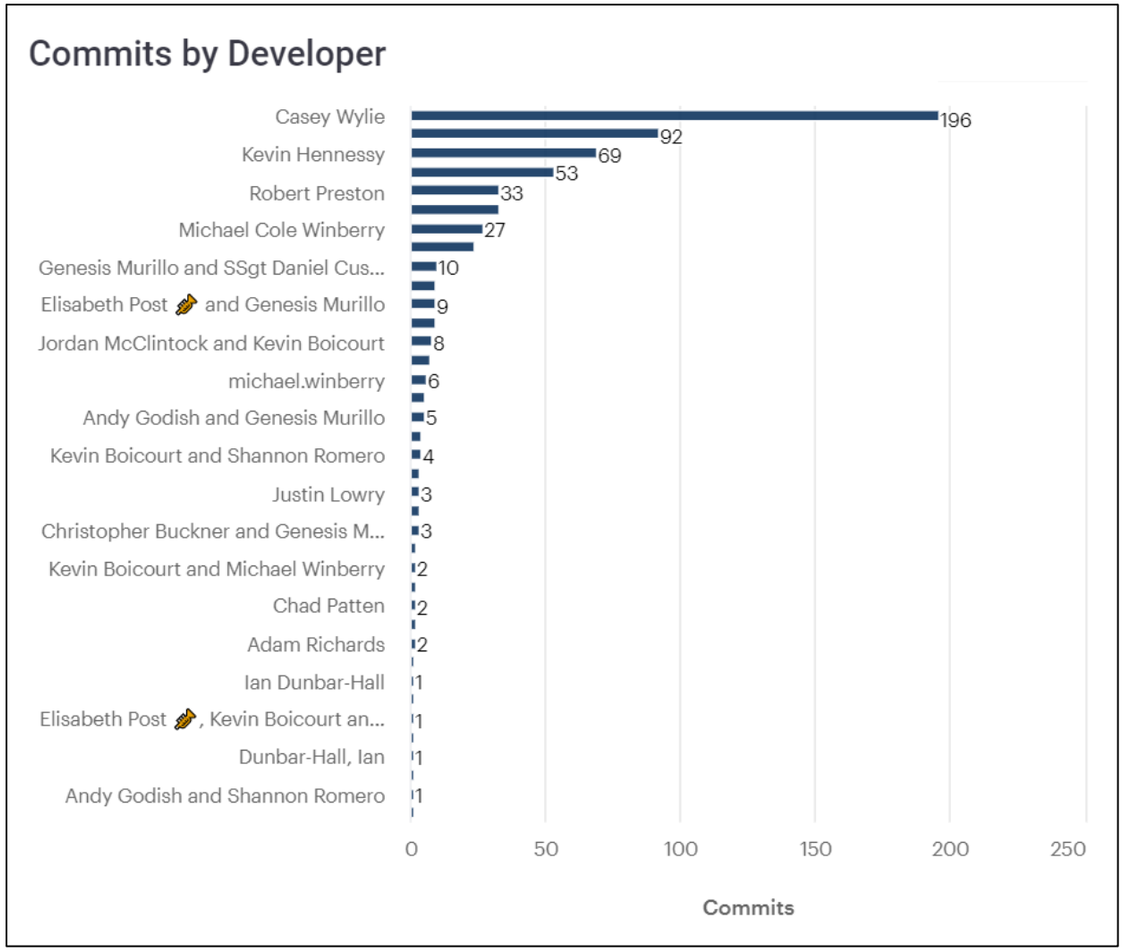
Developer Activity and Expertise

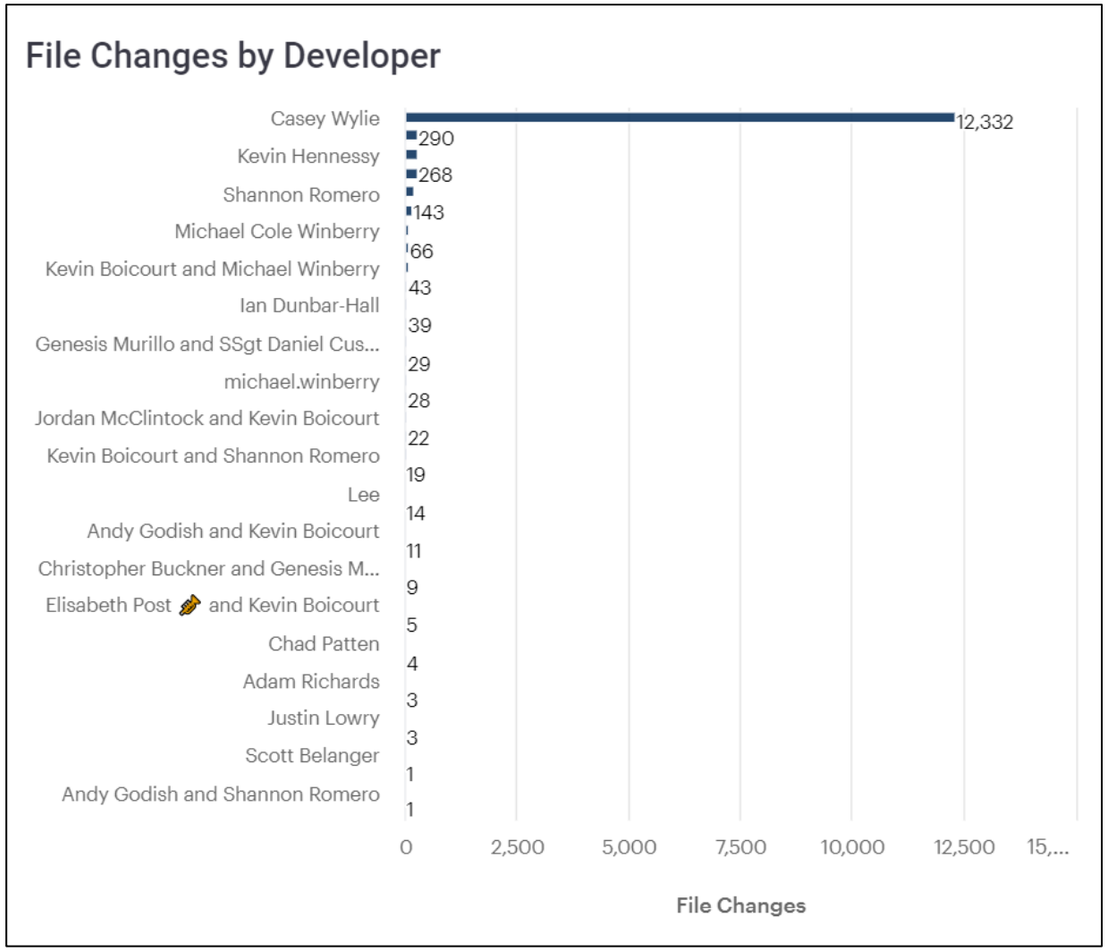
Sema recommends setting consistent commit activity goals over time and manage towards them, with respect to code priorities.

Based on current agile methodology, Sema recommends maximizing the number of full stack developers on a team.

Observation - Pair programming is as a consistent practice in Iron Bank repositories

Commits and File Changes per Developer Iron Bank

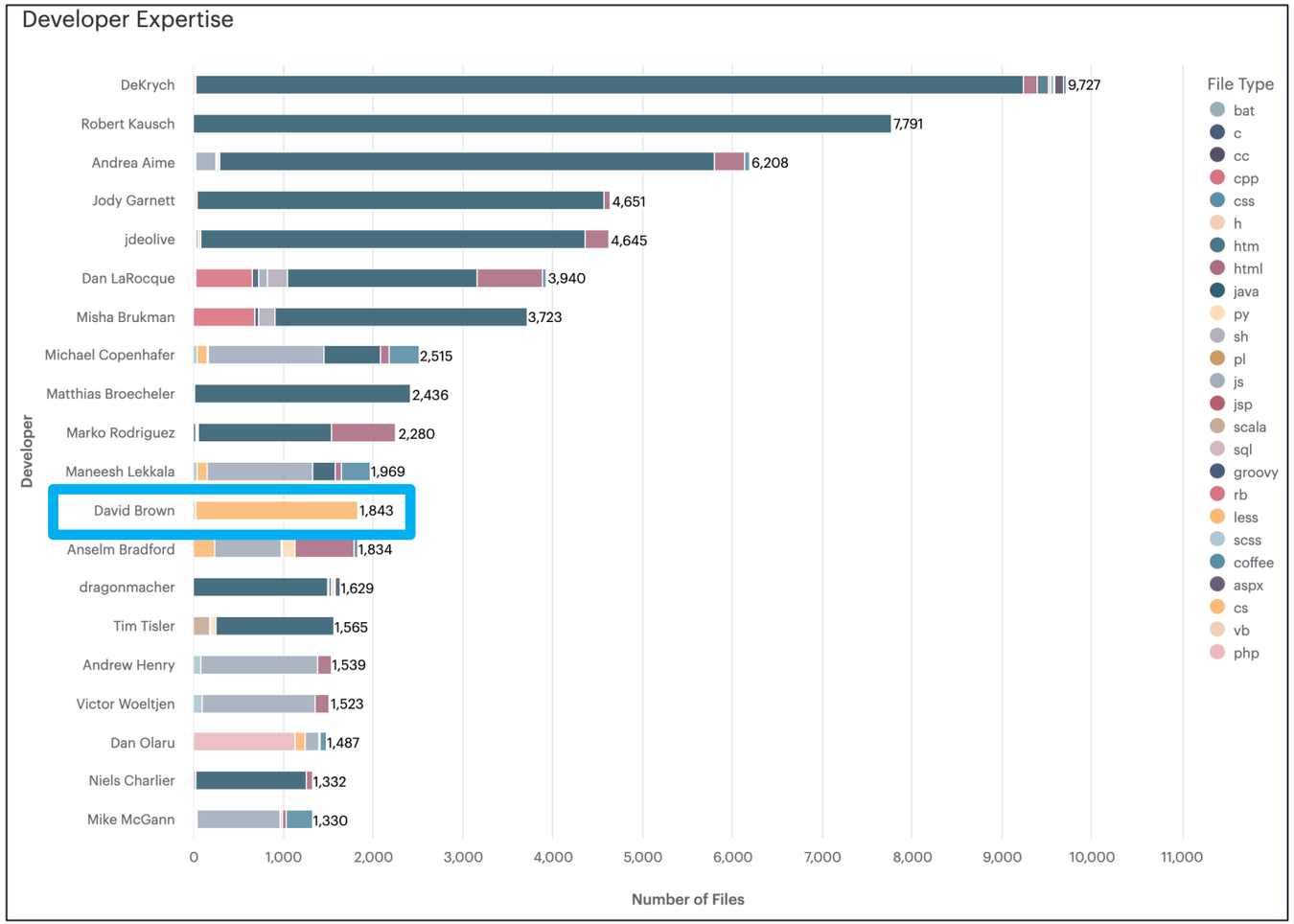




Casey W is carrying out administrative functions in these codebases

sema

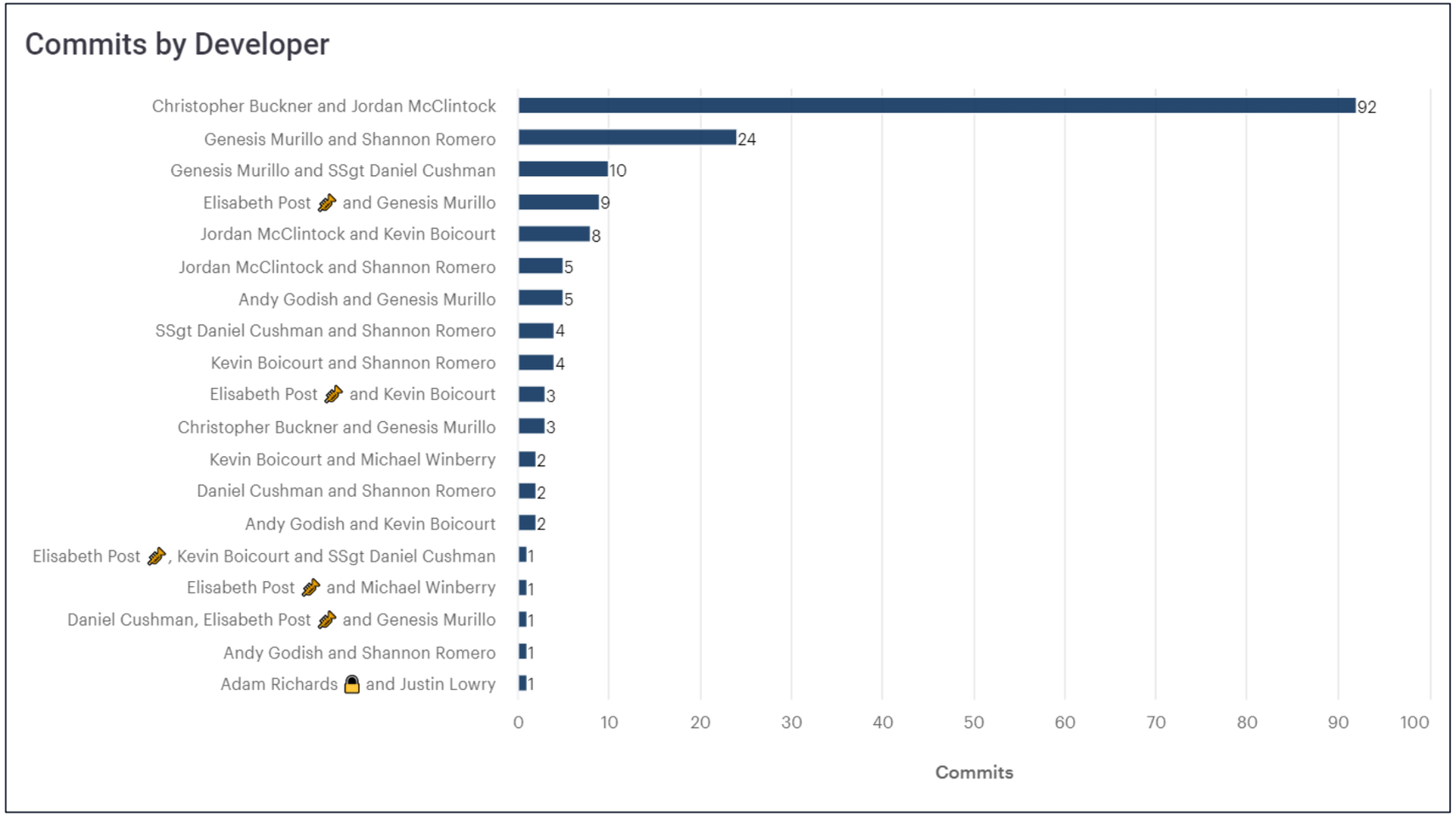
Developer Expertise



- One of the ways to match a developer with a suitable project is to understand where they have the most language expertise
- For example, David Brown has the most csharp experience

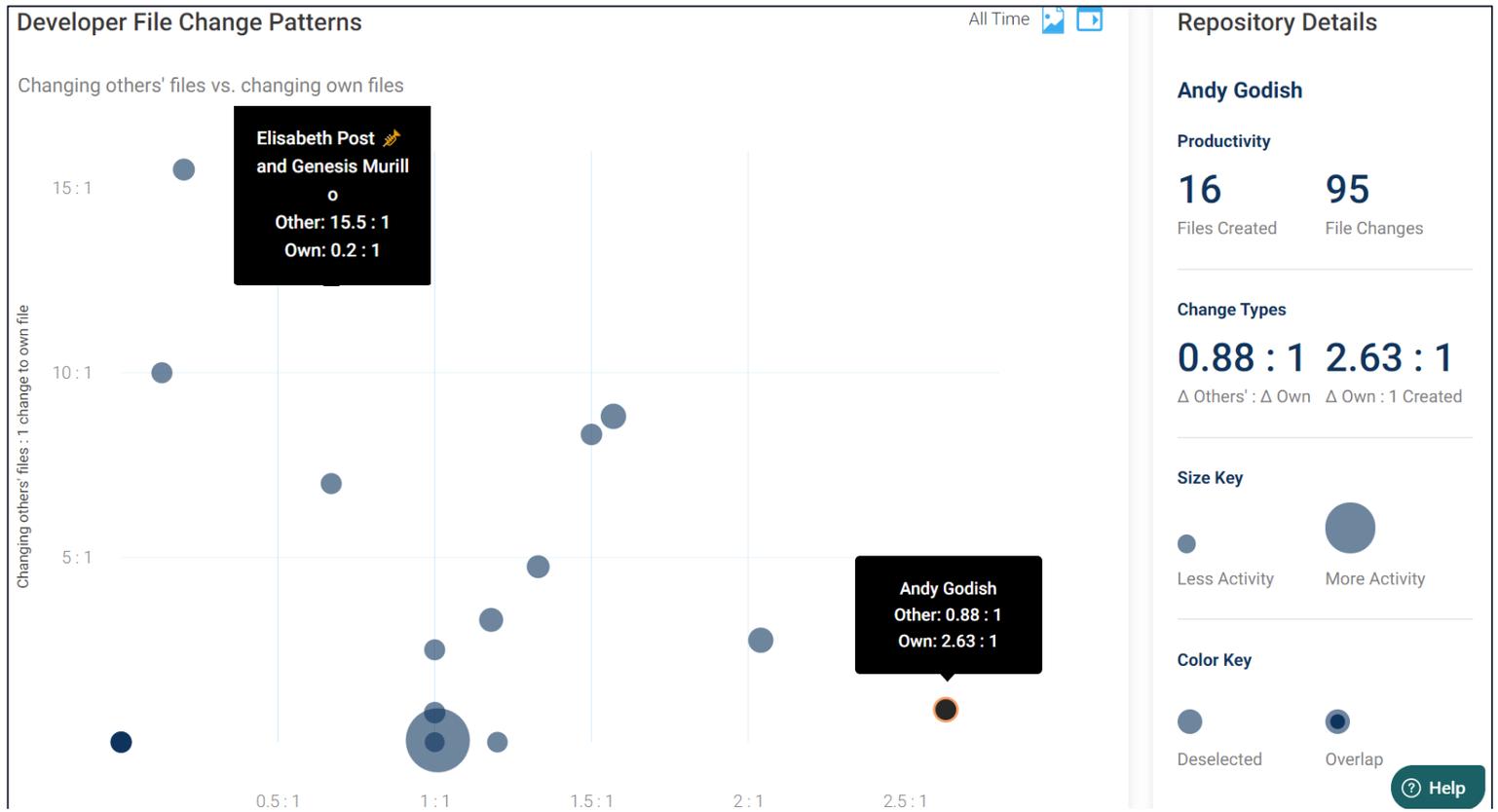
sema

Detail: Pair Programming among Iron Bank applications



sema

Elisabeth P. and Genesis M. are “Editors” while Andy Godish is a “Self-Perfecter” - vat-frontend



- Elisabeth P and Genisis M are “Editors”: changes others’ code 15 times more than changing own code
- Andy G is a “Self-Editor”: changes own code more than creating code, relative to colleagues, but not relative to outside standards
- Developer activity should vary based on organizational role- organizations can use this information to help coach developers, especially in large 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

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

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

Contribution - Ghidra: DeKrych created ghidra, others have edited it

Developer	first commit count	changes to others code count	changes to own code count	changes to other code % of initial	changes to any code % of initial	files created count	total expertise per repo
DeKrych	500	5	1,000	-	10	1,000	1,000
dev747368	1	451	6	417	433	2	33
dragonmacher	12	1,000	45	88	105	24	57
caheckman	1	330	7	328	352	2	91
Paul Moran	0	19	0	500	500	0	36
ghidra1	11	346	31	124	146	23	199
Ryan Kurtz	4	304	9	181	200	7	127
ghidravore	4	249	14	62	77	9	19

- The ways that developers write code are tracked line by line and can be analyzed to understand who does what
- For example, DeKrych was by far the biggest initial writer of ghidra
- dev747368 and dragonmacher are major editors

Skill - Ghidra: Sandor N. and emteere write very clean code

Developer	Environment Sensitive	Misleading	Performance	Potential Bug	Security	Smell	Stylistic	Total
Sandor Nemes	100	100	100	100	100	81	13	594
emteere	100	90	100	100	100	75	23	588
Benjamin Levy	100	100	100	100	100	63	25	588
ghizard	100	75	100	100	100	83	26	583
Ryan Kurtz	90	88	100	100	100	79	23	580
adamopolous	92	86	100	100	100	66	27	570
Xiaoyin Liu	100	100	100	100	100	-	67	567
caheckman	100	68	100	100	100	59	38	566
dragonmacher	90	74	100	100	100	78	22	564
ghidorahrex	100	60	100	100	100	71	31	562
ghidravore	93	68	100	100	100	76	24	561
ghidra1	89	72	100	100	100	71	27	560
dev747368	74	80	100	100	100	74	24	552
saruman9	100	46	100	100	100	40	50	536
Anciety	100	-	100	100	100	96	21	517
James	-	64	100	100	100	63	35	461
DeKrych	85	74	-	-	-	77	24	260

- In addition to the quantity of work, code can be analyzed to determine developer quality
- For ghidra, Sandor N. and emteere wrote code with the least warnings = the cleanest code

DeKrych, Paul M., and Mumbel are the top developers on ghidra by skill

Developer	Skill rank	Knowledge rank
DeKrych	100	100
Paul Moran	73	9
mumbel	73	2
astrelsky	45	2
Christian Blichmann	45	0
FergoFrog	45	0
Andrew Briggs	45	1
Sandor Nemes	43	0

- Skill and knowledge of each developer can be weighted differently based on the Mission goals
- For ghidra, DeKrych both has the highest skill ranking and knows the most about the code
- Paul M and mumble are also high skill developers

- Code
- Process
- Team
- **Appendix**

Description of each Application, by Agency

#	Agency	Application	Application Description
1	CDC	Epi-Info-Community-Edition	Epi Info™ is used worldwide for the rapid assessment of disease outbreaks; for the development of small to mid-sized disease surveillance systems; as ad hoc components integrated with other large scale or enterprise-wide public health information systems; and in the continuous education of public health professionals learning the science of epidemiology, tools, and techniques.
2	CFPB	cfgov-refresh	The master repository for consumerfinance.gov. This Django project includes the front-end assets and build tools, Jinja templates for front-end rendering, code to configure our CMS, Wagtail, and several standalone Django apps for specific parts of the site.
3	Department of Agriculture	ADSM	A simulation of disease spread in livestock populations. Includes detection and containment simulation.
4	Department of Commerce	project-open-data.github.io	This Appendix is meant to be a living document so that collaboration in the open data ecosystem is fostered and the continual update of technology pieces that affect update can happen on a more rapid pace
5	Department of Defense	Anet	The Advisor Network ("ANET") is a tool to track relationships between advisors and advisees.
6	Department of Defense	Dshell	An extensible network forensic analysis framework. Enables rapid development of plugins to support the dissection of network packet captures.
7	Department of Defense	send	A file sharing experiment which allows you to send encrypted files to other users.
8	Department of Homeland Security	pshtt	pshtt was developed to push organizations — especially large ones like the US Federal Government — to adopt HTTPS across the enterprise
9	GSA	calc	CALC is a tool to help contracting personnel estimate their per-hour labor costs for a contract, based on historical pricing information. The tool is live at https://calc.gsa.gov. You can track progress or file an issue on this repo; see our contributor guidelines.
10	GSA	data.gov	Data.gov is an open data website created by the U.S. General Services Administration that is based on two robust open source projects: CKAN and WordPress.
11	GSA (18F)	cloud-gov	18F is a digital services agency within the United States Government. Their purpose is to deliver digital services and technology products.
12	GSA (18F)	credhub	CredHub centralizes and secures credential generation, storage, lifecycle management, and access
13	GSA (18F)	dol-whd-14c	The 14(c) system will become a modern, digital-first service. Applicants will be provided an intuitive online experience, guiding them through the information needed to complete their application correctly.
14	NASA	openmct	Open MCT (Open Mission Control Technologies) is a next-generation mission control framework for visualization of data on desktop and mobile devices.
15	NASA	worldview	Worldview uses OpenLayers to display imagery from the Global Imagery Browse Services (GIBS).
16	NGA	elasticgeo	ElasticGeo provides a GeoTools data store that allows geospatial features from an Elasticsearch index to be published via OGC services using GeoServer.
17	NGA	geoserver	GeoServer is an open source software server written in Java that allows users to share and edit geospatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards.
18	NGA	janusgraph	JanusGraph is a highly scalable graph database optimized for storing and querying large graphs with billions of vertices and edges distributed across a multi-machine cluster. JanusGraph is a transactional database that can support thousands of concurrent users, complex traversals, and analytic graph queries.
19	NGA	mrgei	MrGeo is a geospatial toolkit designed to provide raster-based geospatial capabilities that can be performed at scale. MrGeo is built upon Apache Spark and the Hadoop ecosystem to leverage the storage and processing of hundreds of commodity computers. See the wiki for more details. https://github.com/ngageoint/mrgeo/wiki
20	NGA	opensphere-desktop	Open Sphere is an application used to visualize temporal/geospatial data.
21	NSA	lemongraph	LemonGraph is a log-based transactional graph (nodes/edges/properties) database engine that is backed by a single file.
22	Treasury	data-act-broker-backend	The DATA Act Broker backend is a collection of services that power the DATA Act's central data submission platform.
23	US Customs	GTAS	Global Travel Assessment System | A passenger data screening and analysis system for enhancing global security https://us-cbp.github.io/GTAS/