Carnegie Mellon University
Software Engineering Institute

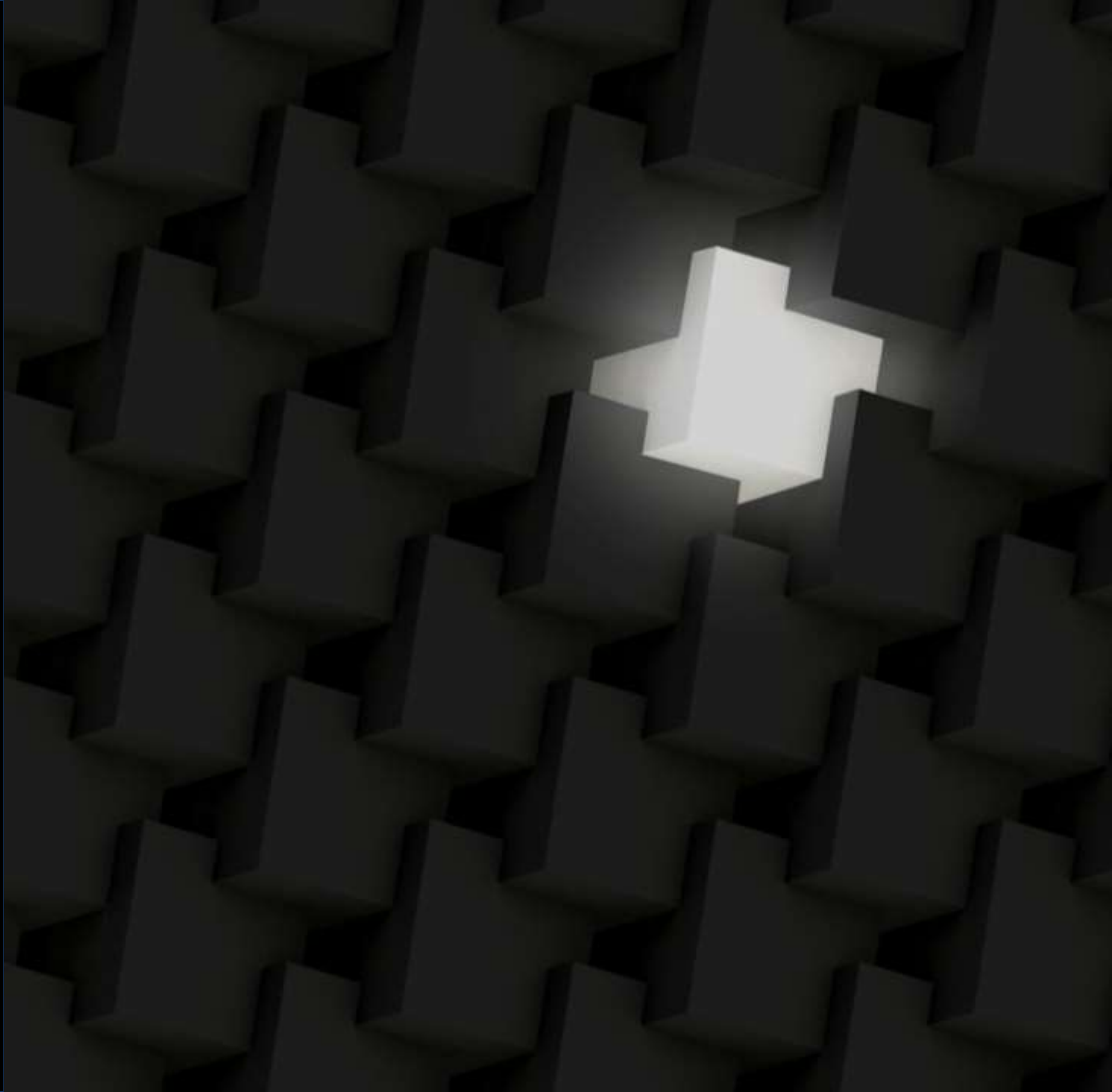
SCAIFE and Static Analysis Classification Research

Presentation for NASA Software Engineering & Assurance Working Group

Sept. 25, 2020

Presenter: Dr. Lori Flynn (PI)

FY20 Team: Ebonie McNeil, Matt Sisk, David Svoboda, Hasan Yasar, Joseph Yankel, David Shepard, and Shane Ficorilli



This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

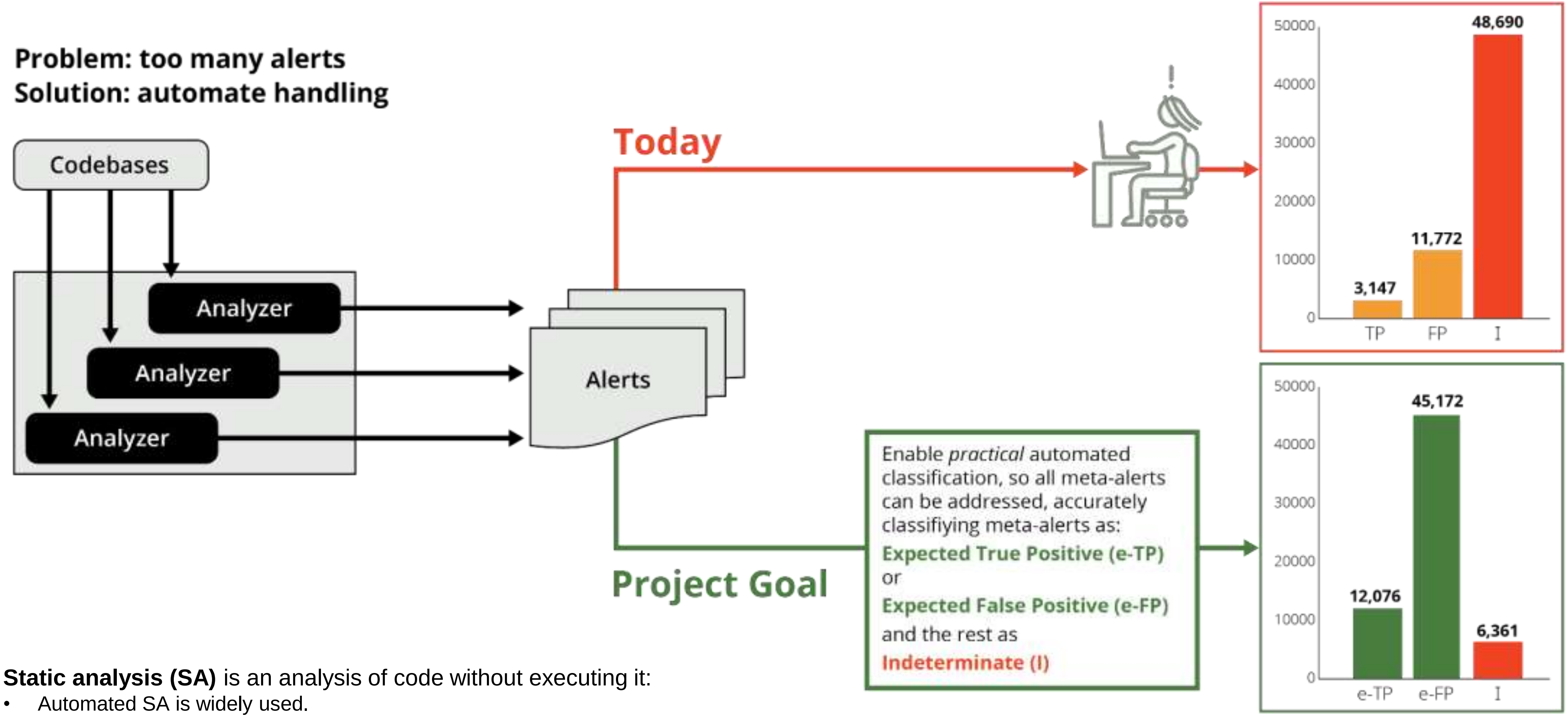
DM20-0867

Overview

Definitions: An *alert* is an SA warning (with a checker ID, line #, filepath, message); an *alertCondition* is an alert mapped to a code flaw taxonomy item (e.g., CWE-190); and a *meta-alert* is mapped to by the set of all alertConditions that differ only by checker ID. We do classification and adjudication at the meta-alert level.

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

Problem: too many alerts
Solution: automate handling



Enable *practical* automated classification, so all meta-alerts can be addressed, accurately classifiying meta-alerts as:
Expected True Positive (e-TP)
or
Expected False Positive (e-FP)
and the rest as
Indeterminate (I)

Static analysis (SA) is an analysis of code without executing it:

- Automated SA is widely used.
- It is a normal part of testing by DoD and commercial organizations.

Static Analysis Classification Research FY16-20

Five Years in Two Slides

FY16-19 Static Analysis Meta-Alert Classification Research

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed.

FY16

Improve classifier precision & recall
Data quality
Wide variety of labeled data
Enable classifier use via modular architecture
Enable classifier use in CI systems

- Issue addressed: classifier accuracy
- Novel approach: use **multiple static analysis tools as features**
- Result: increased accuracy

FY17

Improve classifier precision & recall
Data quality
Wide variety of labeled data
Enable classifier use via modular architecture
Enable classifier use in CI systems

- Issues addressed: **data quality**, **too little labeled data** for accurate classifiers for some conditions (e.g., CWEs, coding rules)
- Novel approach: **audit rules+lexicon; use test suites to automate the production of labeled** (True/False) **meta-alert data* for many conditions**
- Result: high precision for more conditions

FY18-19

Improve classifier precision & recall
Data quality
Wide variety of labeled data
Enable classifier use via modular architecture
Enable classifier use in CI systems

- Issue addressed: **little use of automated meta-alert classifier technology** (requires $$, data, experts)
- Novel approach: **develop an extensible architecture with a novel test-suite data method**
- Result: **wider use of classifiers (less $$, data, experts)** with an extensible architecture, API, software to instantiate architecture, and adaptive heuristic research

* By the end of FY18, ~38K new labeled (T/F) meta-alerts from eight SA tools on the Juliet test suite (vs. ~7K from CERT audit archives over 10 years)

FY20 Static Analysis Meta-Alert Classification Research

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort.

FY20 (of a two-year project, FY20-21)

Improve classifier precision & recall
Data quality
Wide variety of labeled data
Enable classifier use via modular architecture
Enable classifier use in CI systems

- Issue addressed: It takes too much time to adjudicate (i.e., audit) static analysis meta-alerts during continuous integration (CI).
- Novel approach: During CI builds, use **classifiers** with **precise cascading** and **CI/CD features**.
- Results
  - Design for CI-SCAIFE system integration
  - SCAIFE System v 1 release (classifier defined, run, and results can be viewed from [G]UI module)
  - Defined cascading API
  - Less-precise cascading using the API
  - Test results for less-precise cascading
  - Significant progress on CI-SCAIFE system integration development
  - Deployment and testing by DoD collaborators (multiple rounds)
  - A published RC_Data open dataset for improved classifier research
  - APIs, technical manuals, and SCALe public publication
- FY21 plan: a precise cascading algorithm, improved classifiers, full integration

Data Quality: Lexicon and Rules

- We developed a **lexicon** and auditing **rule set** for our collaborators.
- It includes a standard set of well-defined **determinations** for static analysis meta-alerts.
- It also includes a set of **auditing rules** to help auditors make consistent decisions in commonly encountered situations.

Different auditors should make the **same determination** for a given meta-alert.

Improve the **quality and consistency** of audit data for the purpose of building **machine learning classifiers**.

Help organizations make **better-informed** decisions about **bug fixes**, **development**, and **future audits**.

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

Improve classifier precision & recall

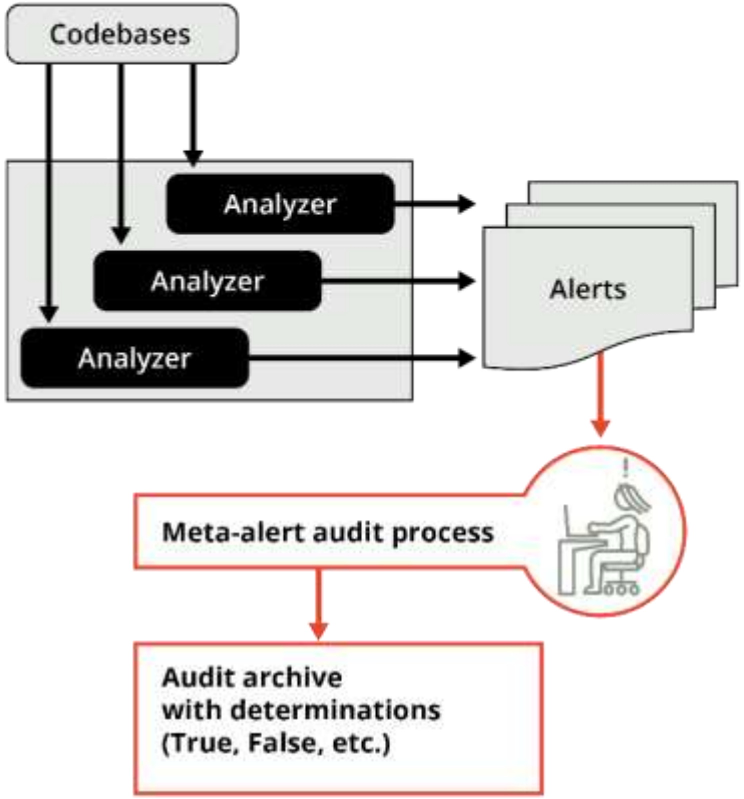
Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

SEI SCALe Framework: Background



Static Analysis Meta-Alert Auditing Framework
Developed by the SEI for ~10 years.

- GUI front end to examine meta-alerts and associated code
- Meta-alert adjudications (true, false) stored in database

Use for Research Projects

- We enhance the framework with features for research.
- Collaborators use it on their codebases.
- Researchers analyze audit data.

After running SA tools, meta-alert adjudication can happen at any point in the software development lifecycle.

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

Prioritization Schemes

Practical use of classification

Improve classifier precision & recall

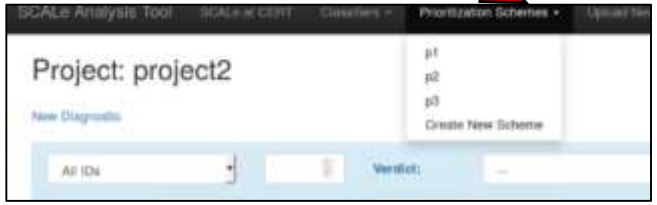
Data quality

Wide variety of labeled data

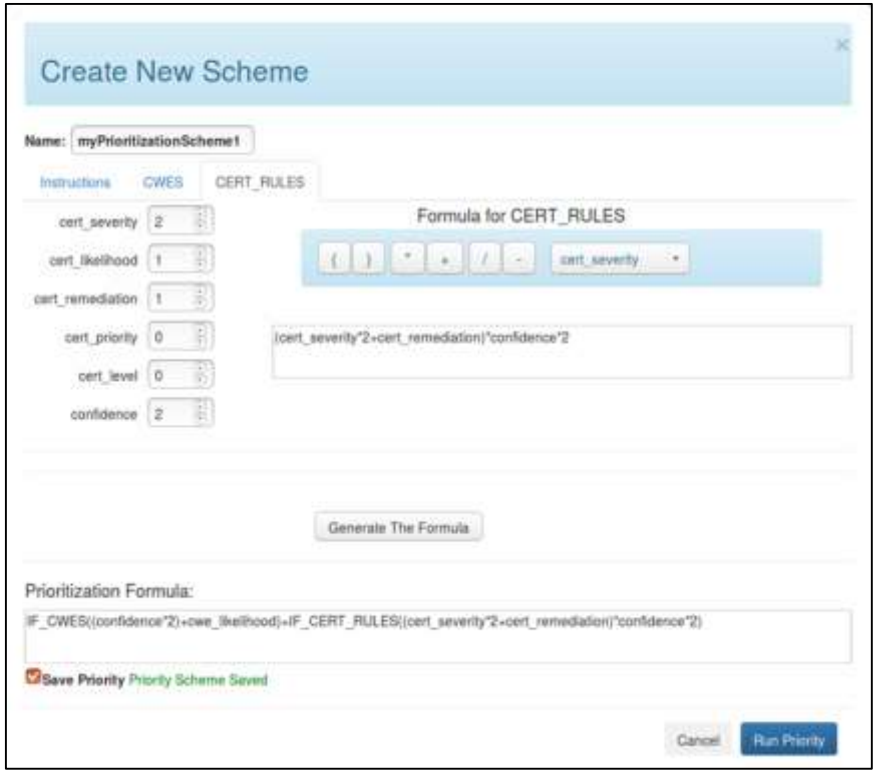
Enable classifier use via modular architecture

Enable classifier use in CI systems

Prioritization schemes with mathematical formulas user can create and/or use



Project: project2



Create New Scheme

Name: myPrioritizationScheme1

cert_severity	2
cert_likelihood	1
cert_remediation	1
cert_priority	0
cert_level	0
confidence	2

Formula for CERT_RULES

(cert_severity*2+cert_remediation)*confidence*2

Prioritization Formula:

IF_CWES((confidence*2)+cwe_likelihood)+IF_CERT_RULES((cert_severity*2+cert_remediation)*confidence*2)

☑ Save Priority Priority Scheme Saved

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

User Field Uploads

Practical use of classification

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

User field uploads

- These uploads are for advanced users who can work with SQLite databases and generate values.
- Uploaded fields can be used in priority schemes.
- The CSV uploaded file has the following:
  - One line per project meta-alert ID
  - A left-most field with a meta-alert ID
  - A top row that holds field labels

```
meta_alert_id,safeguard_countermeasure,
vulnerability,residual_risk,impact,
threat,risk,complexity,severity,coupling
112,5,1,4,9,1,1,5,5,1
2,9,3,3,3,1,1,1,9,3
3,3,1,1,1,8,1,5,5,1
4,6,1,1,5,2,1,8,8,1
5,2,1,1,2,3,1,7,7,5
6,5,1,4,4,1,2,4,5,1
7,8,5,3,4,8,2,4,9,9
8,2,1,3,2,8,3,8,8,1
9,6,4,3,6,9,1,4,4,4
10,3,2,2,5,7,1,4,5,9
11,6,1,1,9,6,1,7,7,1
12,2,8,4,1,6,1,4,4,8
```

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Archive Sanitizer for Collaborator Data Sharing

We added a data sanitizer to SCALe that has the following functions:

- Anonymizes sensitive fields
- Has an SHA-256 hash with salt
- Enables analysis of features correlated with meta-alert confidence

The audit archive for the project is in a database:

- DB fields may contain sensitive information.
- The sanitizing script anonymizes or discards fields:
  - Diagnostic message
  - Path, including directories and filename
  - Function name
  - Class name
  - Namespace/package
  - Project filename

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

Practical use of classification

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Analysis of Juliet Test Suite: Initial 2018 Results

Automated Adjudication	Labeled Meta-Alert (counts a fused alertCondition once)
TRUE	**13,330**
FALSE	**24,523**

Lots of new data for creating classifiers

(37,853 labeled meta-alerts)

Big savings: a manual audit of 37,853 meta-alerts from non-test-suite programs would take an unrealistic minimum of 1,230 hours (117 seconds per meta-alert audit*).

- The first 37,853 meta-alert audits wouldn't cover many conditions (and sub-conditions) covered by the Juliet test suite.
- We needed true and false labels for classifiers.
- **Realistically**, an enormous amount of manual auditing time is required to develop that much data.

These are initial metrics; we will collect more data as we use more tools and test suites.

*N. Ayewah and W. Pugh. "The Google FindBugs Fixit", *International Symposium on Software Testing and Analysis,* ACM, 2010.

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

SCAIFE Definitions

SCAIFE is **a modular architecture that enables static analysis meta-alert classification** plus advanced prioritization.

- The **SCAIFE API** defines interfaces between the modular parts.
- **SCAIFE systems** are software systems that instantiate the API.
- Our SCAIFE system releases include a SCALe module plus much more.

SCAIFE = Source Code Analysis Integrated Framework Environment

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

SCAIFE Architecture Approach

For efficient development of a robust API to enable widespread classifier use, we need a system architecture that:

- Integrates with existing static analysis tools and aggregators (including SCALe)
- Supports classification and adaptive heuristic functionality
- Demonstrates fast response times for average and worst-case scenarios
- Provides extensibility for future research in static analysis, classification, architecture, and SecDevOps

Swagger/OpenAPI Open-Source Development Toolset

- Quickly develops APIs following the OpenAPI standard
- Auto-generates code for servers and clients in many languages
- Tests server and client controllers with Swagger UI
- Is widely used (10,000 downloads/day)

- Big O analysis was useful.
- Design decisions required balancing goals and analyzing tradeoffs.

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

SCAIFE Architecture

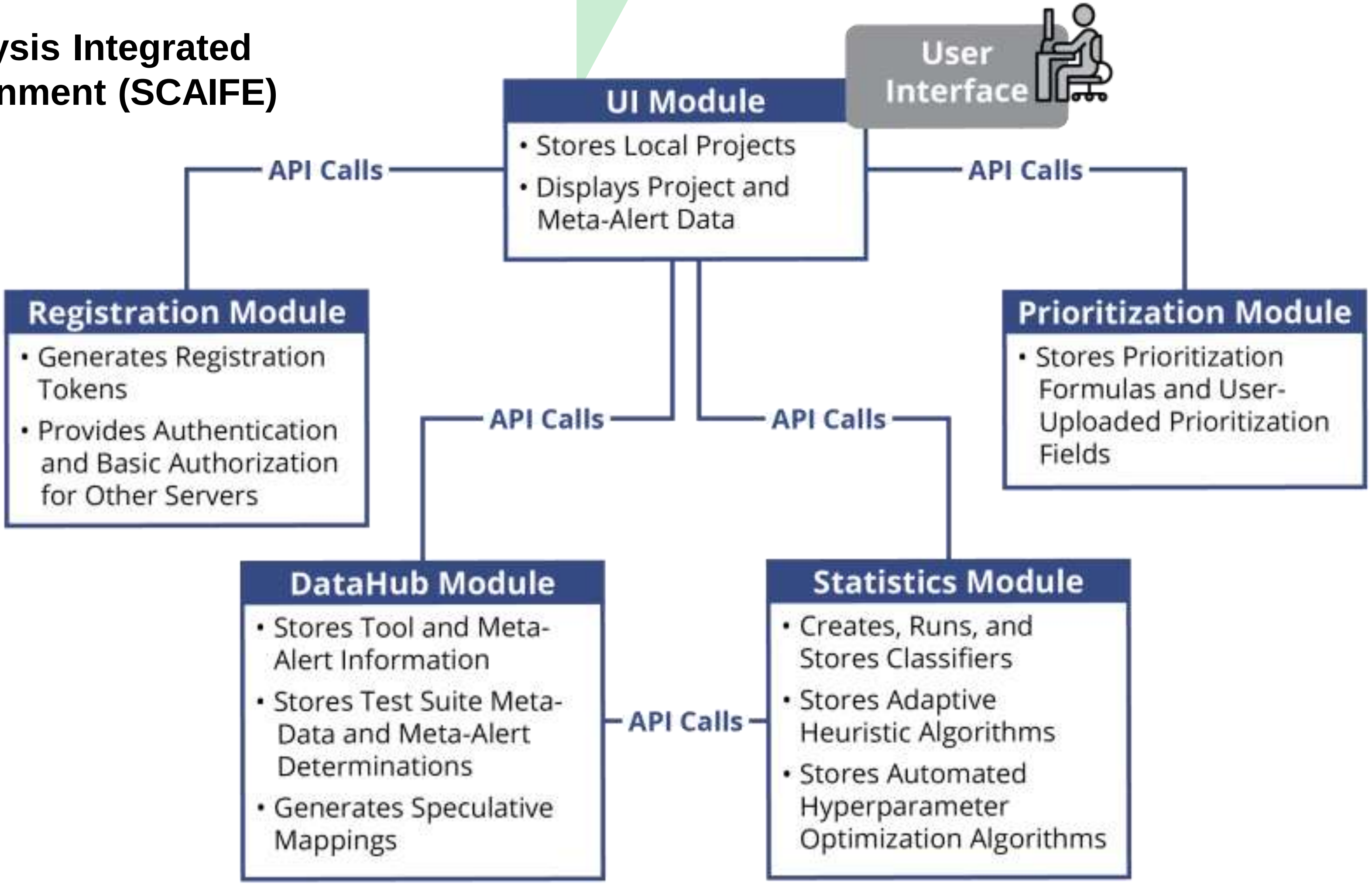
Source Code Analysis Integrated Framework Environment (SCAIFE)

Any static analysis tool can instantiate APIs to become a UI Module. For example

- SEI SCALe
- DHS SWAMP
- CCDC C5ISR SwAT
- Other aggregator tools
- Single static analysis tools

SCAIFE is a modular architecture that **enables users to efficiently start to use classifiers with a wide variety of systems and tools:**

- The formal SCAIFE API definition enables automated code generation to quickly instantiate API calls and generate server stubs in many code languages. This reduces the effort required to integrate existing systems and tools.
- The UI Module instantiation of SCALe is publicly available (GitHub scaife-scale branch).
- Collaborators can get a full SCAIFE instantiation and use it as-is or substitute any module(s) and use the others.



User Interface

UI Module

- Stores Local Projects
- Displays Project and Meta-Alert Data

Registration Module

- Generates Registration Tokens
- Provides Authentication and Basic Authorization for Other Servers

Prioritization Module

- Stores Prioritization Formulas and User-Uploaded Prioritization Fields

DataHub Module

- Stores Tool and Meta-Alert Information
- Stores Test Suite Meta-Data and Meta-Alert Determinations
- Generates Speculative Mappings

Statistics Module

- Creates, Runs, and Stores Classifiers
- Stores Adaptive Heuristic Algorithms
- Stores Automated Hyperparameter Optimization Algorithms

Improve classifier precision & recall

Data quality

Wide variety of labeled data

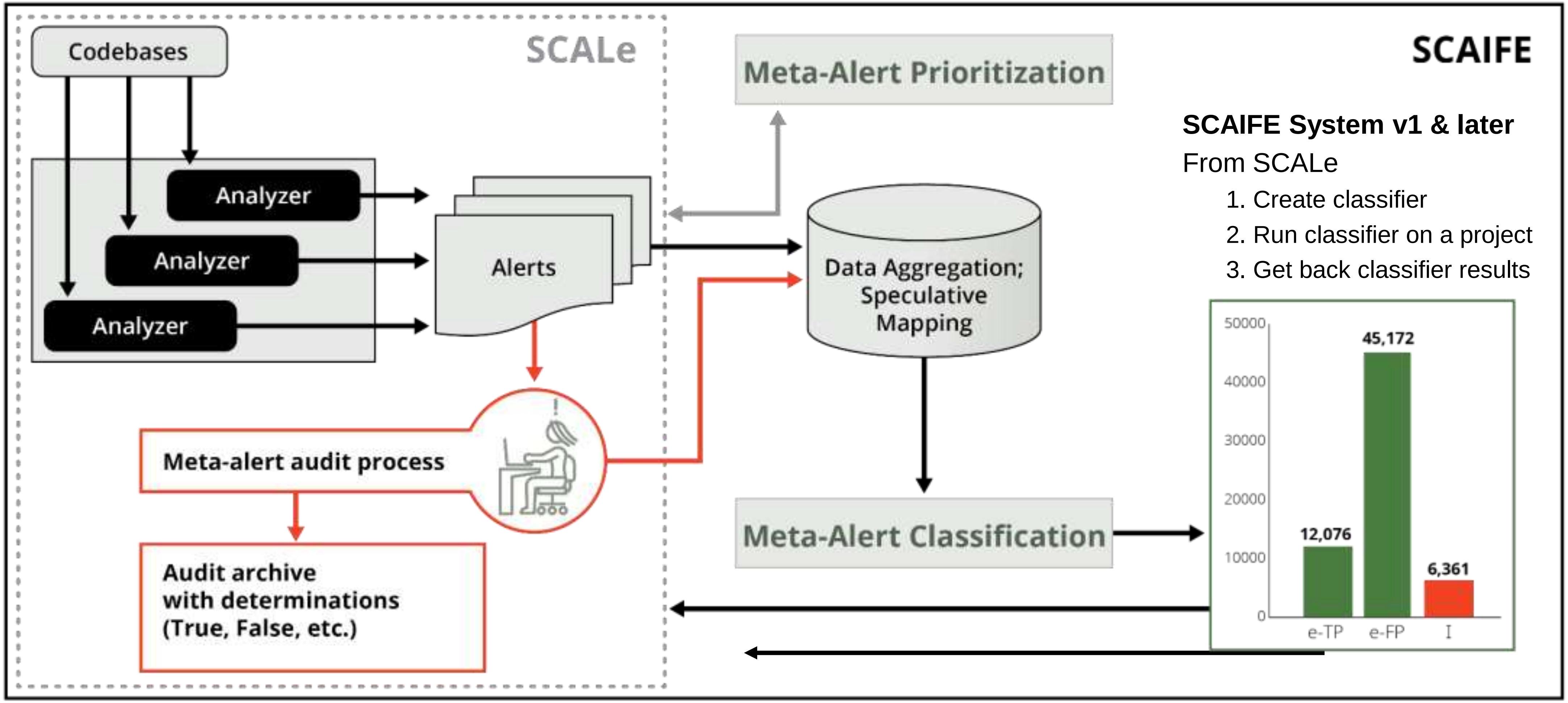
Enable classifier use via modular architecture

Enable classifier use in CI systems

L. Flynn, E. McNeil, and J. Yankel. "How to Instantiate SCAIFE API Calls: Using SEI SCAIFE Code, the SCAIFE API, Swagger-Editor, and Developing Your Tool with Auto-Generated Code." SEI Technical Manual. July 2020.

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

SCAIFE Meta-Alert Dataflow with SCALe Module



SCALe

Codebases

Analyzer

Analyzer

Analyzer

Alerts

Meta-alert audit process

Audit archive with determinations (True, False, etc.)

SCAIFE

Meta-Alert Prioritization

Data Aggregation; Speculative Mapping

Meta-Alert Classification

SCAIFE System v1 & later

From SCALe

1. Create classifier
2. Run classifier on a project
3. Get back classifier results

Improve classifier precision & recall

Data quality

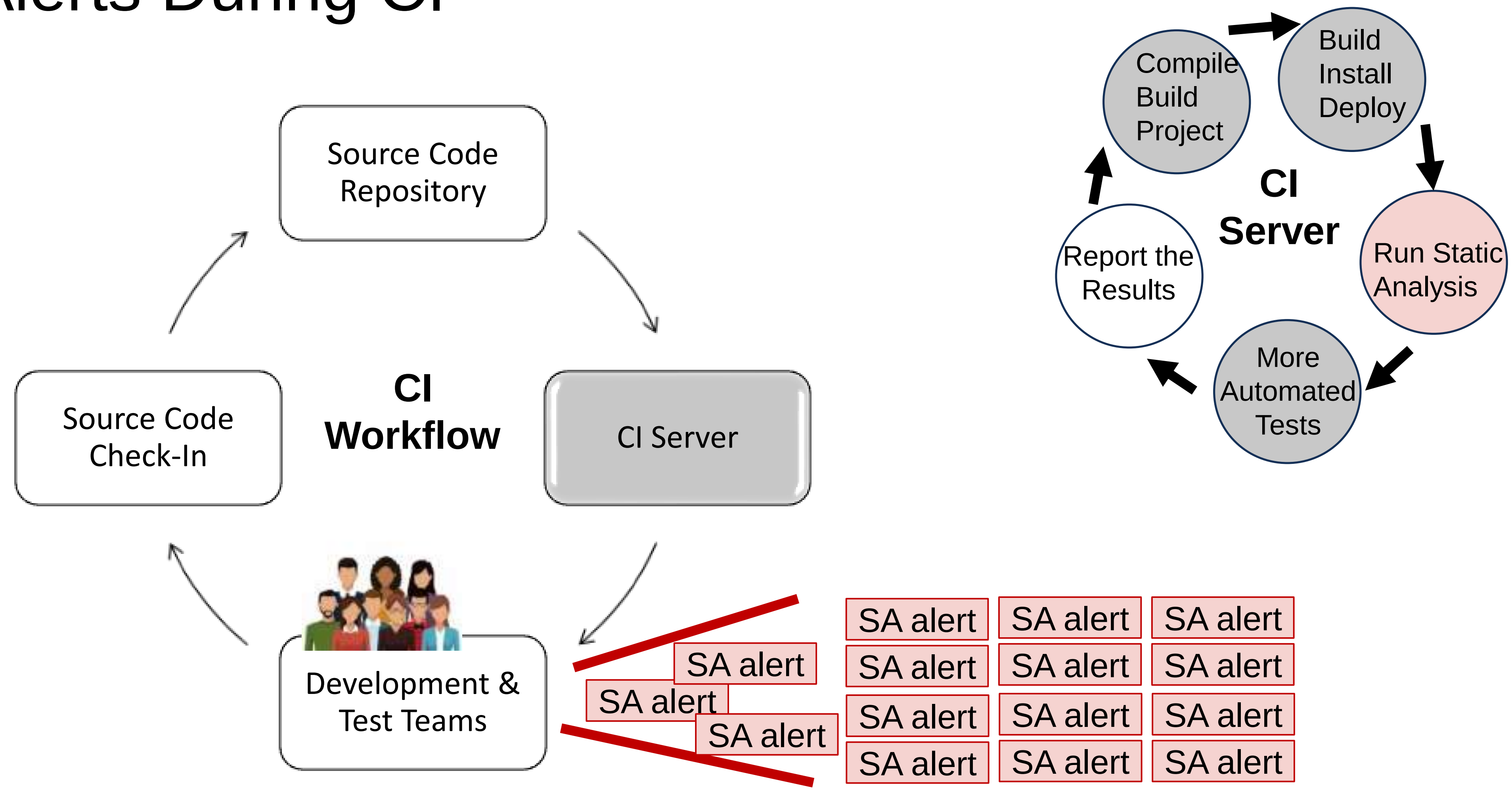
Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

Rapid Adjudication of Static Analysis Alerts During CI



Improve classifier precision & recall

Data quality

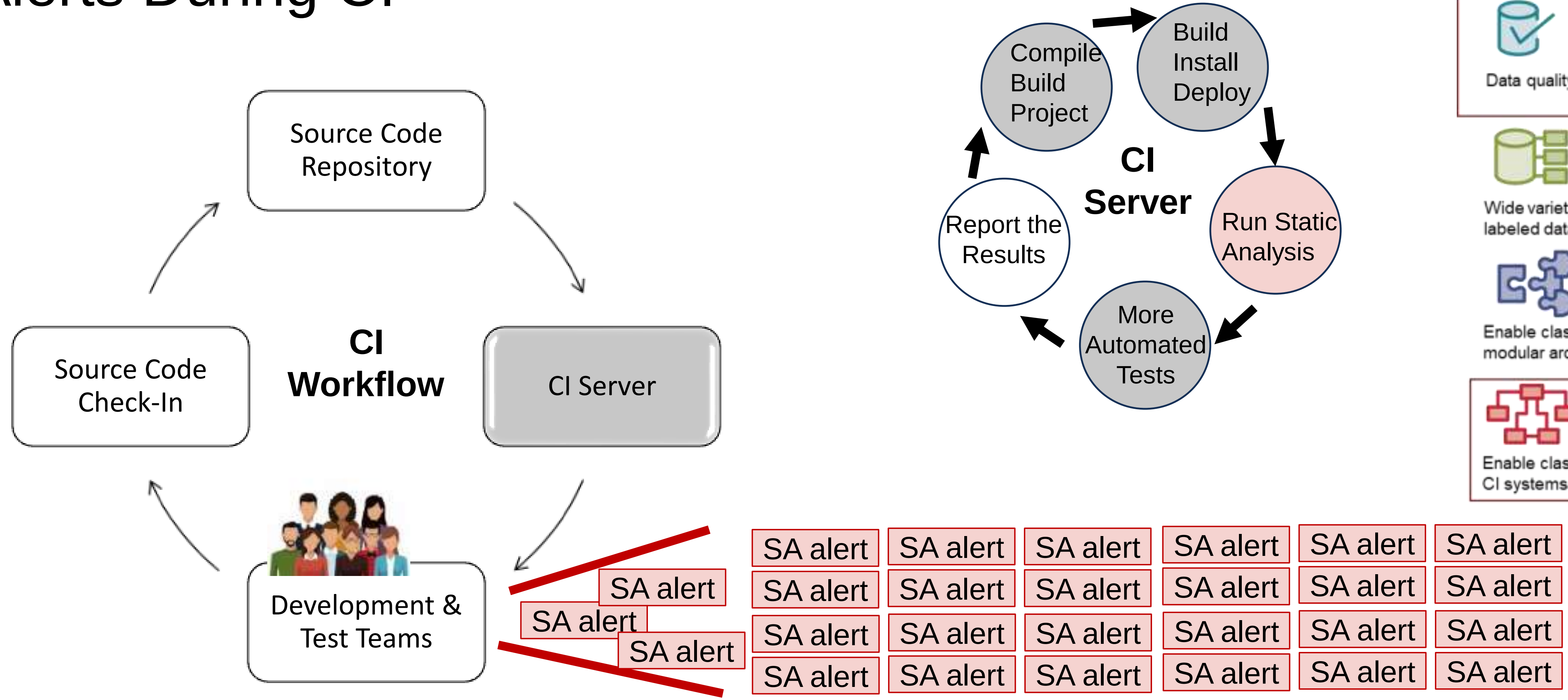
Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

Rapid Adjudication of Static Analysis Alerts During CI



Improve classifier precision & recall

Data quality

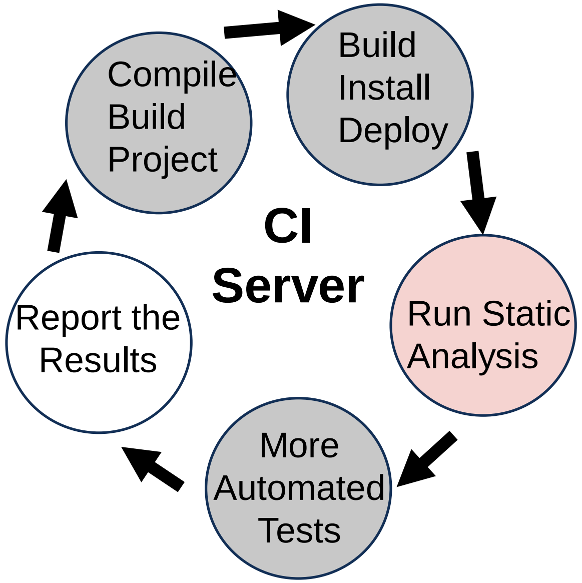
Wide variety of labeled data

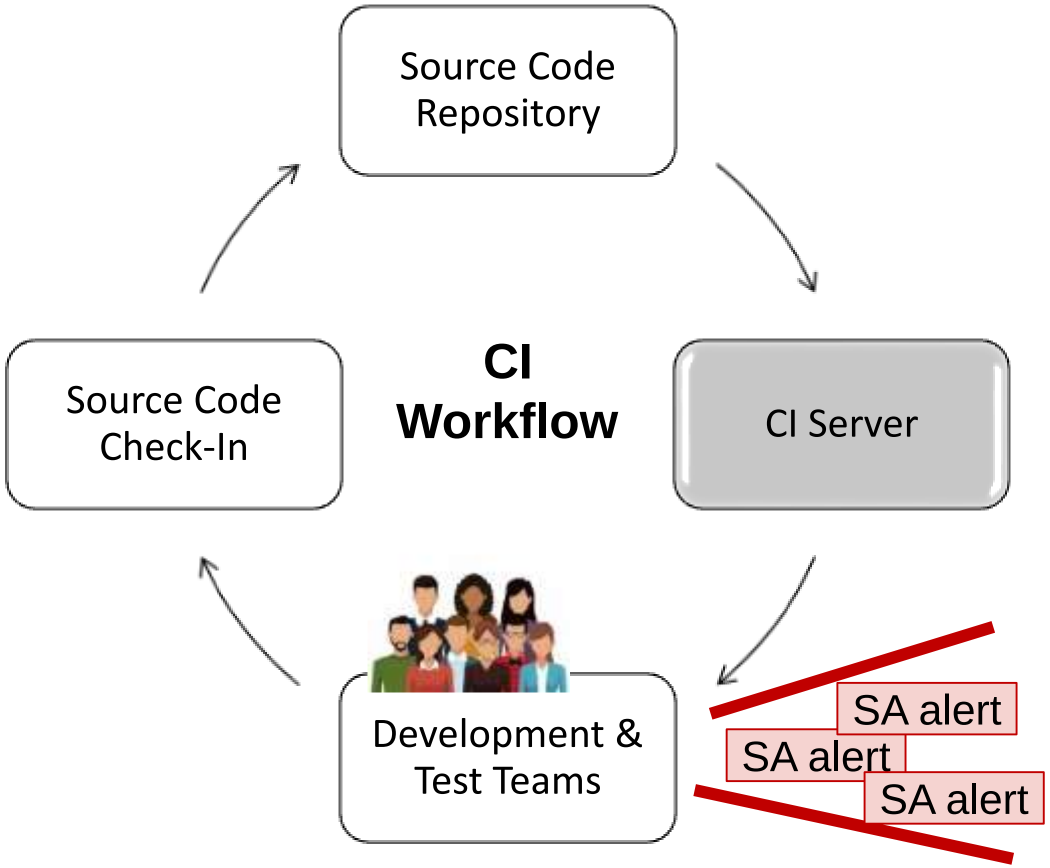
Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

Rapid Adjudication of Static Analysis Alerts During CI





SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert
SA alert

Improve classifier precision & recall

Data quality

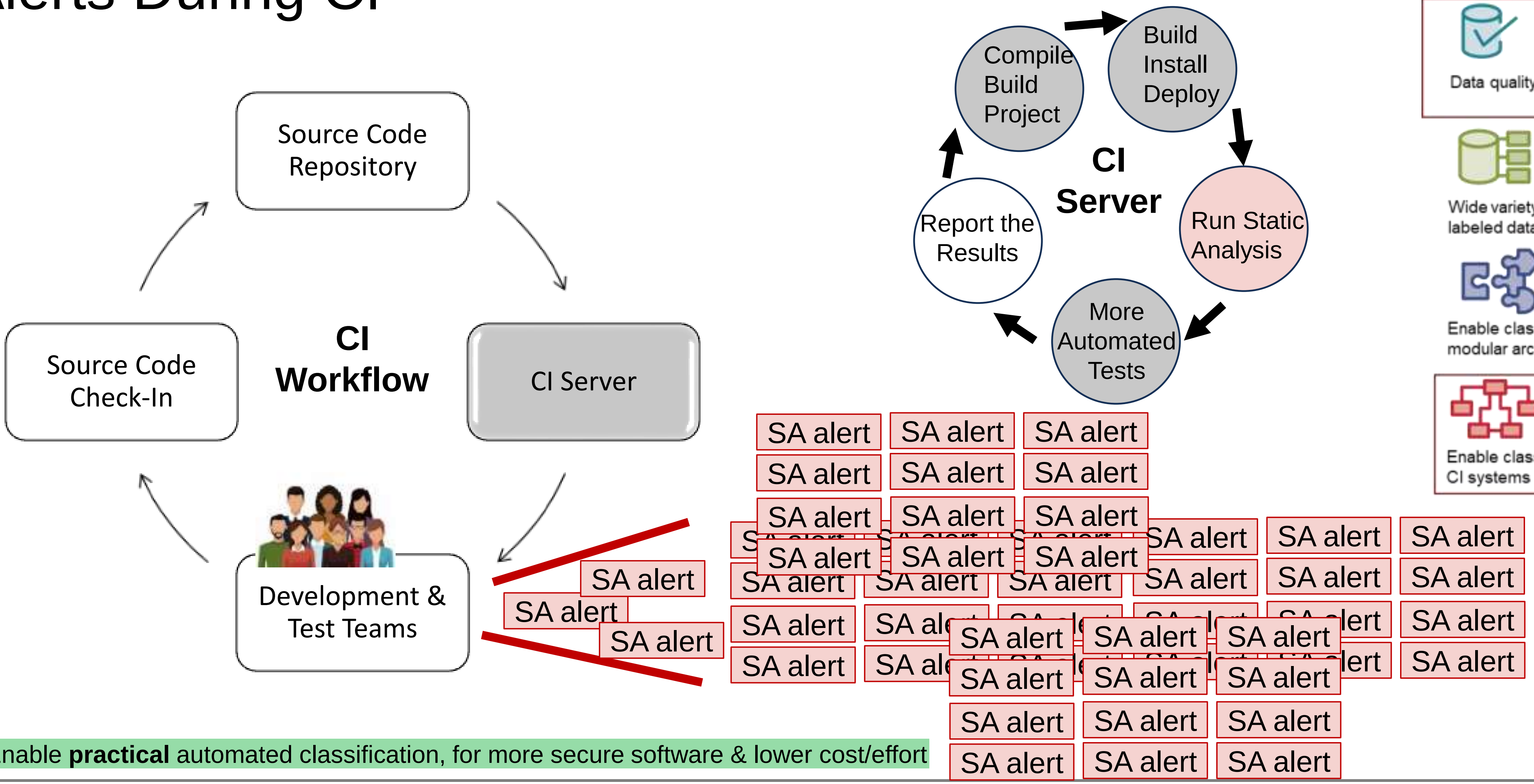
Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

Rapid Adjudication of Static Analysis Alerts During CI



Improve classifier precision & recall

Data quality

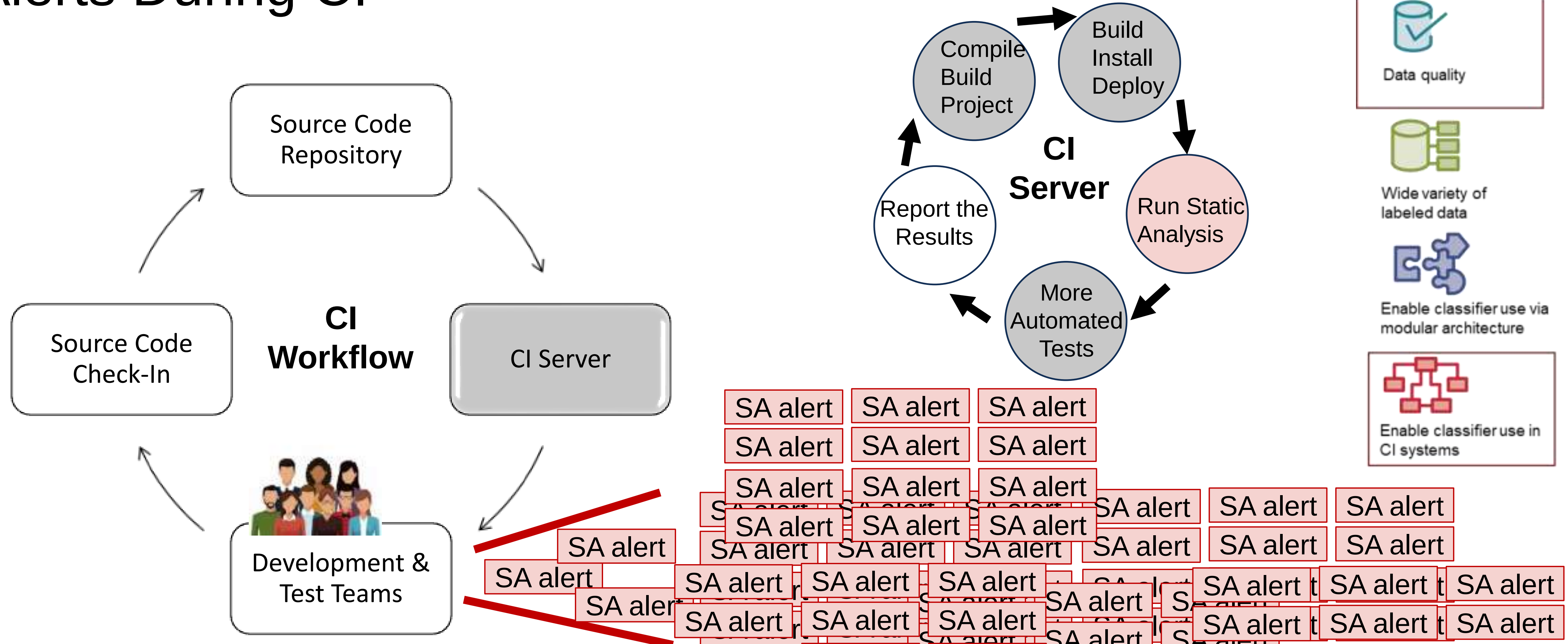
Wide variety of labeled data

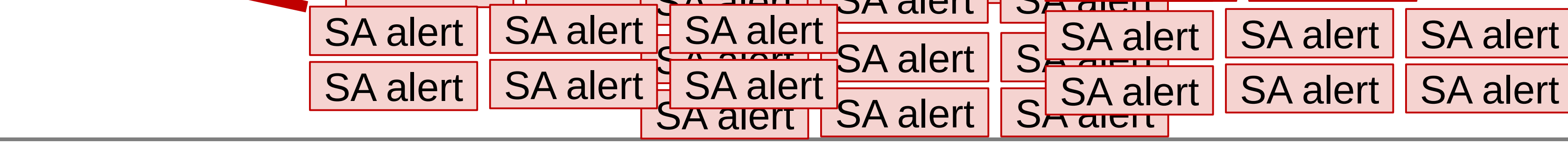
Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

Rapid Adjudication of Static Analysis Alerts During CI





Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

Rapid Adjudication of Static Analysis Alerts During CI

DoD is moving to CI/CD but doesn't have a solution to this problem

Problem: It takes too much time to adjudicate alerts from static analysis tools during continuous integration (CI).

Static analysis (SA) is incompletely integrated in CI development projects in the DoD, and the selection of SA tools is limited to those with very few false positives.

Current practice is too labor-intensive. We will automate it.

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Two Methods of Alternative Incomplete Approaches

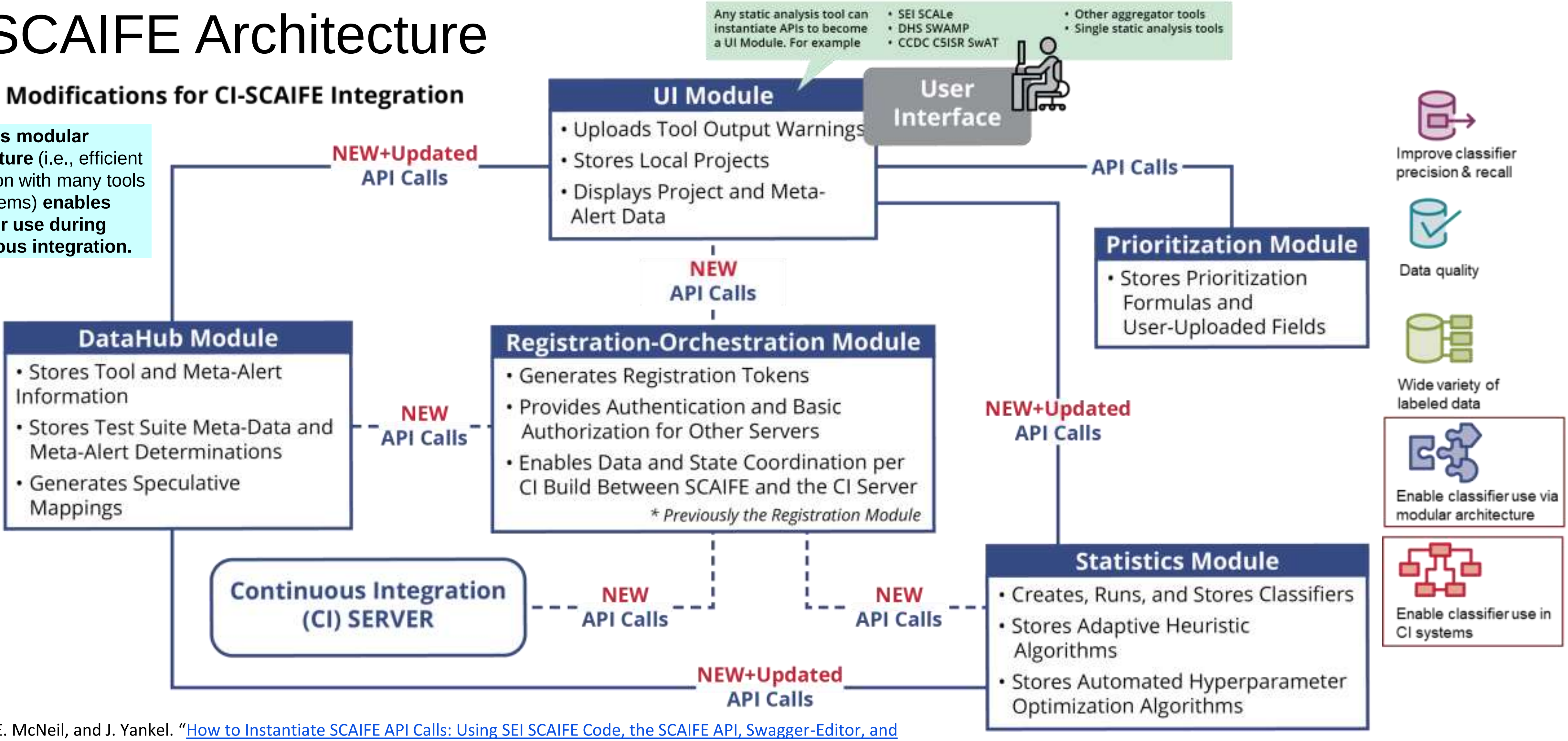
Methods:

1. Adjudicate very few alert types in CI
   - Our method builds on this
2. Run SA automatically in CI but don’t adjudicate during CI

Improve classifier precision & recall
Data quality
Wide variety of labeled data
Enable classifier use via modular architecture
Enable classifier use in CI systems

SCAIFE Architecture

SCAIFE's modular architecture (i.e., efficient integration with many tools and systems) **enables classifier use during continuous integration.**



Modifications for CI-SCAIFE Integration

Any static analysis tool can instantiate APIs to become a UI Module. For example

- SEI SCALe
- DHS SWAMP
- CCDC C5ISR SwAT
- Other aggregator tools
- Single static analysis tools

UI Module

- Uploads Tool Output Warnings
- Stores Local Projects
- Displays Project and Meta-Alert Data

User Interface

Prioritization Module

- Stores Prioritization Formulas and User-Uploaded Fields

DataHub Module

- Stores Tool and Meta-Alert Information
- Stores Test Suite Meta-Data and Meta-Alert Determinations
- Generates Speculative Mappings

Registration-Orchestration Module

- Generates Registration Tokens
- Provides Authentication and Basic Authorization for Other Servers
- Enables Data and State Coordination per CI Build Between SCAIFE and the CI Server

** Previously the Registration Module*

Continuous Integration (CI) SERVER

Statistics Module

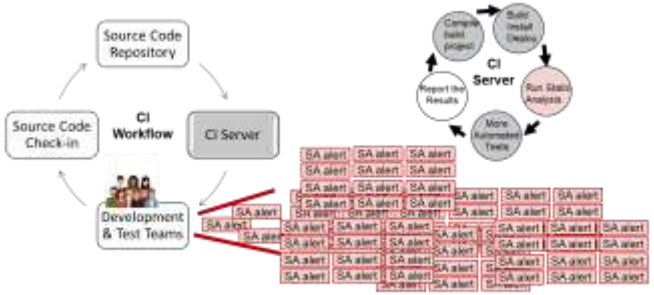
- Creates, Runs, and Stores Classifiers
- Stores Adaptive Heuristic Algorithms
- Stores Automated Hyperparameter Optimization Algorithms

NEW+Updated API Calls
API Calls
NEW API Calls
NEW API Calls
NEW+Updated API Calls
NEW API Calls
NEW API Calls
NEW+Updated API Calls
Improve classifier precision & recall
Data quality
Wide variety of labeled data
Enable classifier use via modular architecture
Enable classifier use in CI systems

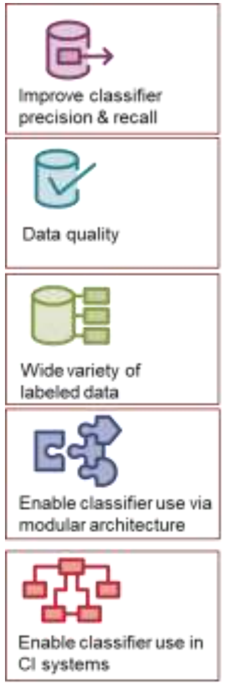
L. Flynn, E. McNeil, and J. Yankel. "How to Instantiate SCAIFE API Calls: Using SEI SCAIFE Code, the SCAIFE API, Swagger-Editor, and Developing Your Tool with Auto-Generated Code." SEI Technical Manual. July 2020.

Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

Integrated CI-SCAIFE Design Highlights



- **SCAIFE-fail or SCAIFE-pass**
  - Fail: If any critical condition meta-alert lacks a cascaded FP *and* classifier confidence FP is less than the threshold.
  - Pass: All other cases
- **The CI build only passes if SCAIFE *and* other tests all pass**
- **Complex design aspects:** tracking all build data through SCAIFE, enabling non-build data to improve the classifier simultaneously, making it all fast
- Project specifies critical build conditions (e.g., CWE-190 and INT31-C)
- Project specifies confidence threshold (e.g., 90%) for classifier predictions
- CI sends build data to SCAIFE
  - Code change commit data and associated tool output
- SCAIFE cascades adjudications
- SCAIFE classifies remaining non-adjudicated meta-alerts


Improve classifier precision & recall
Data quality
Wide variety of labeled data
Enable classifier use via modular architecture
Enable classifier use in CI systems

Meta-alert Classification in CI: Impact

If this project is successful:

- Organizations that develop tools and analyze code
  - Cut number of alerts manually adjudicated in half (save $$s), double adjudicated meta-alerts (more security at same cost), or some mix of cost-savings and increased adjudication
  - By integrating with CI, catch and fix more SA-identified flaws early in development, saving money
  - Use precise cascader developed in this project to improve code security analyses.
  - Use other code and algorithms developed in this project (e.g., SCAIFE system, API, and classification/active learning) to enable practical meta-alert classification in their systems
- Targeted on-ramps for transition:
  - Research project collaborators
  - Discussions started with SEI engineers on DoD contract projects
    - One project could analyze double the SA meta-alerts with the same effort
    - Another project could integrate SA meta-alert adjudication in their CI

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

SA Classification in CI: Relevance/Impact for General DoD State of the Practice

Enable the DoD to more efficiently address SA meta-alerts in CI/CD time constraints, by halving time to manually adjudicate meta-alerts for the same level of security.

Envisioned classifier-use scenario in Authorization to Operate (ATO):

- DoD Program PMO must provide evidence how software risks managed
  - PMO needs ATO by Authorizing Official
  - How to do this for CI/CD systems is pretty much being developed + experimented, now
    - Possibly CATO (Continuous ATO) option
      - ✓ CWEs and other flaw conditions might be required to adjudicate meta-alerts and fix TPs
- **We envision this classifier-use scenario in CATOs**:
  - CATO covers more code flaw conditions
  - Meta-alerts classified expected-False would not require manual adjudication
  - Even if condition not mentioned in a CATO, classifier use frees more adjudication effort

Improve classifier precision & recall

Data quality

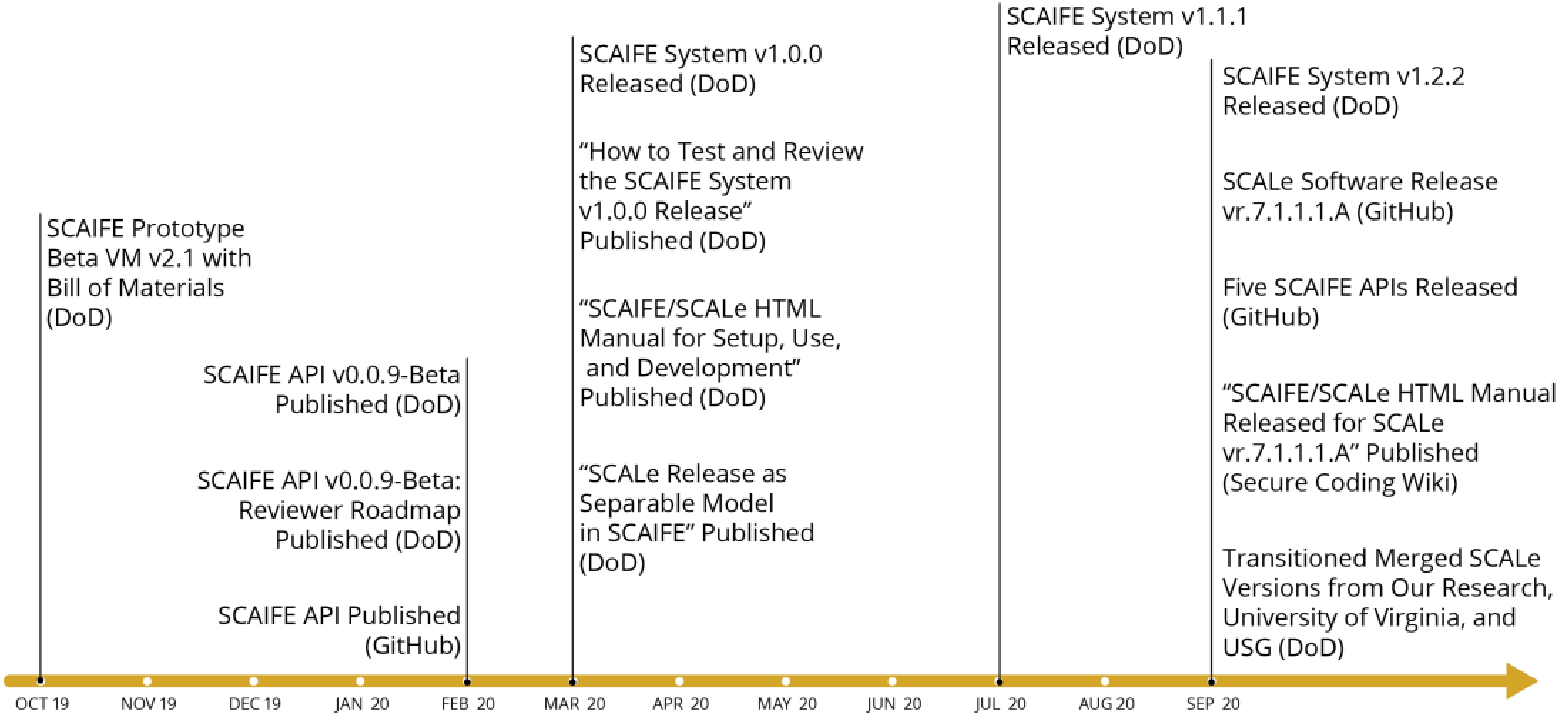
Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

FY20: Select Code/API Artifacts

- DoD can get full implementation
- SCALe + SCAIFE API publicly-published (Sept 2020 versions)
- Significant CI integration; to be completed in FY21



SCAIFE Prototype Beta VM v2.1 with Bill of Materials (DoD)

SCAIFE API v0.0.9-Beta Published (DoD)

SCAIFE API v0.0.9-Beta: Reviewer Roadmap Published (DoD)

SCAIFE API Published (GitHub)

SCAIFE System v1.0.0 Released (DoD)

“How to Test and Review the SCAIFE System v1.0.0 Release” Published (DoD)

“SCAIFE/SCALe HTML Manual for Setup, Use, and Development” Published (DoD)

“SCALe Release as Separable Model in SCAIFE” Published (DoD)

SCAIFE System v1.1.1 Released (DoD)

SCAIFE System v1.2.2 Released (DoD)

SCALe Software Release vr.7.1.1.1.A (GitHub)

Five SCAIFE APIs Released (GitHub)

“SCAIFE/SCALe HTML Manual Released for SCALe vr.7.1.1.1.A” Published (Secure Coding Wiki)

Transitioned Merged SCALe Versions from Our Research, University of Virginia, and USG (DoD)

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

FY20 Select Artifacts (New Detail or Item) –1

- (Oct 2019 and Feb, April, and Sept 2020) GitHub publication of SCAIFE API versions https://github.com/cmu-sei/SCAIFE-API
- (04/2/20) Published the open dataset "RC_Data" for classifier research to the SEI CERT Secure Coding webpage "Open Dataset RC_Data for Classifier Research". Database with static analysis alerts from open-source tools, adjudications, code metrics, and more for two codebases.
- (06/18/20) Presentation "Automated Classifiers to Adjudicate Static Analysis Alerts: Challenges, Progress, and Next Steps" (Lori Flynn, Stephen Adams, and Tim Sherburne) to DoD's DEVCOM Cyber Community of Interest.
- (06/25/20) Presentation "Automated Classifiers to Adjudicate Static Analysis Alerts: Challenges, Progress, and Potential Collaborations with NASA IV&V" (L. Flynn) to leaders of the NASA IV&V Static Code Analysis Working Group (SCAWG).

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

FY20 Select Artifacts (New Detail or Item) –2

- (07/8/20) Technical manual "How to Instantiate SCAIFE API Calls: Using SEI SCAIFE Code, the SCAIFE API, Swagger-Editor, and Developing Your Tool with Auto-Generated Code" (L. Flynn, E. McNeil, and J. Yankel) Instructions for three types of SCAIFE System code access: (1) none, (2) access to SCALe code, or (3) full access.
- (07/13/20) Auto-generated Java client code for the five SCAIFE API modules for a DoD collaborator, to help them quickly start to instantiate SCAIFE API calls from their tool
- (09/14/2020) Blog post "Managing Static Analysis Alerts with Efficient Instantiation of the SCAIFE API into Code and an Automatically Classifying System" by Lori Flynn
- (09/22/2020) Presentation "Rapid Adjudication of Static Analysis Meta-Alerts During Continuous Integration", Software Assurance Community of Practice (SwA CoP).
- (Sept. 2020) SCALe code at https://github.com/cmu-sei/SCALe/tree/scaife-scale
- (Sept. 2020) Test data generated with 'diff' cascading, for comparison to precise cascading

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

SCAIFE and Static Analysis Classification Research

Invitation to Collaborate

DoD Orgs that do CI Development: Invitation to Test

I need DoD collaborators that do CI development, to test our tooling

- Current collaborators test but not doing CI
- Full system implementation release currently limited to DoD
- CI testing does *not* have to include data sharing (next slide)
- If interested please contact me lflynn@cert.org

Deployment and testing supported by project

- release system containerized and with configuration files (ports, URLs, names) to ease integration in wide variety of systems
- comes with much documentation, we’ve extended that a lot in last year per collaborator feedback
- Part of FY21 project specifically is for helping collaborators use the system

All: Might you be able to help us get labeled data?

- Effort to label data on particular open-source codebases
- SCALe (scaife-scale branch) on GitHub can be used to do the adjudication and store results
- Even better, SEI can provide full SCAIFE system to DoD orgs (includes SCALe + classification etc.)
- Auditing self-training support via published materials (next slide)
- Possibly your own stored archives, sanitized before sharing

High-quality manually labeled data would help us improve our DoD sponsored classification research.

If our research succeeds, the improved classification techniques and data will help your orgs to secure your code and save money.

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

Self-Training Resources for Auditing Meta-Alerts

- Paper “Static Analysis Alert Audits: Lexicon & Rules” (D. Svoboda, L. Flynn, W. Snavely) IEEE SecDev
- Presentation “Hands-On Tutorial: Auditing Static Analysis Alerts Using a Lexicon and Rules” (L. Flynn, D. Svoboda, W. Snavely) https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=505451
- Webcast (1 hour video, hands-on SCALe use): “Improve Your Static Analysis Audits Using CERT SCALe’s New Features” by L. Flynn. (The SCAIFE System includes the SCALe tool, as a separable part of SCAIFE.) https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=538843 (video) and https://resources.sei.cmu.edu/asset_files/Presentation/2018_017_101_532198.pdf (slides)
- Video “Rapid Construction of Accurate Automatic Alert Handling System” Nov. 2019 https://youtu.be/dwYbhgko3to
- Slides “Rapid Construction of Accurate Automatic Alert Handling System” Nov. 2019 https://resources.sei.cmu.edu/asset_files/Presentation/2019_017_001_635435.pdf

It will increase the quality of data if your team studies definitions of the code flaw types ("conditions") they will inspect static analysis meta-alerts for, as defined in a formal code flaw taxonomy.

For this classification research, the taxonomies currently of the most interest are:
- MITRE CWE https://cwe.mitre.org/data/index.html
- CERT coding rules for C: https://wiki.sei.cmu.edu/confluence/display/c/SEI+CERT+C+Coding+Standard
- CERT coding rules for Java: https://wiki.sei.cmu.edu/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java
- CERT coding rules for C++: https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682

The SCALe (scaife-scale branch) GitHub release includes a SCAIFE/SCALe HTML manual with extensive information about how to use the SCAIFE and SCALe systems to adjudicate (aka ‘audit) static analysis meta-alerts.

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

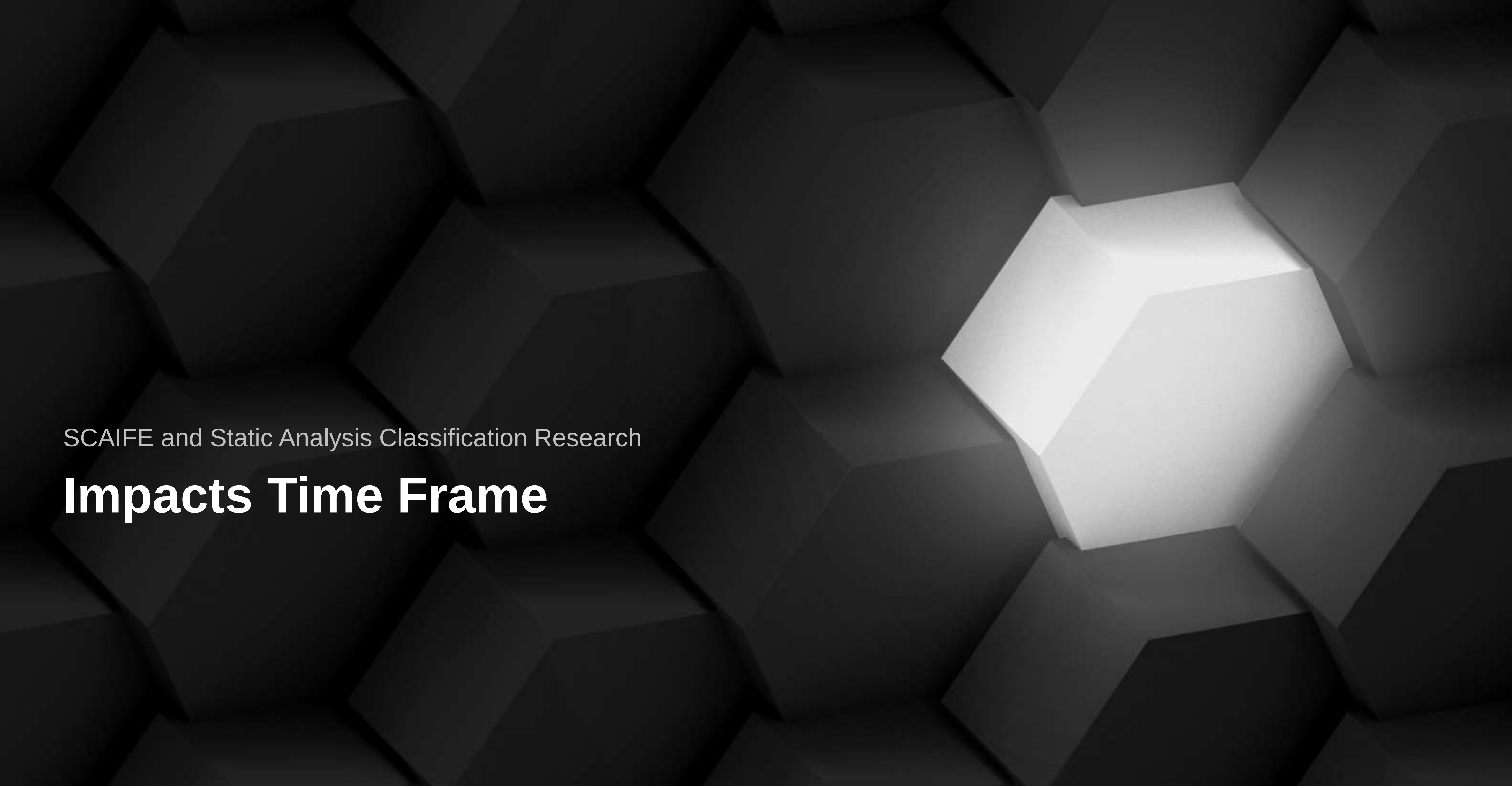
Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems



SCAIFE and Static Analysis Classification Research

Impacts Time Frame

Project Impacts Time Frame

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

NEAR

Public can use/review SCAIFE API and SCALe* module.

DoD collaborators will further test SCAIFE to

- provide data and feedback
- integrate their tools using the API

The FY20-21 research project incorporates continuous integration (CI) into architecture design.

MID

More collaborators (DoD and non-DoD) to test SCAIFE with CI.

Design improvements for transition include

- classification precision
- latencies
- bandwidth/disk/memory use
- business continuity
- scalability

FAR

A wide variety of systems will do automated meta-alert classification, using

- SCAIFE System
- SCAIFE API

Goal: Provide better software security, or less time and cost for the same security (DoD and non-DoD).

** Version of SCALe used in SCAIFE System implementation*

Goal: Enable **practical** automated classification, for more secure software & lower cost/effort

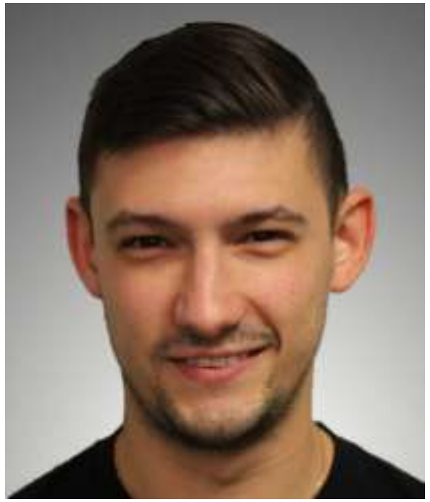
FY20 Project Team

Dr. Lori Flynn

Ebonie McNeil

David Svoboda

Matt Sisk



Hasan Yasar

Joseph Yankel

Shane Ficorilli

David Shepard

Thanks + Contact Info

Thank you for listening!

Questions?

Feedback and potential collaborations are welcome, here's my contact info:

Lori Flynn, PhD

Senior Software Security Researcher

lflynn@sei.cmu.edu

Carnegie Mellon University

Software Engineering Institute

4500 Fifth Avenue

Pittsburgh, PA 15213-2612