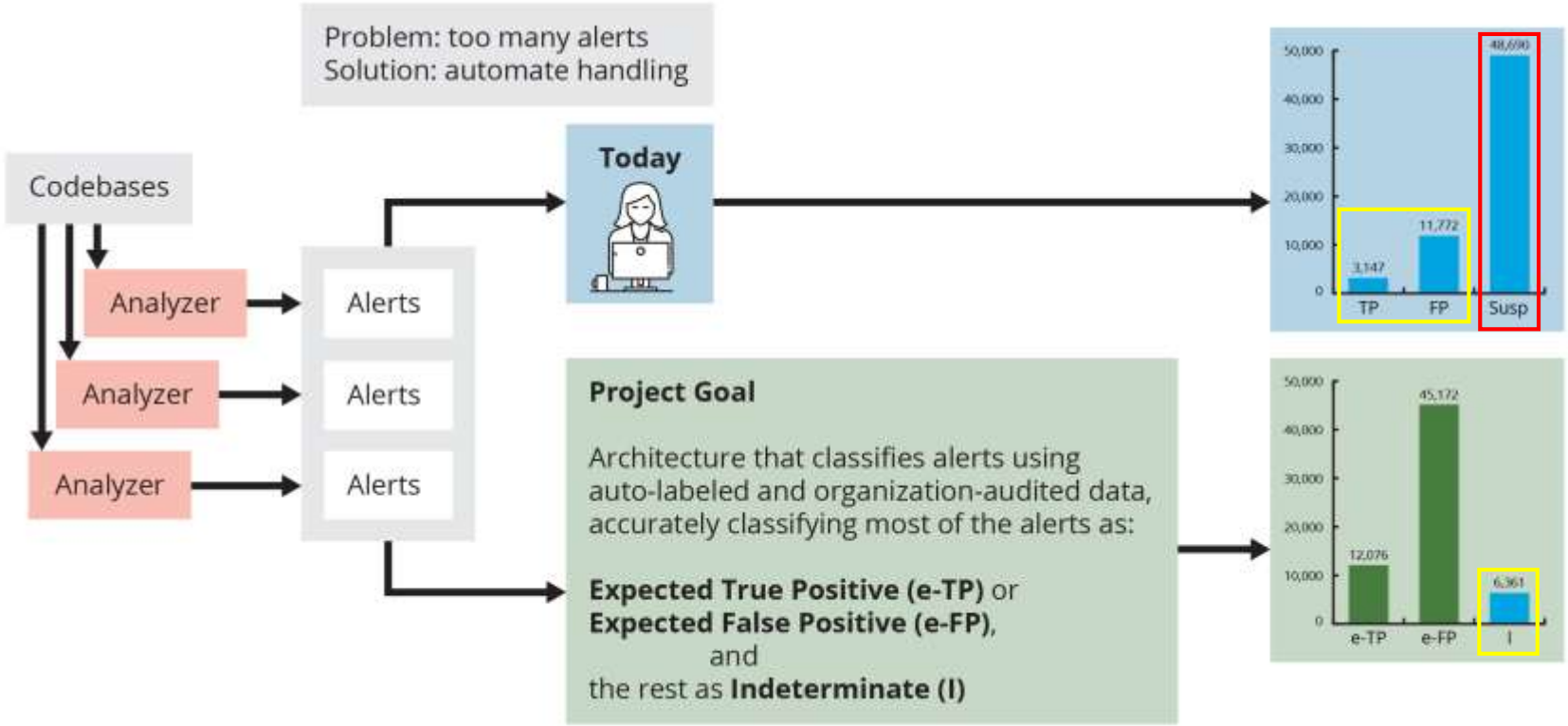
Rapid Construction of an Accurate Automatic Alert-Handling System: Research Progress Review

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM18-1206

Overview



FY16-18 Static Analysis Alert Classification Research

FY16

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

FY17

- Issue addressed: **too little labeled data** for accurate classifiers for some conditions (CWEs, coding rules)
- Novel approach: **use test suites to automate production of labeled** (True/False) **alert archives for many conditions**
- Result: high accuracy for more conditions

FY18

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

Code

API definition (swagger) and code development

SCALe v2.1.3.0 static analysis alert auditing tool

- New features for prioritization and classification
  - Fused alerts, CWEs, new determinations (etc.) for collaborators to generate data
- Released to collaborators Dec. 2017–Feb. 2018
- GitHub publication Aug. 2018

First public SCALe release (2.1.4)

SCALe v3.0.0.0 released Aug. 2018 to collaborators

Develop and test classifiers. Novel work includes

- enabling cross-taxonomy test suite classifiers (using precise mappings)
- enabling “speculative mappings” for tools (e.g., GCC)

Non-code Publications & Papers FY18

For collaborators, others to implement API calls or use new SCALe

Architecture API definition and new SCALe features

- Special Report: “Integration of Automated Static Analysis Alert Classification and Prioritization with Auditing Tools” (Aug. 2018)
  - Technical Report: public version (Oct. 2018)
- SEI blog post: “SCALe: A Tool for Managing Output from Static Code Analyzers” (Sep. 2018)

Explain research methods and results

Classifier development research methods and results

- Paper “Prioritizing Alerts from Multiple Static Analysis Tools, using Classification Models,” SQUADE (ICSE workshop)
- SEI blog post: “Test Suites as a Source of Training Data for Static Analysis Alert Classifiers” (Apr. 2018)
- SEI Podcast (video): “Static Analysis Alert Classification with Test Suites” (Sep. 2018)
- In-progress conference papers (4): precise mapping, architecture for rapid alert classification, test suites for classifier training data, API development

Static analysis tool developers can automatically test for CERT rule coverage (some rules)

Precise mappings on CERT C Standard wiki

- CERT manifest for Juliet (created to test CWEs) to test CERT rule coverage
- Per-rule precise CWE mapping [1] [2]

For code flaws you care about, understand your tool coverage

Analysis of Juliet Test Suite: Initial CWE Results

Alert Type	Labeled Fused Alerts (counts a fused alert once)
TRUE	**13,330**
FALSE	**24,523**

Lots of new data for creating classifiers

(37,853 labeled alerts)

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

- First 37,853 alert audits wouldn't cover many conditions (and sub-conditions) covered by the Juliet test suite!
- Need true and false labels for classifiers.
- **Realistically:** enormous amount of manual auditing time to develop that much data.

These are initial metrics (more data as we use more tools and test suites).

*Nathaniel Ayewah and William Pugh, "The Google FindBugs Fixit," Proceedings of the 19th International Symposium on Software Testing and Analysis, ACM, 2010.

Juliet Test Suite Classifiers: Initial Results (Hold-out Data)

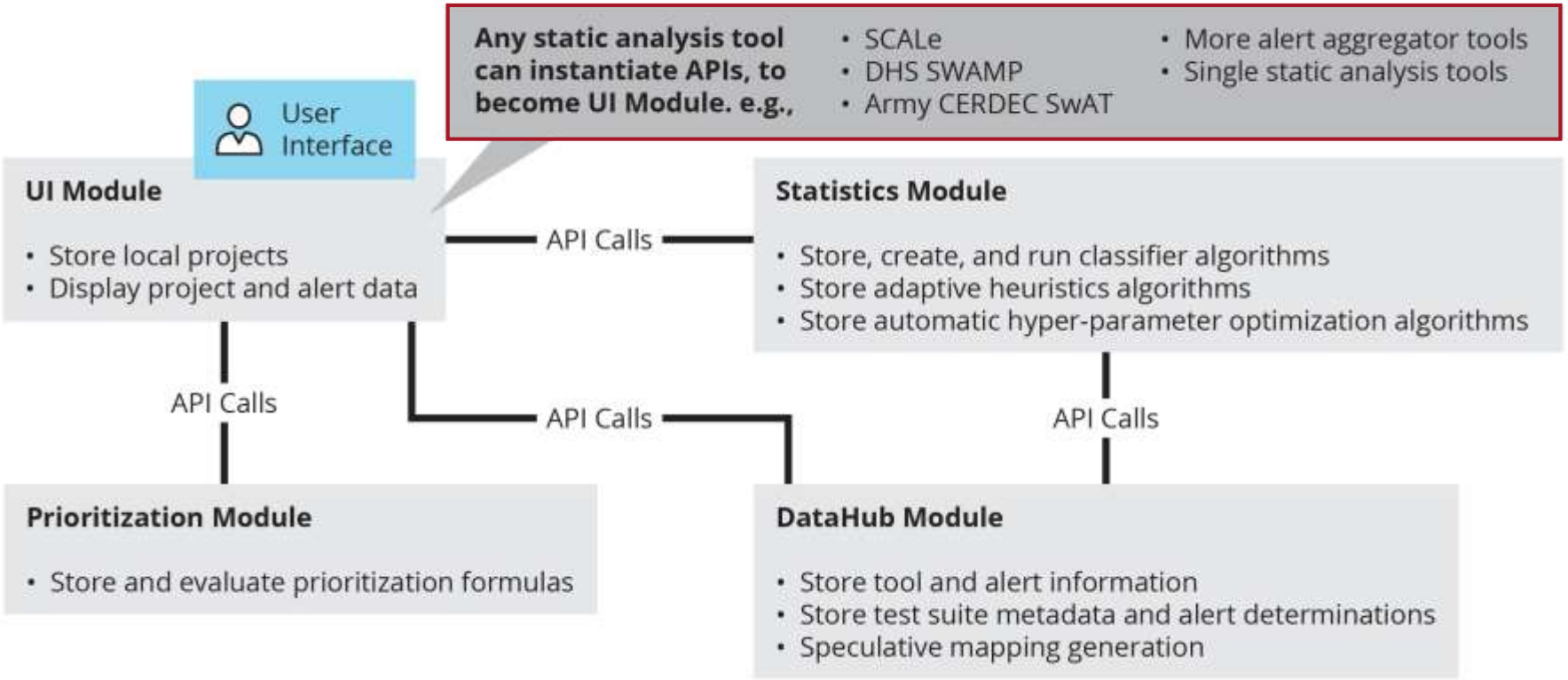
Classifier	Accuracy	Precision	Recall
rf	0.938	0.893	0.875
lightgbm	0.942	0.902	0.882
xgboost	0.932	0.941	0.798
lasso	0.925	0.886	0.831

		Actual condition	
	Total population	Condition true	Condition false
Predicted condition	Predicted condition true	True positive	False positive
	Predicted condition false	False negative	True negative
		True positive rate, recall, sensitivity = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ (Condition true)}}$	False positive rate = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ (Condition false)}}$

Accuracy = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$

Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition true}}$

Architecture



Architecture Development

Representational State Transfer (REST)

- Architectural style that defines a set of constraints and properties based on HTTP
- RESTful web services provide interoperability between systems
- Client-server

We chose to develop a RESTful API.

- Swagger/OpenAPI open-source development toolset
  - Develop APIs
  - Auto-generate code for server stubs and clients
  - Test server controllers with GUI
  - Wide use (10,000 downloads/day)

SCALe Development for Architecture Integration

SCALe will make UI Module API calls in prototype system.

- Other alert auditing tools (e.g., DHS SWAMP) also can instantiate UI Module API.

Continue FY19: Classifier Research and Development

Using test suite data for classifiers, research:

- Adaptive heuristics:
  - How classifiers incorporate new data
  - Test suite vs. non-test-suite data
  - Weighting recent data
- Semantic features for cross-project prediction
  - Test suites as different projects

FY19 Next Steps

Collaborator API implementation

More collaborator audit archive data sharing

Metrics of success:

- Compare classifier precision on DoD datasets (cross-validation on test set):
  - Test with semantic features
  - Variations of adaptive heuristics
- Test fault detection rates by tracking true positives detected versus number of manual alert inspections
- Goal: minimum 60% classified e-TP or e-FP with 95% accuracy against collaborator data
- Test architecture generality using varied plug-ins to API