Rapid Construction of Accurate Automatic Alert Handling System: Architecture and Prototype

Problem

Static analysis alerts for security-related code flaws require too much manual effort to triage, and **there is little use of automated alert classifier technology because of barriers of cost, expertise, and lack of labeled data.**

Solution

Develop extensible architecture for classification and advanced prioritization, building on novel test-suite data method we developed.

- Implement prototype
- Enable organizations to quickly start using classifiers and advanced prioritization by making API calls from their alert auditing tools
- Develop adaptive heuristics for classifier to adapt as it learns from test suite and "natural program" data

Approach

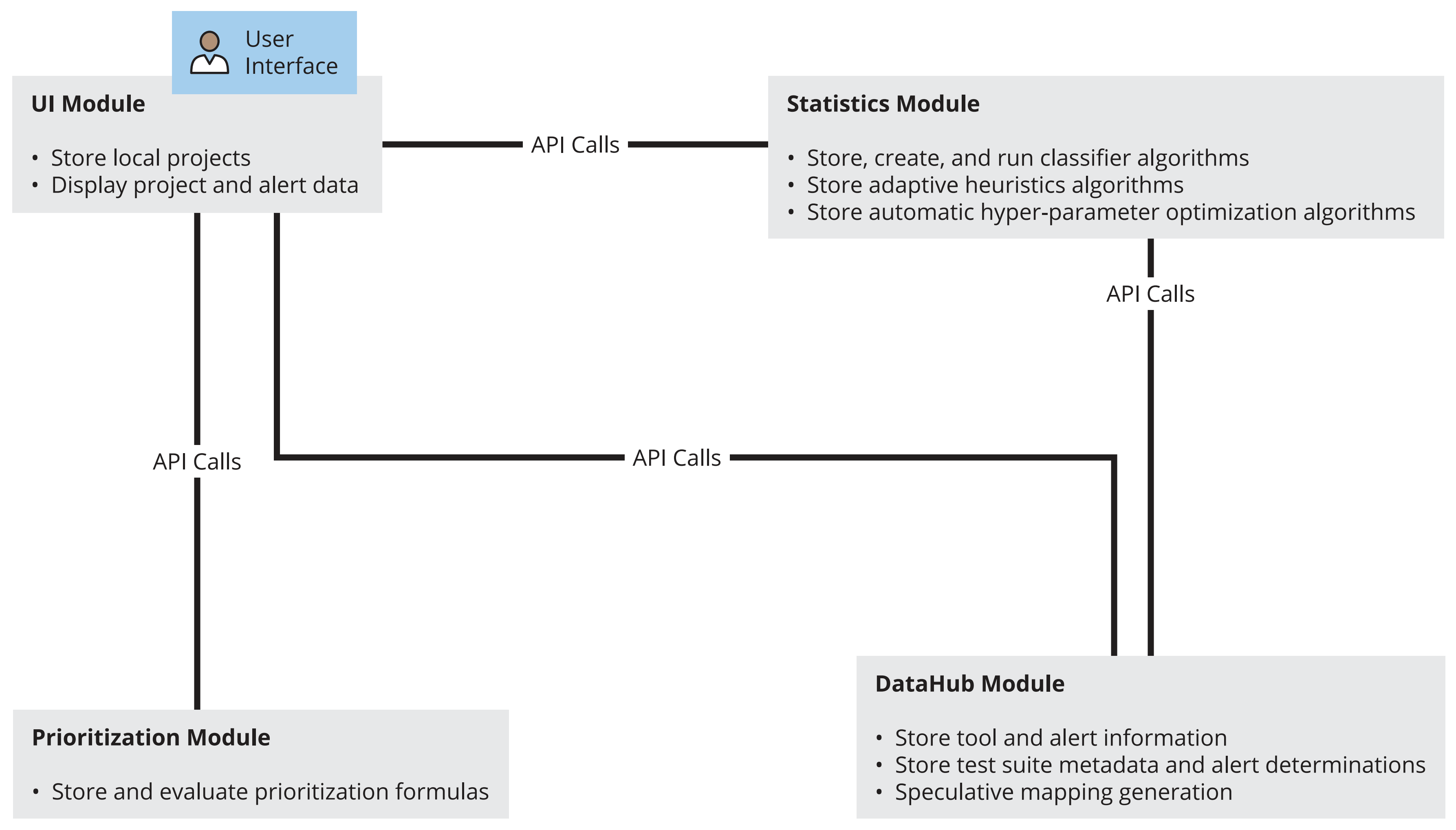
1. Design architecture
2. Develop API definition
3. Implement prototype system
4. Develop adaptive heuristics
5. Test adaptive heuristics with datasets combining test suite and real-world (DoD) data
6. Collaborators test architecture and prototype

Juliet test suite classifiers: initial results (hold-out data)

All four classification methods had high accuracy.

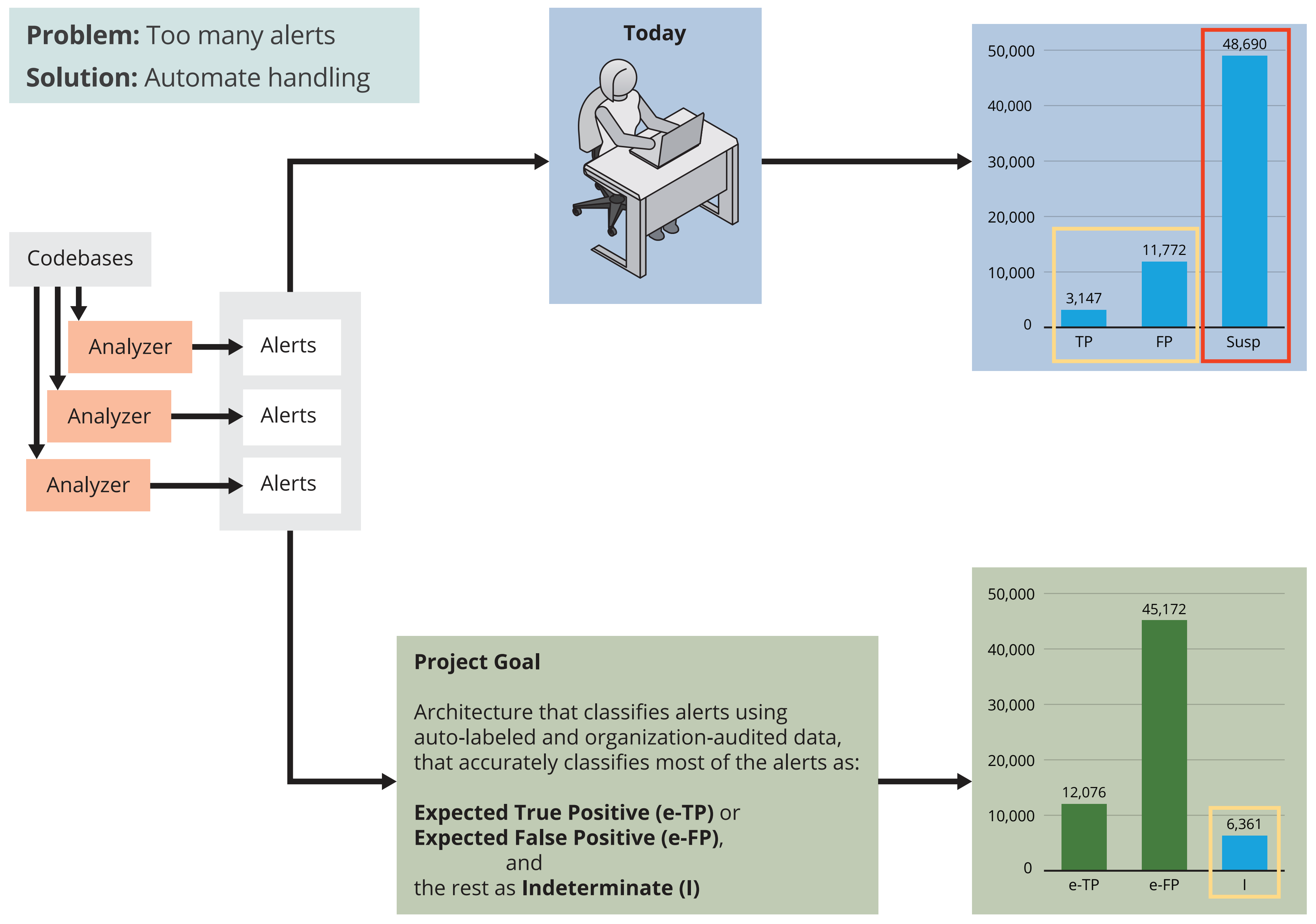
CLASSIFIER	ACCURACY	PRECISION	RECALL	AUROC
Random Forest	0.938	0.893	0.875	0.991
Lightgbm	0.942	0.902	0.882	0.992
Xgboost	0.932	0.941	0.798	0.987
Lasso	0.925	0.886	0.831	0.985

Architecture


User Interface
UI Module
• Store local projects
• Display project and alert data
API Calls
Statistics Module
• Store, create, and run classifier algorithms
• Store adaptive heuristics algorithms
• Store automatic hyper-parameter optimization algorithms
API Calls
API Calls
API Calls
Prioritization Module
• Store and evaluate prioritization formulas
DataHub Module
• Store tool and alert information
• Store test suite metadata and alert determinations
• Speculative mapping generation

Problem and Goal

Problem: Too many alerts
Solution: Automate handling



Project Goal

Architecture that classifies alerts using auto-labeled and organization-audited data, that accurately classifies most of the alerts as:

Expected True Positive (e-TP) or **Expected False Positive (e-FP)**, and the rest as **Indeterminate (I)**

Artifacts

Code and Test Results

- API definition (swagger, RESTful)
- SCALe v2 static analysis alert auditing tool with new features required for collaborators to generate data (also published on GitHub)
- SCALe v3 released Aug. 2018 (collaborators-only) with advanced prioritization schemes and features for classification
- Code development for prototype system
- Expanded archive of auto-labeled alerts
- Test results from cross-taxonomy test suite classifiers using precise mappings
- Code enabling novel "speculative mapping" method for tools without mappings to test suite metadata's code flaw taxonomy
- Adaptive heuristic development and testing results (in progress)

Non-Code Publications + Papers

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 (Sep. 2018)
- SEI blog post: "SCALe: A Tool for Managing Output from Static Code Analyzers" (Sep. 2018)

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

Precise mappings on CERT C Standard wiki
- Metadata for Juliet (created to test CWEs) to test CERT rule coverage
- Per-rule precise CWE mapping

Continuing in FY19

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

This project developed an architecture and API definition for static analysis alert classification and advanced alert prioritization, plus major parts of a prototype system.

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

Carnegie Mellon University
Software Engineering Institute

Lori Flynn| lflynn@sei.cmu.edu