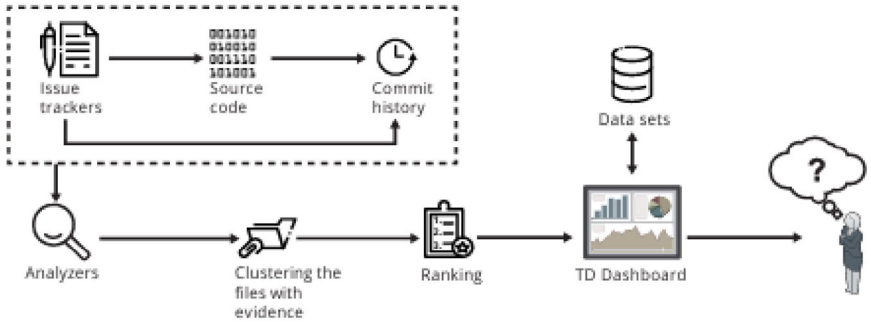
Data-Driven Technical Debt Analysis

Technical debt conceptualizes the tradeoff between the short-term benefits of rapid delivery and the long-term value of developing a software system that is easy to evolve, modify, repair, and sustain. In this work, we extended and developed tools that integrate data from multiple commonly available sources to pinpoint problematic design decisions and quantify their consequences in a repeatable and reliable way for uncovering technical debt. Our results provide a set of analysis approaches for developers' and software managers' technical debt management tool box.

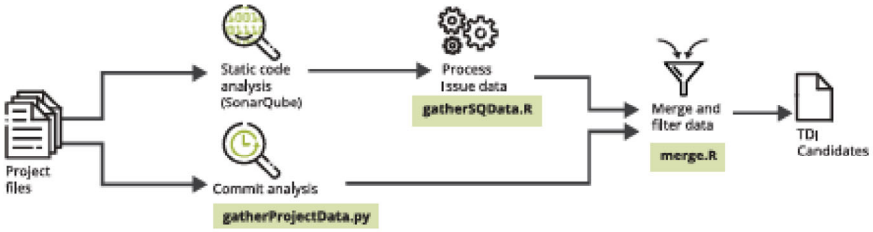
Our outcomes include the following:

- Classifier using gradient-boosting machines that assists in estimating the tickets that contain technical debt discussions by developers
- Design violation view from code quality rules, reducing the space of technical debt investigation by 95%
- Pipeline to extract candidate technical debt items, ranked by amount of evidence, which teams can use for assessing and scoping their technical debt

Technical Debt Analytics

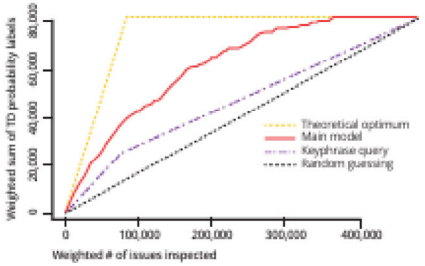


Analysis Pipeline to Generate Candidate Technical Debt Items



Technical Debt Classifier Performance

- First of its kind active-learning pipeline, which resulted in 1,934 labeled technical debt examples
- Feature engineering to combine discussion length, *n*-grams, key phrases, concepts, and document context with guidelines for key phrases that signal technical debt



	Accuracy*	Precision*	Recall*	AUROC*
no TD	0.90	NA	0.00	0.50
keyphrase query	0.83	0.26	0.35	0.62
main model	0.87	0.40	0.62	0.88

Analysis Pipeline to Generate Candidate Technical Debt Items

Design problems, frequently the result of optimizing for delivery speed, are a critical part of long-term software costs. We developed an algorithm and analysis pipeline that maps existing static analysis rules to design issues. The goal of the analysis is to help teams focus their attention in areas of the codebase causing extra work earlier in the lifecycle.

Sample Technical Debt Items

TDI candidate	Design paradigm technical debt issue
DFSA***.java	Logging should be centralized to avoid security and data management issues
DFS***s.java	Credentials and IP addresses should not be hard coded
	Deprecated code should be removed
FS*****.java	Redundant exceptions propagate errors and create vulnerabilities and resource management issues
-F****m.java	
-F****ry.java	Connected files propagate issues
-B****.java	
-F*****p.java	

The SEI team has been a pioneer in advancing the practices and research agenda in managing technical debt. The experiences of the team will culminate in a practitioner book, scheduled to be published in early 2019 by Addison-Wesley. You can find all tools and resources at **sei.cmu.edu/go/technicaldebt**.