A Large-Scale Empirical Study on Self-Admitted Technical Debt

Cédric Noiseux, MSc in Computer Engineering
Giuliano Antoniol, PhD Ing

Polytechnique Montréal
November 24th 2016

Presentation Plan

1. Introduction
2. Empirical Study
    1. Context Selection
    2. Data Extraction
    3. Data Analysis
3. Study Results
    1. RQ1
    2. RQ2
    3. RQ3
4. Threats To Validity
5. Future Work
6. Conclusion

Introduction

Definition : *Technical debt instances intentionally introduced by developers and explicitly documented in code comments*

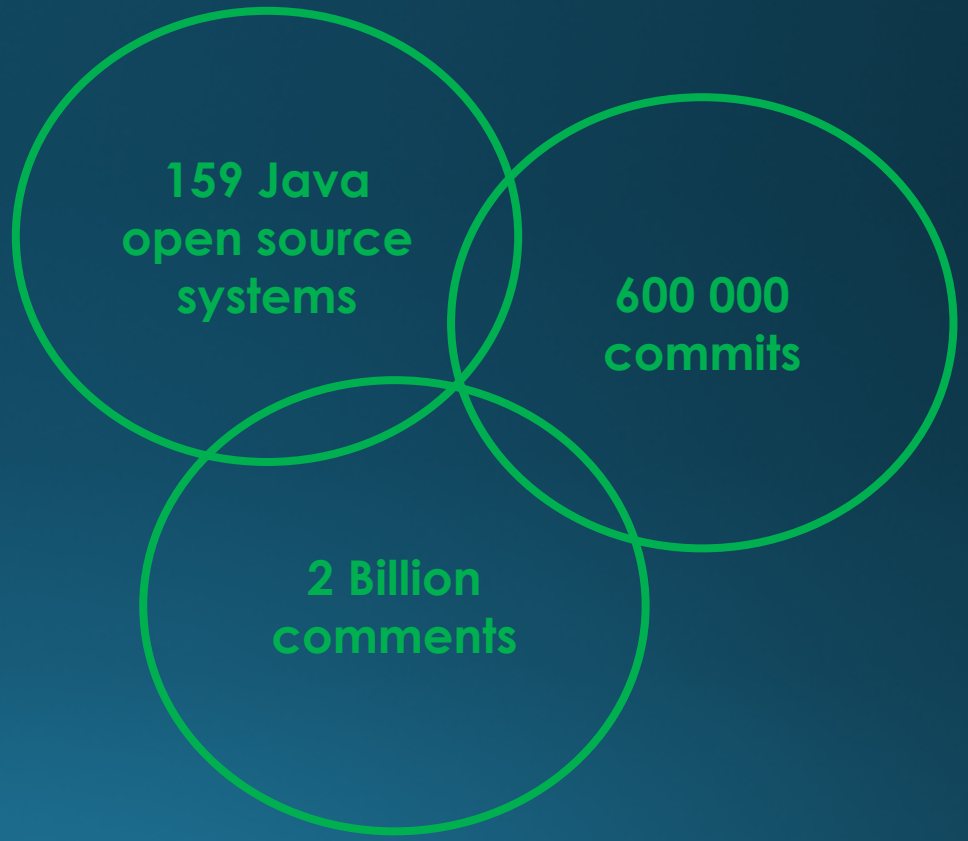
How : Mining code comments

Contribution : Large-Scale empirical study

RQ1 SATD Diffusion
SATD Types

RQ2 SATD Evolution and Survivability
Actors

RQ3 Low Quality <-> SATD



Empirical Study (1/3)
Context Selection

Limitations: Java projects and Active projects

Apache and Eclipse ecosystems

- Different sizes
- Different architectures
- Different development bases

Table 1: Characteristics of ecosystems under analysis.

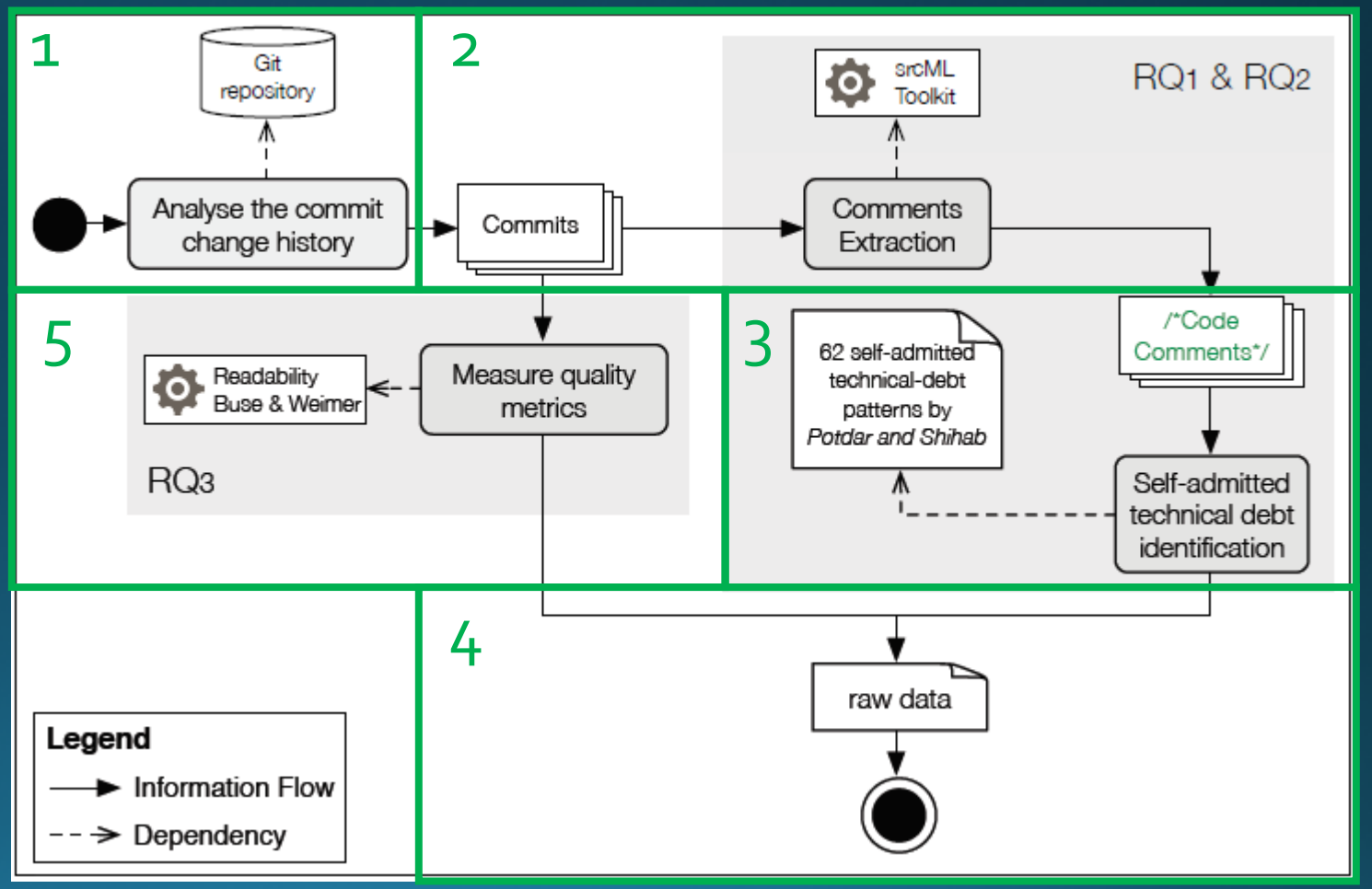
Ecosystem	#Projects	KLOC				#Commits				#Contributors			
		min	max	mean	overall	min	max	mean	overall	min	max	mean	overall
Apache	120	1	1,550	272	32,692	99	22,433	4,332	519,937	2	463	54	6,426
Eclipse	39	1	1,105	246	9,613	39	26,628	3,540	138,089	5	304	40	1,559
Overall	**159**	**1**	**1,550**	**266**	**42,305**	**39**	**26,628**	**4,138**	**658,026**	**2**	**463**	**50**	**7,985**

Bavota, Gabriele, and Barbara Russo. "A large-scale empirical study on self-admitted technical debt." *Proceedings of the 13th International Workshop on Mining Software Repositories*. ACM, 2016.

Empirical Study (2/3)
Data Extraction

1. Git Repositories Cloning
2. Comments Extraction
3. SATD Identification
4. SATD Instances Output
5. Complexity, Coupling and Readability Measurement
   - McCabe's Cyclomatic Complexity
   - Coupling Between Object
   - Buse and Weimer Metric

Data Extraction Process



Bavota, Gabriele, and Barbara Russo. "A large-scale empirical study on self-admitted technical debt." *Proceedings of the 13th International Workshop on Mining Software Repositories*. ACM, 2016.

Empirical Study (3/3)
Data Analysis

SATD Diffusion (RQ1) : - The number of SATD (discrete number)
- The number of SATD (% of comments)
- Manual types analysis

SATD Evolution (RQ2) : - The change in percentage of SATD comments
- The number of SATD introduced and fixed
- The survival of SATD in terms of number of commits

SATD Relation to Quality (RQ3) : - Spearman Rank Correlation between quality metrics and number of SATD

Study Results (1/5)

RQ1 : SATD diffusion in open source system

Number of instances : 51 instances or 0.3% of comments

Code > Defect and Requirement > Design

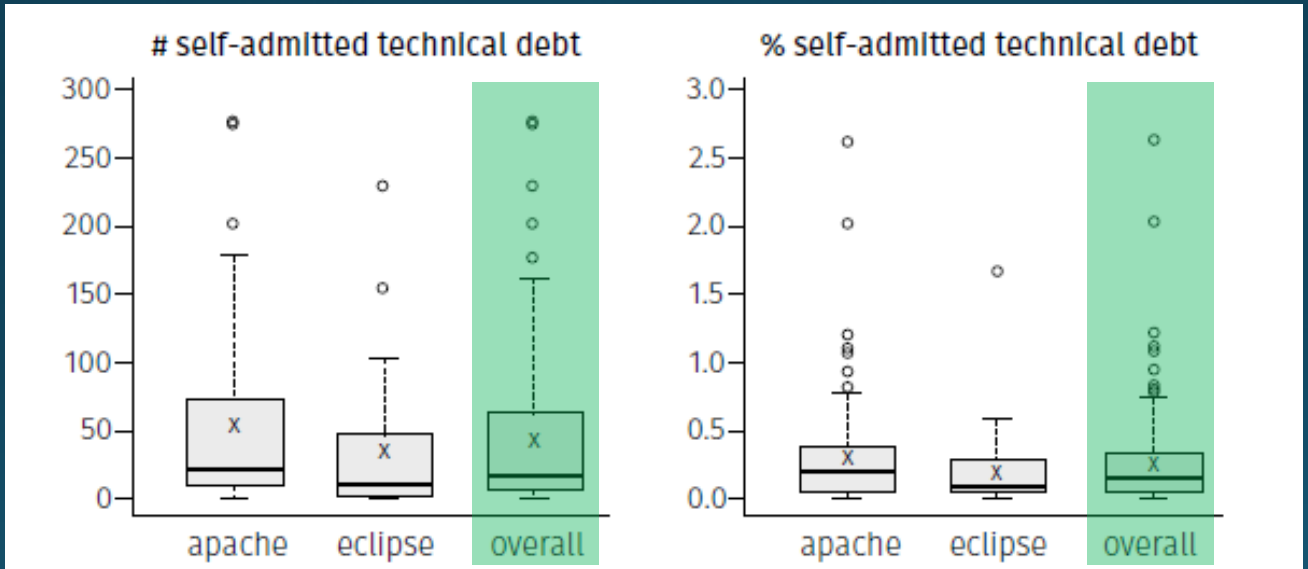


Figure 2: RQ_1: Diffusion of self-admitted technical debt. Absolute number (#) and percentage of comments reporting it (%).

Bavota, Gabriele, and Barbara Russo. "A large-scale empirical study on self-admitted technical debt." *Proceedings of the 13th International Workshop on Mining Software Repositories*. ACM, 2016.

Study Results (2/5)

RQ1 : SATD diffusion in open source system

Code : Quality of source code

Design : Object-Oriented design principles

Documentation : Misleading Information

Defect : Known and possible defects

Test : Quality of testing activities

Requirement : Functional and non functional requirement trade-offs

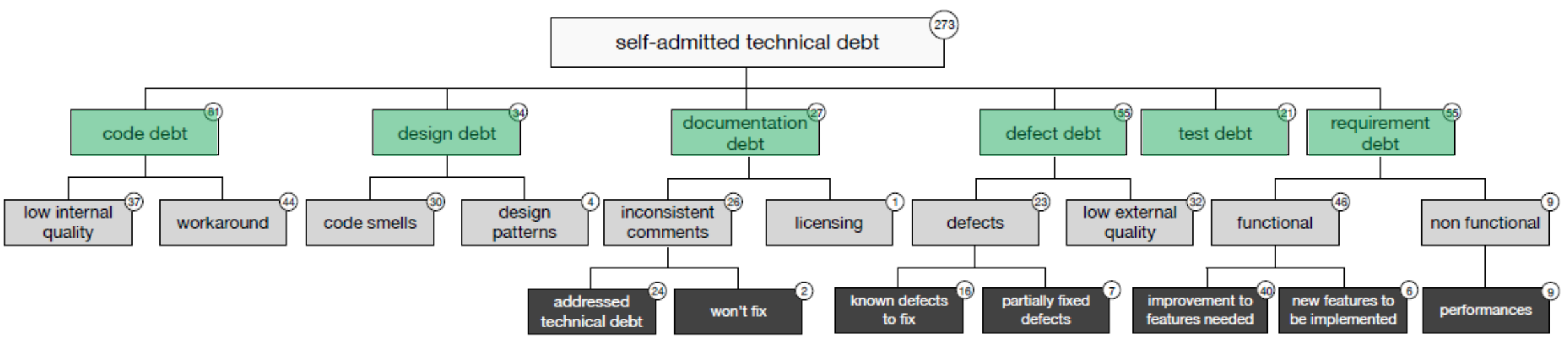


Figure 3: RQ_1: Manual categorisation of 366 self-admitted technical debt instances. Excluding 93 false positives.

Bavota, Gabriele, and Barbara Russo. "A large-scale empirical study on self-admitted technical debt." *Proceedings of the 13th International Workshop on Mining Software Repositories*. ACM, 2016.

Study Results (3/5)

RQ2 : SATD evolution during change history

Actors : The developer that introduces the SATD is generally the same that fix the SATD

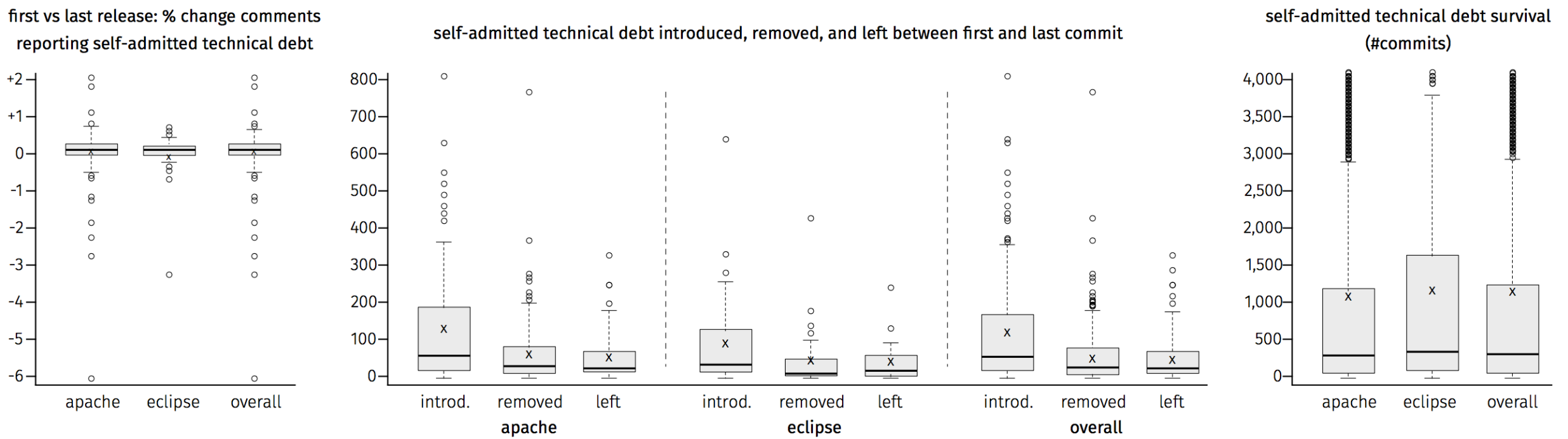


Figure 4: RQ_2: Evolution of self-admitted technical debt.

Bavota, Gabriele, and Barbara Russo. "A large-scale empirical study on self-admitted technical debt." *Proceedings of the 13th International Workshop on Mining Software Repositories*. ACM, 2016.

Study Results (4/5)

RQ2 : SATD evolution during change history

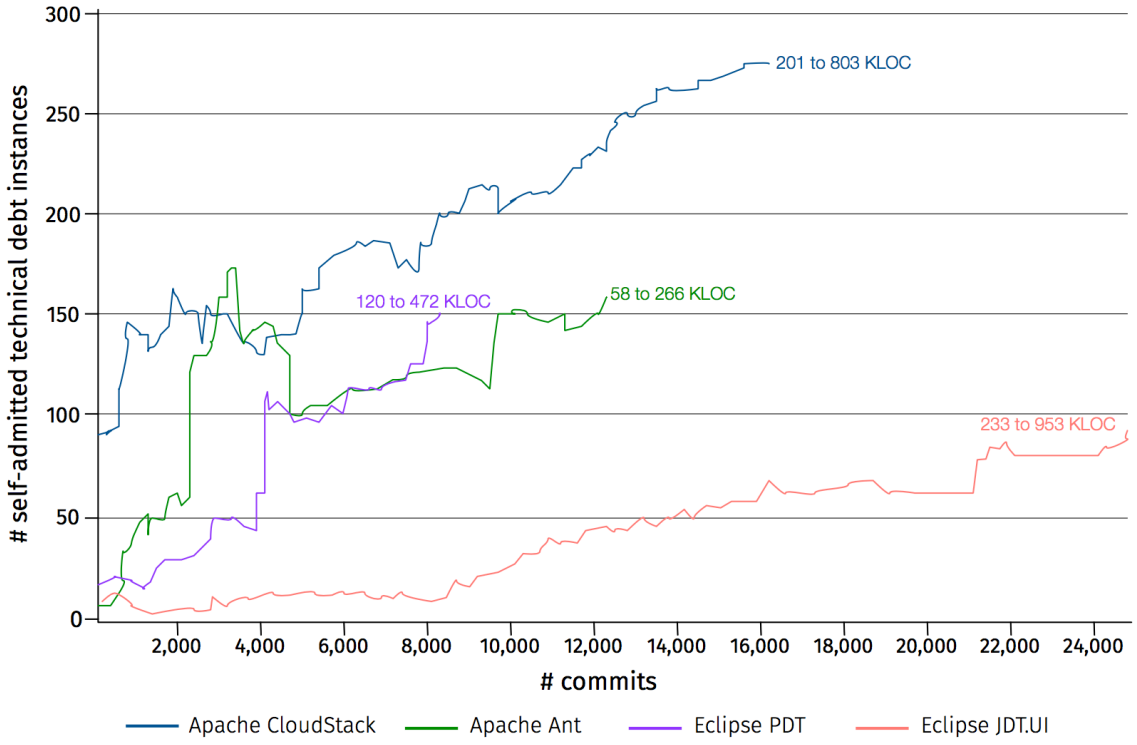


Figure 5: RQ_2: Evolution of self-admitted technical debt in four systems.

Bavota, Gabriele, and Barbara Russo. "A large-scale empirical study on self-admitted technical debt." *Proceedings of the 13th International Workshop on Mining Software Repositories*. ACM, 2016.

Study Results (5/5)

RQ3 : Proneness of low quality components to SATD

Table 2: RQ_3: Spearman partial correlation between quality attributes and instances of self-admitted technical debt (using LOC as controlling variable).

Coupling (CBO)		
Ecosystem	p-value	ρ
Apache	<0.001	0.02 (No Correlation)
Eclipse	<0.001	0.03 (No Correlation)
Overall	<0.001	0.03 (No Correlation)

Complexity (WMC)		
Ecosystem	p-value	ρ
Apache	<0.001	0.03 (No Correlation)
Eclipse	<0.001	0.03 (No Correlation)
Overall	<0.001	0.03 (No Correlation)

Readability		
Ecosystem	p-value	ρ
Apache	<0.001	0.01 (No Correlation)
Eclipse	<0.001	0.02 (No Correlation)
Overall	<0.001	0.02 (No Correlation)

Bavota, Gabriele, and Barbara Russo. "A large-scale empirical study on self-admitted technical debt." *Proceedings of the 13th International Workshop on Mining Software Repositories*. ACM, 2016.

Threat To Validity

- Imprecisions in the identification of SATD
- Subjectivity in the manual classification
- Imprecisions in the measurement of SATD survivability
- Impact of controlled factors
- Observational study
- Weak variety of projects studied

Future Work (1/3)
Technical Debt Analysis Tool

Self-Admitted Technical Debt: Intentionally introduced and documented

Technical Debt: Unintentionally introduced and not documented

Why?

- To fix bugs
- Initial implementation of components
- Poor knowledge of project

+++ Easy to implement
+++ Fast to implement

--- Energetically inefficient
--- Slow execution
--- Low maintainability

Future Work (2/3)
Technical Debt Analysis Tool

How to identify and locate technical debts not explicitly documented in a software project?

Develop a tool to identify SATDs

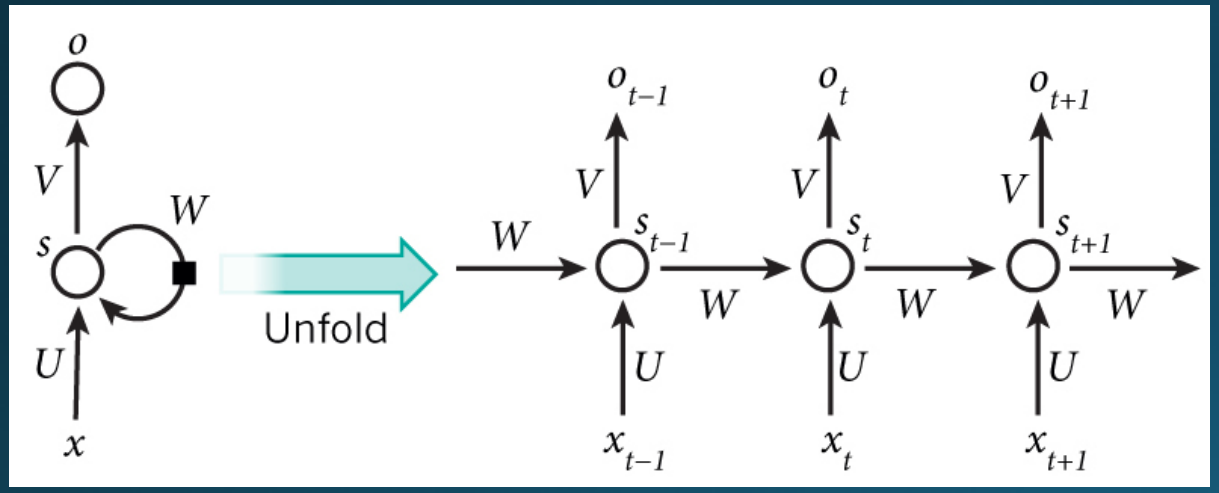
and

Use them to implement a **machine learning approach (neural network, Bayesian network, etc.)** to classify changes in a software project in term of technical debts

Purpose

Assess the quality of software projects

Future Work (3/3)
Technical Debt Analysis Tool



http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

1. Identify SATDs
2. Analyze the relevancy of the SATDs
3. Define the metrics to use for machine learning
4. Implement the machine learning algorithm to detect technical debts
5. Refine the machine learning algorithm
6. Evaluate software quality

Conclusion

SATD Diffusion

51 instances per system

Code, Defect, Requirement and Design Debt are most diffused

SATD Evolution

Survives 1000+ commits

Increases over project lifetime

Same person introduces and fixes a SATD

SATD Relation To Quality

Lack of correlation between internal quality and number of SATDs

Future Work

Use of a RNN model to identify technical debts in a software project

Questions