An Empirical Study On the Removal of Self-Admitted Technical Debt

Everton da S. Maldonado   Rabe Abdalkareem   Emad Shihab   Alexander Serebrenik

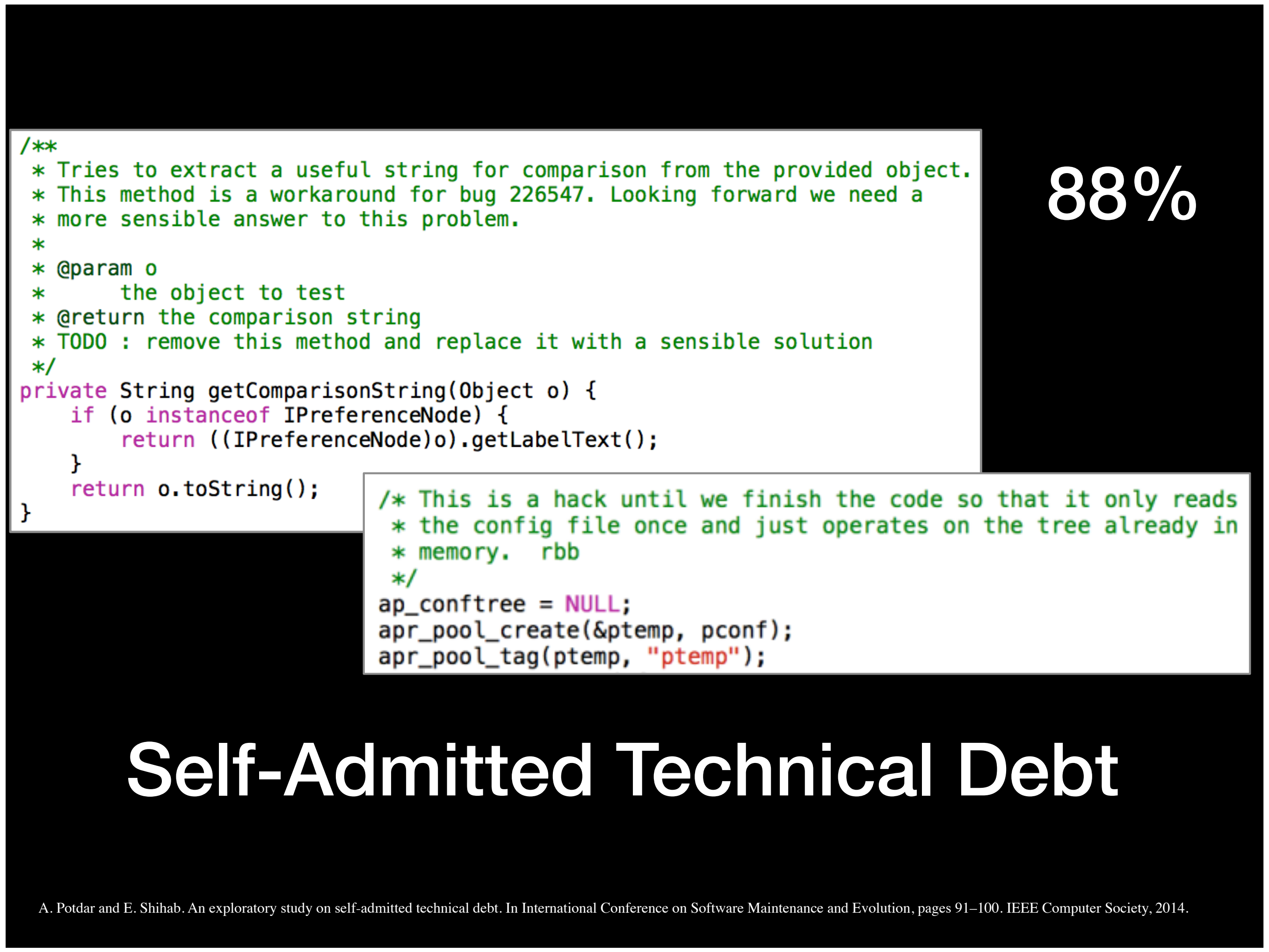
Concordia University, Canada   Eindhoven University of Technology, The Netherlands

@tonmaldonado   @rabekareem   @EmadShihab   @aserebrenik

Technical Debt

Shipping first-time code is like going into debt. A little debt speeds development so long as it is paid back promptly by rewriting. <…> The danger occurs when the debt is not repaid.

W. Canningham. The WyCash portfolio management system..OOPSLA '92 Addendum to the proceedings on Object-oriented programming systems, languages, and applications, pp. 29-30



```
/**
 * Tries to extract a useful string for comparison from the provided object.
 * This method is a workaround for bug 226547. Looking forward we need a
 * more sensible answer to this problem.
 *
 * @param o
 *      the object to test
 * @return the comparison string
 * TODO : remove this method and replace it with a sensible solution
 */
private String getComparisonString(Object o) {
    if (o instanceof IPreferenceNode) {
        return ((IPreferenceNode)o).getLabelText();
    }
    return o.toString();
}
```

88%

```
/* This is a hack until we finish the code so that it only reads
 * the config file once and just operates on the tree already in
 * memory.  rbb
 */
ap_conftree = NULL;
apr_pool_create(&ptemp, pconf);
apr_pool_tag(ptemp, "ptemp");
```

Self-Admitted Technical Debt

A. Potdar and E. Shihab. An exploratory study on self-admitted technical debt. In International Conference on Software Maintenance and Evolution, pages 91–100. IEEE Computer Society, 2014.

SATD refers to the situation where developers know that the current implementation is not optimal and write comments alerting the inadequacy of the solution. 88% of poor implementation choices are annotated through code comments, i.e. SATD [Vassalo et al. ICSME 2016]

Removal of Self-Admitted Technical Debt

So when one admits an error, then the next (expected) step would to be correct it. This is why we are looking at removal.

RQ 1: How much SATD gets removed?

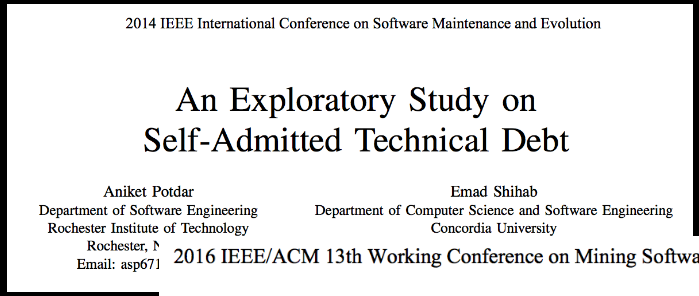
RQ 2: Is SATD more likely to be self-removed or removed by others?

RQ 3: How long does SATD survive in a project?

RQ 4: What activities lead to the removal of SATD?

RQ 1: How much SATD gets removed?

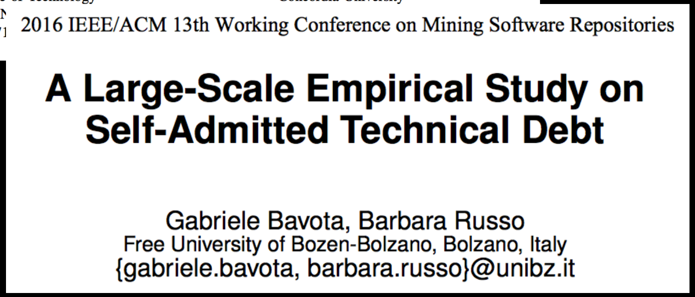
RQ 2: Is SATD more likely to be self-removed or removed by others?


2014 IEEE International Conference on Software Maintenance and Evolution

An Exploratory Study on Self-Admitted Technical Debt

Aniket Potdar
Department of Software Engineering
Rochester Institute of Technology
Rochester, N
Email: asp671

Emad Shihab
Department of Computer Science and Software Engineering
Concordia University


2016 IEEE/ACM 13th Working Conference on Mining Software Repositories

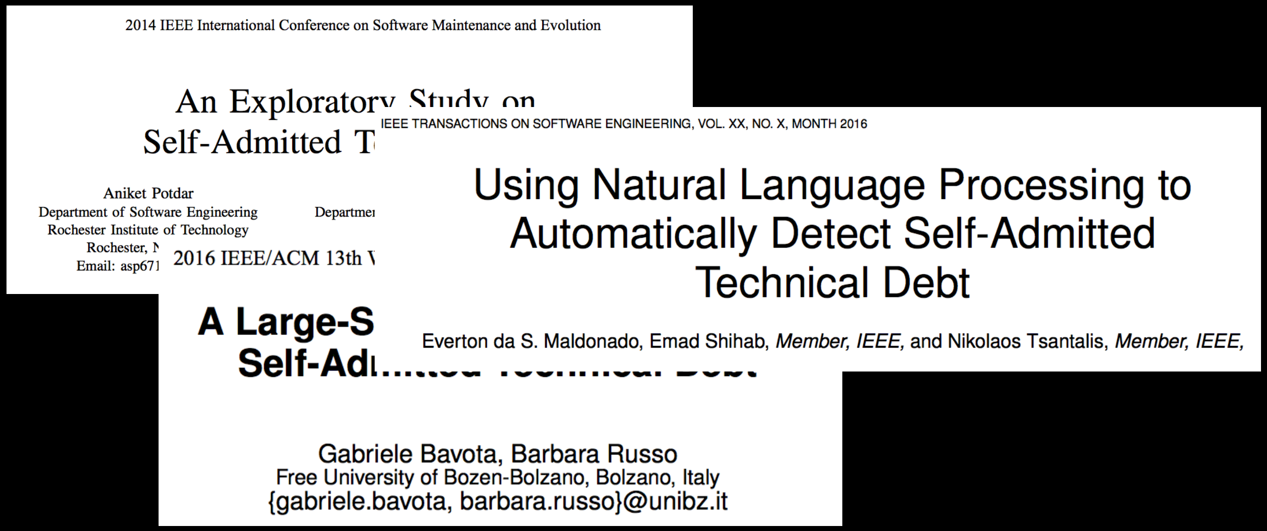
A Large-Scale Empirical Study on Self-Admitted Technical Debt

Gabriele Bavota, Barbara Russo
Free University of Bozen-Bolzano, Bolzano, Italy
{gabriele.bavota, barbara.russo}@unibz.it

Why would we study these questions if they have been investigated in the past by Potdar&Shihab, and by Bavota&Russo?

RQ 1: How much SATD gets removed?

RQ 2: Is SATD more likely to be self-removed or removed by others?



2014 IEEE International Conference on Software Maintenance and Evolution

An Exploratory Study on
Self-Admitted T

Aniket Potdar
Department of Software Engineering
Rochester Institute of Technology
Rochester, N
Email: asp671

Departme

2016 IEEE/ACM 13th V

A Large-S
Self-Ad

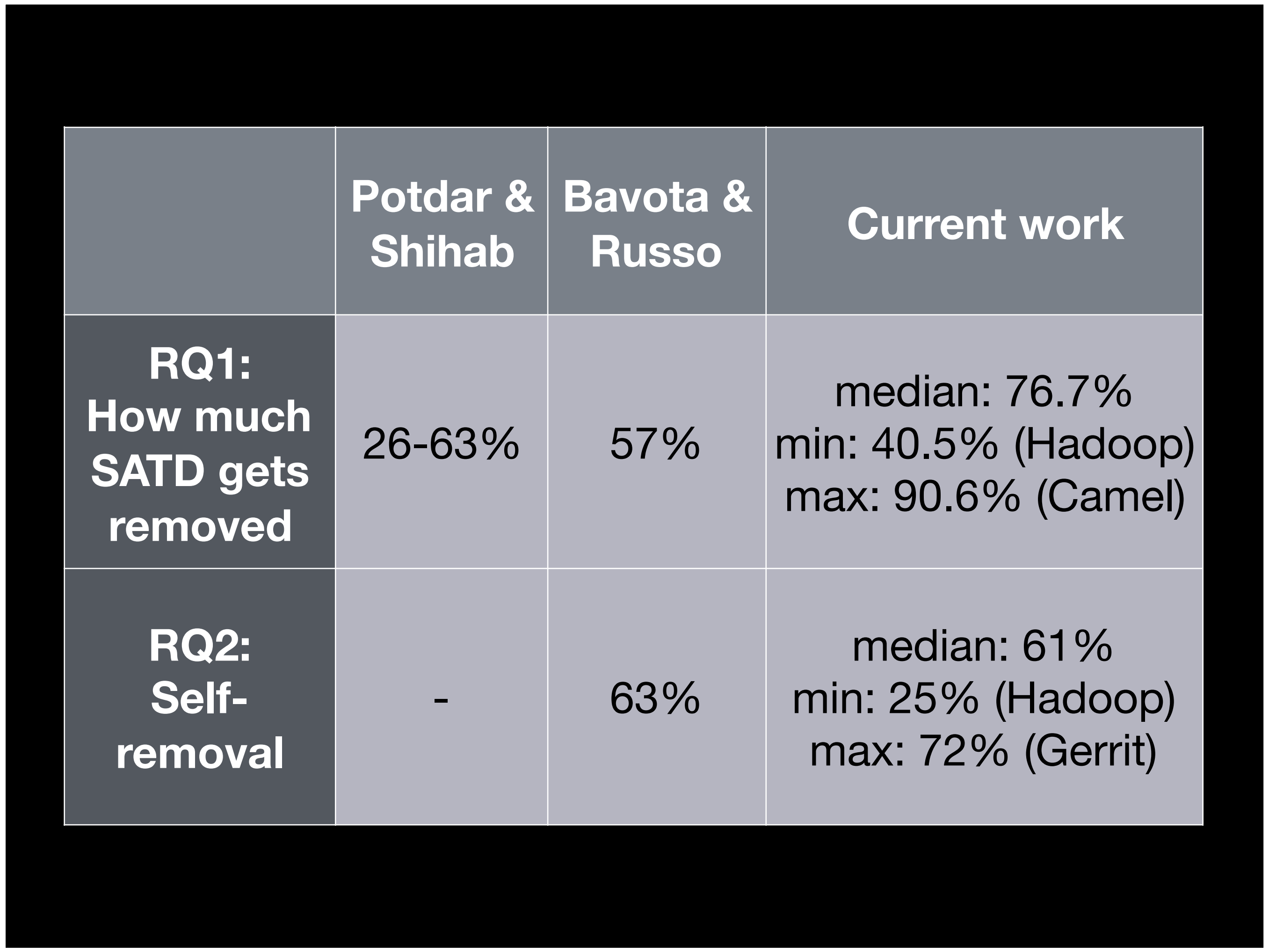
Gabriele Bavota, Barbara Russo
Free University of Bozen-Bolzano, Bolzano, Italy
{gabriele.bavota, barbara.russo}@unibz.it

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. XX, NO. X, MONTH 2016

Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt

Everton da S. Maldonado, Emad Shihab, *Member, IEEE,* and Nikolaos Tsantalis, *Member, IEEE,*

Why would we study these questions if they have been investigated in the past by Potdar&Shihab, and by Bavota&Russo? Of course, we have a slightly different data but most important we have a different detection technique which improves the detection accuracy (mostly in terms of recall) by a significant margin over the comment pattern approach used in the prior work (2.3X improvement).



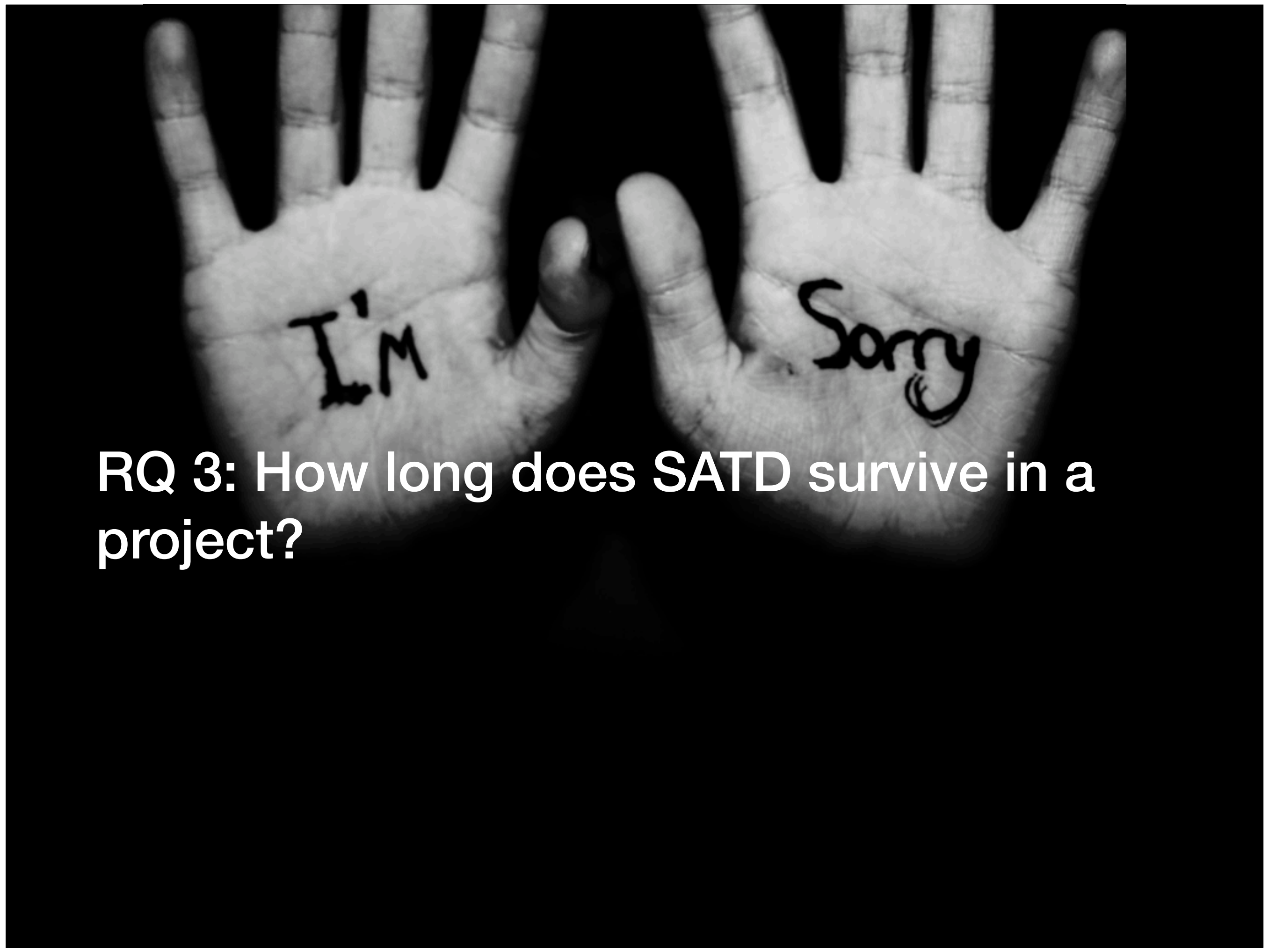
	Potdar & Shihab	Bavota & Russo	Current work
RQ1: How much SATD gets removed	26-63%	57%	median: 76.7% min: 40.5% (Hadoop) max: 90.6% (Camel)
RQ2: Self-removal	-	63%	median: 61% min: 25% (Hadoop) max: 72% (Gerrit)

Results are consistent with the previous study.

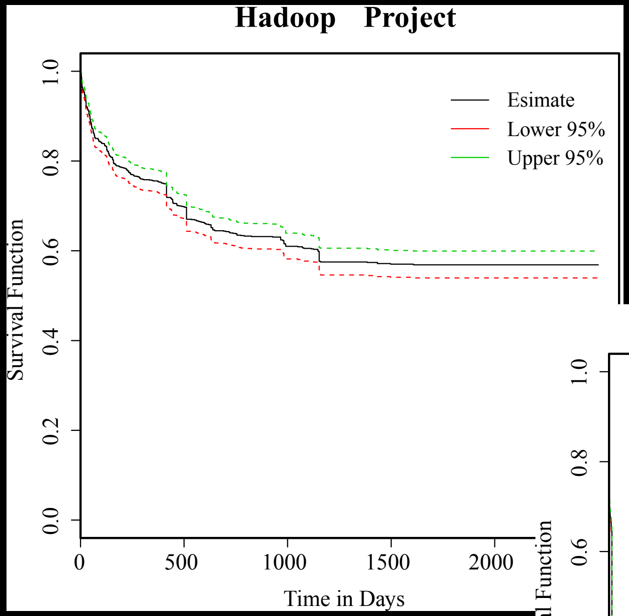
Potdar & Shihab: 4 large systems (Eclipse, Chromium, Apache httpd, Arg UML)

Bavota & Russo: 159 projects from Apache and Eclipse

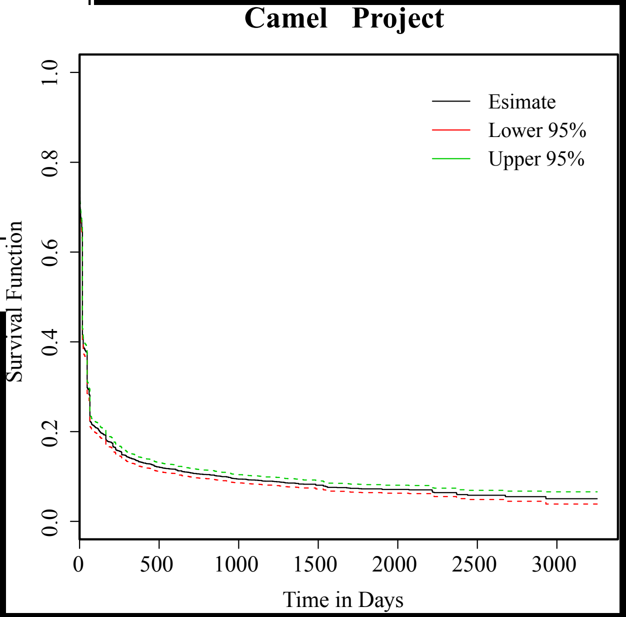
We: 5 systems (Camel, Gerrit, Hadoop, Log4j, Tomcat)



RQ 3: How long does SATD survive in a project?



Hadoop is slow

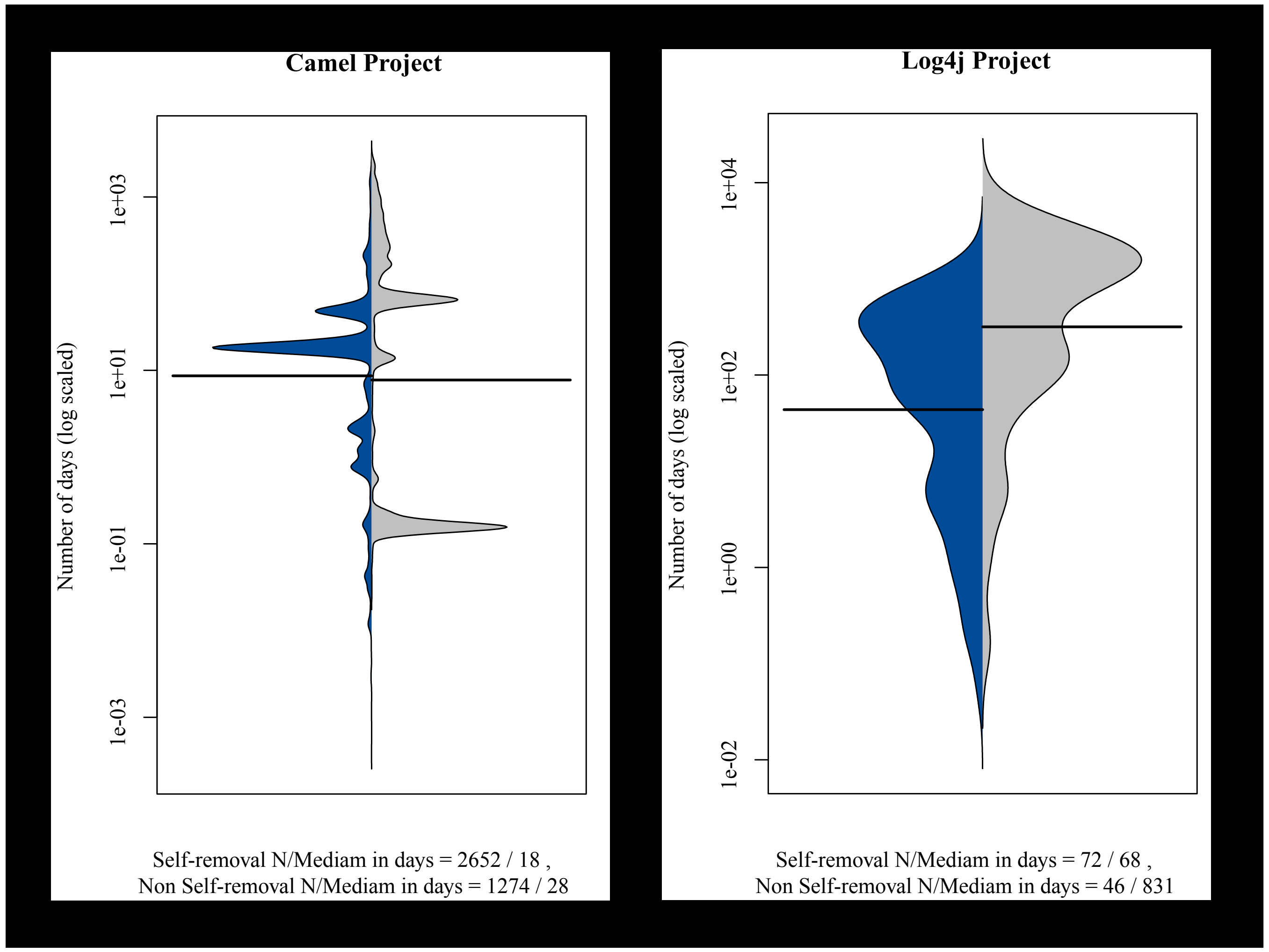


Camel, Gerrit, Log4j are fast

Bavota & Russo This 57% of fixed instances stays in the system for very long time (over 1,000 commits on average (median=266)). We see big differences between the projects here. Tomcat is somewhere in between

Aware + Guilty = Quick(er) Fix?

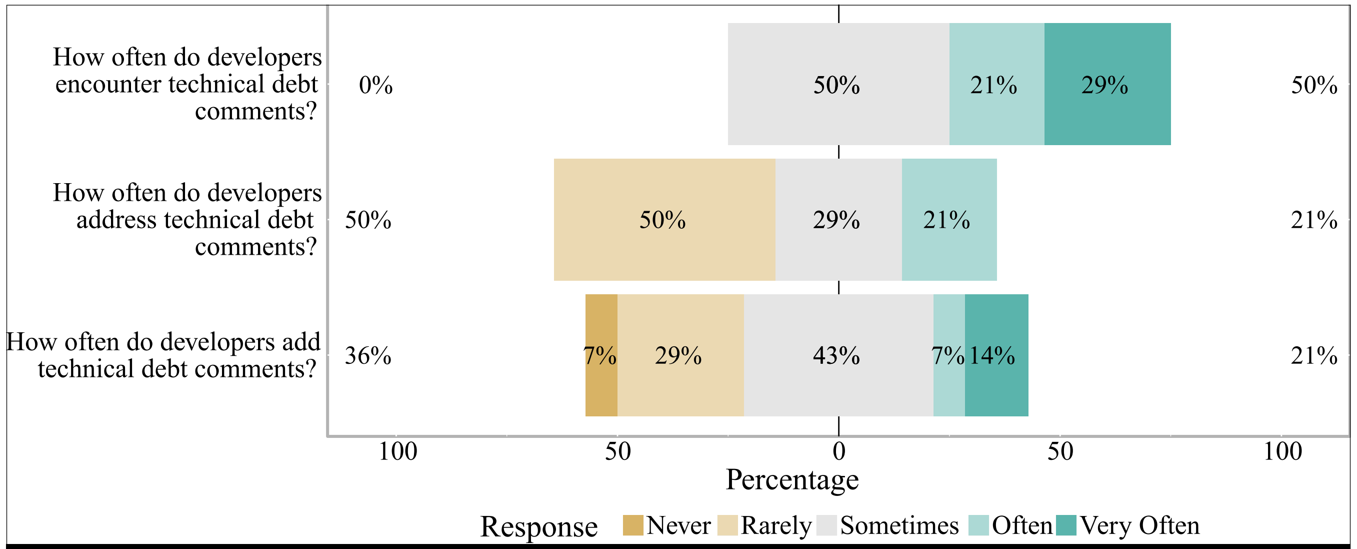
Furthermore, we distinguish between self-removal and non-self-removal, and expect that the self-removal is faster since we believe that developers that admit their mistakes should be more eager to fix them, as opposed to other developers that first need to find the mistakes and then might be less inclined to fix somebody else’s mistakes (aware + guilty as opposed to not aware and not guilty).



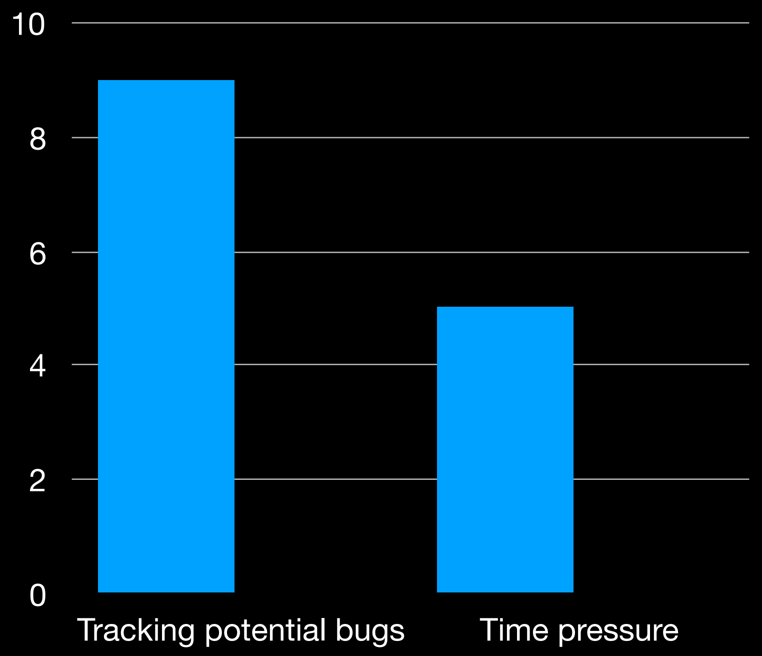
Differences are always statistically significant. For Camel the effect size is small (both self and non-self are fast), for all other projects - large. Log4j is just an example. NB: log scale

RQ 4: What activities lead to the removal of SATD?

To answer this question we have contacted all developers that have been involved in introducing or removing SATD in the five projects we have studied + 2 additional projects. 250 identified, 188 mails successfully sent, 14 responses
“the area of technical debt is difficult to discuss, especially since some developers may feel they or their projects will be negatively perceived”

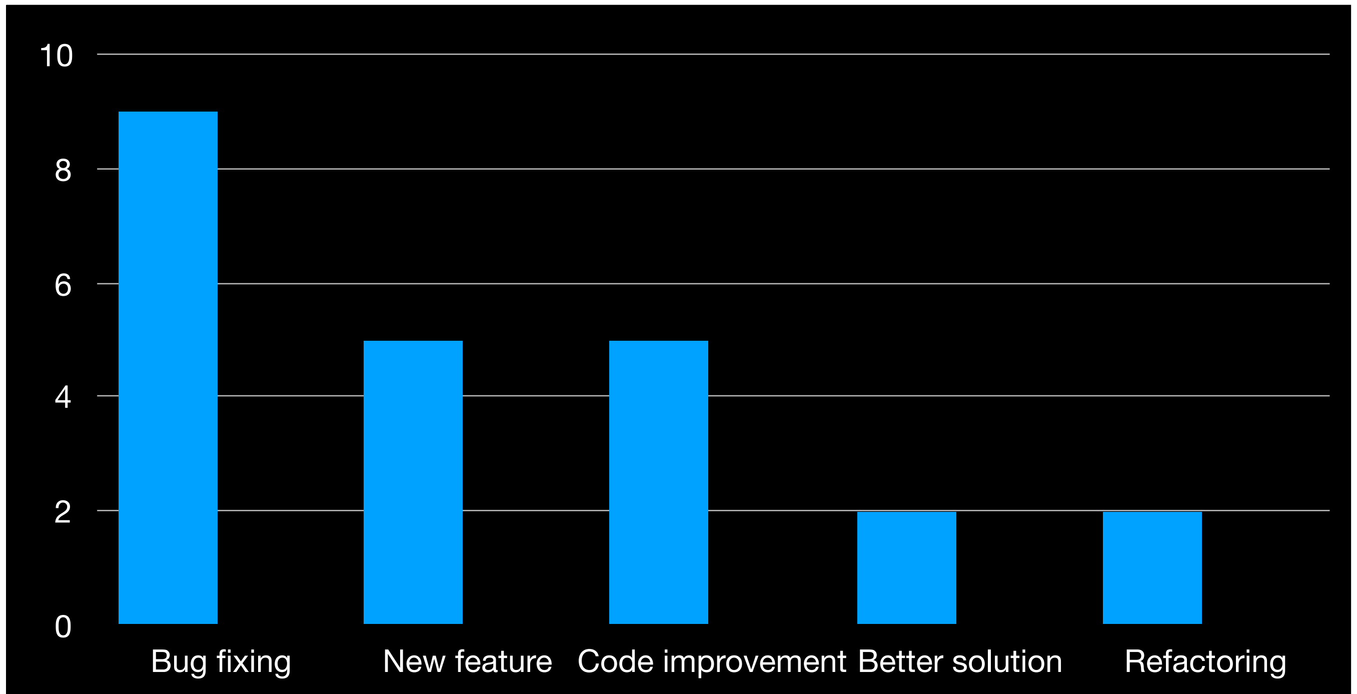


Developers often encounter SATD but rarely address it



Reasons for adding SATD

Number of answers, the same respondent could have provided several answers



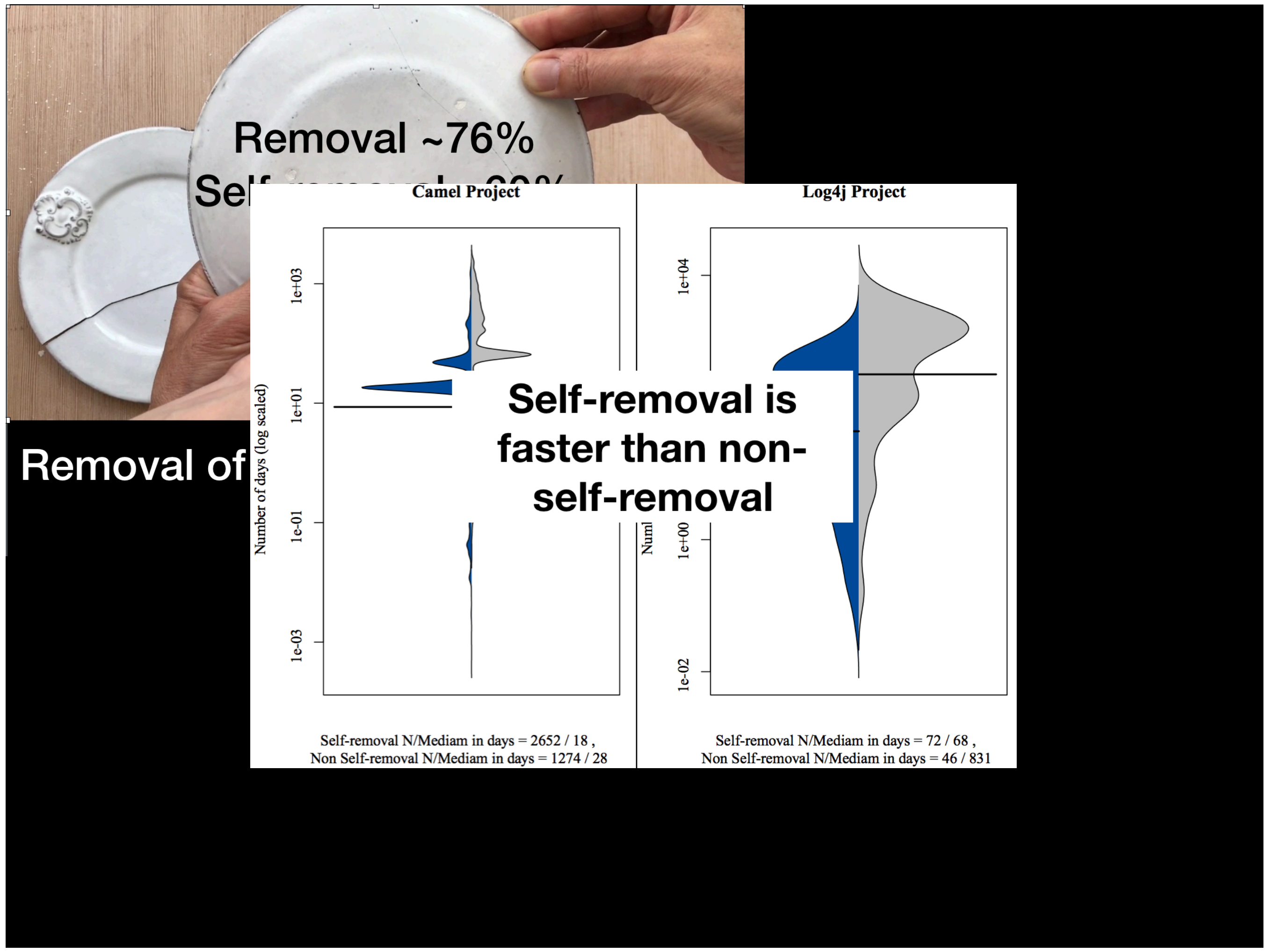
Reasons for addressing SATD

Number of answers, the same respondent could have provided several answers. There is no formal process of SATD removal, SATD is being removed as part of regular development activities such as bug fixing/implementation of new features. Similar to FLOSS refactoring as opposed to root canal refactoring (Black & Murphy-Hill)

Is Code improvement alias of SATD removal?

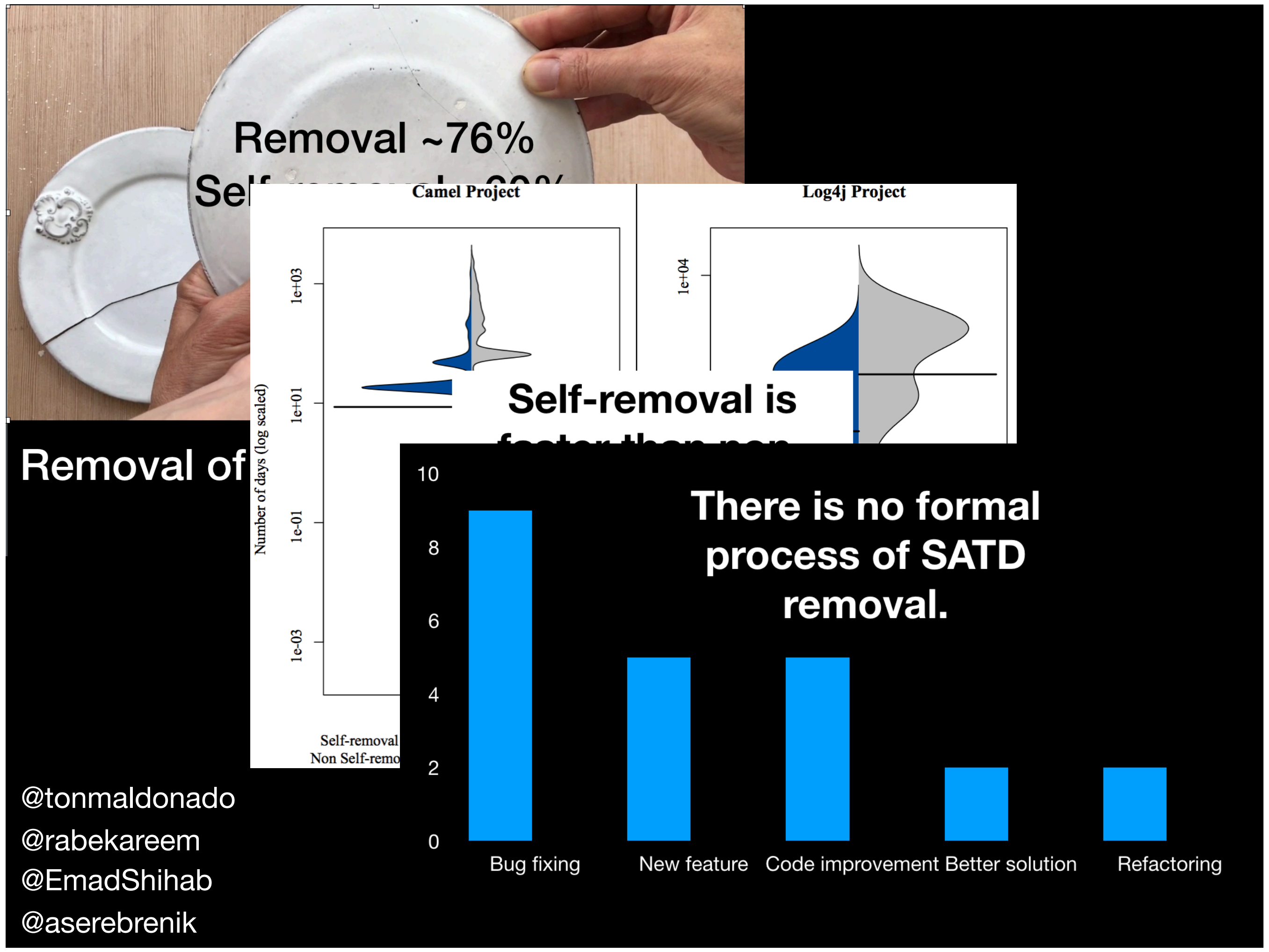
Removal of Self-Admitted Technical Debt

Removal of SATD can be seen as an important activity since large share of it is being removed (baseline).



Self-removal is faster than non-self-removal

Moreover, we observe that self-removal is faster than non-self removal and conjecture that developers are aware of their own SATD and are motivated to pay it back.



Removal of

Self-removal is

There is no formal process of SATD removal.

@tonmaldonado
@rabekareem
@EmadShihab
@aserebrenik

Moreover, we observe that self-removal is faster than non-self removal and conjecture that developers are aware of their own SATD and are motivated to pay it back.