A Method to Identify and Correct Problematic Software Activity Data: Exploiting Capacity Constraints and Data Redundancies

Qimu Zheng
Peking University

Audris Mockus
University of Tennessee

Minghui Zhou
Peking University


Software activity data
are important.

Code

Bug report

More Available Data

Mailing list

Social media

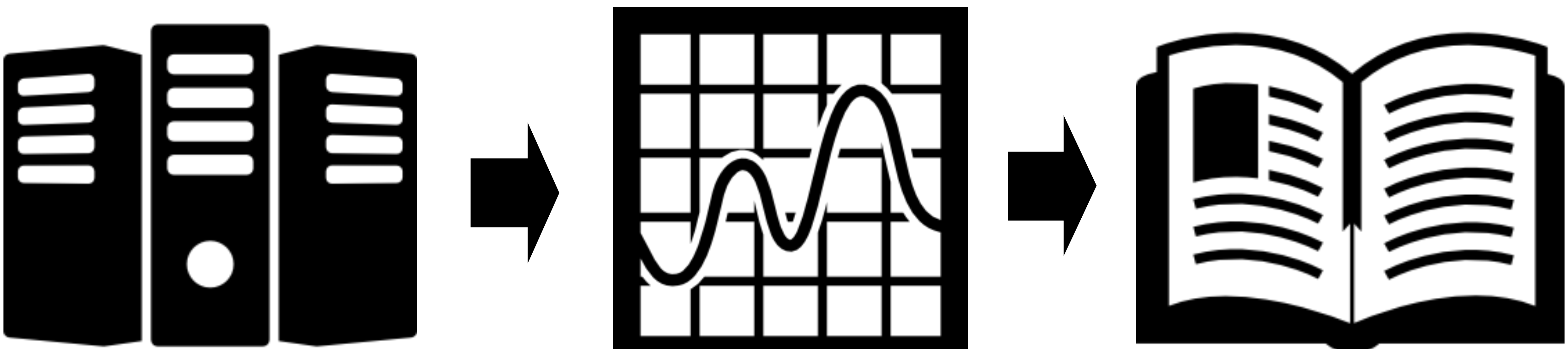
GitHub
Commits

Atlassian Bitbucket

stackoverflow

JIRA

Empirical SE Research



Software Activity Data Statistical Model Research Result

Various Topics

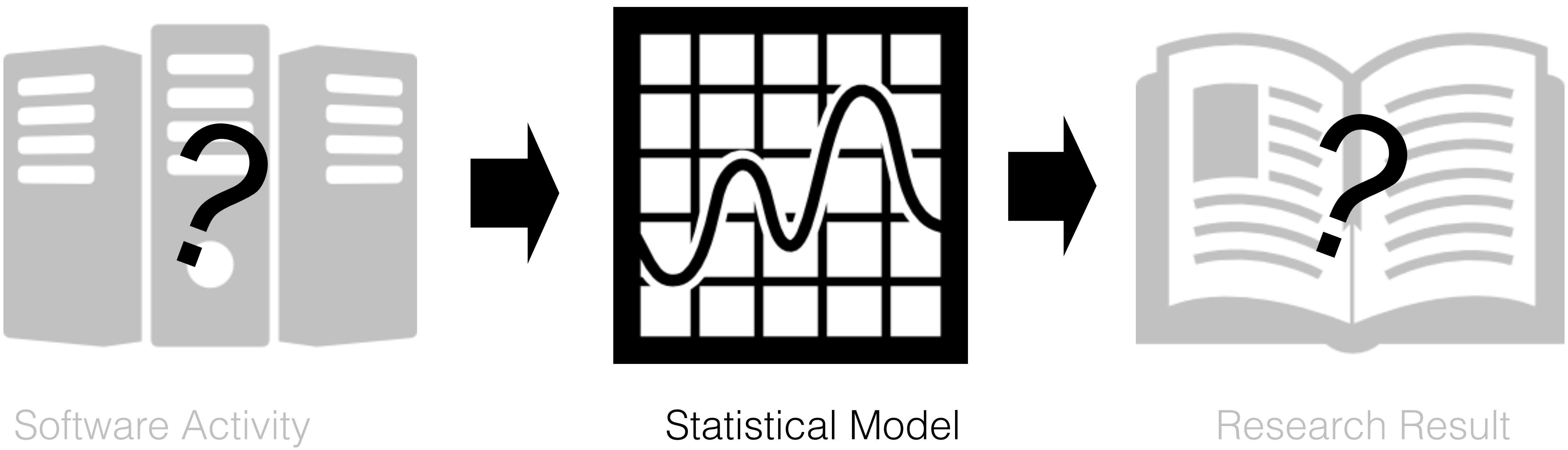
- Measure Productivity[1]
- Duplicate Bug Report Prediction[2]
- Bug-fix Time Prediction[3]
- …

[1] W. F. Boh, S. A. Slaughter, and J. A. Espinosa. Learning from experience in software development: A multilevel analysis.

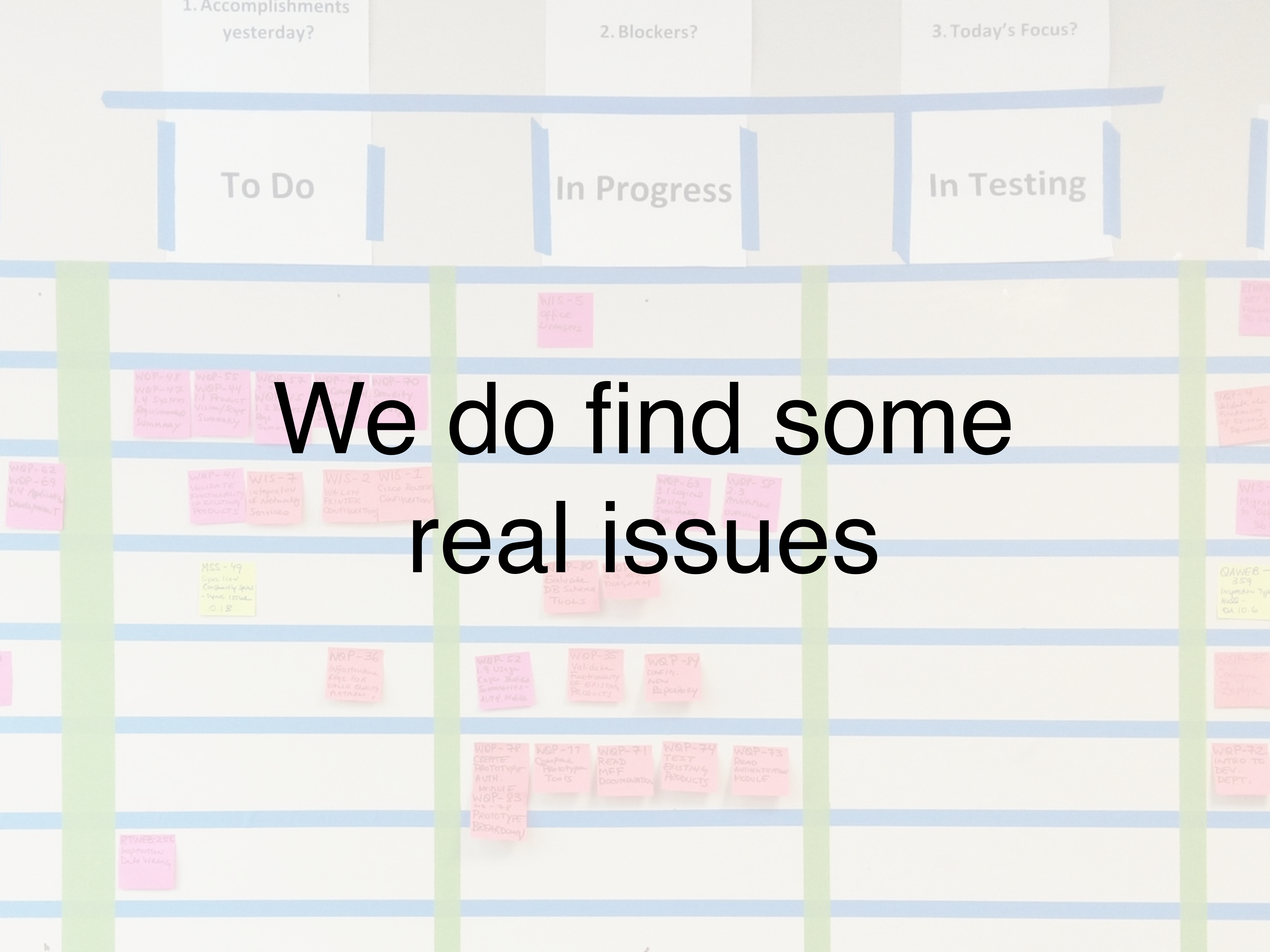
[2] Chengnian Sun; Lo, D.; Siau-Cheng Khoo; Jing Jiang, Towards more accurate retrieval of duplicate bug reports

[3] P. Bhattacharya and I. Neamtiu. Bug-fix time prediction models: Can we do better?

However …



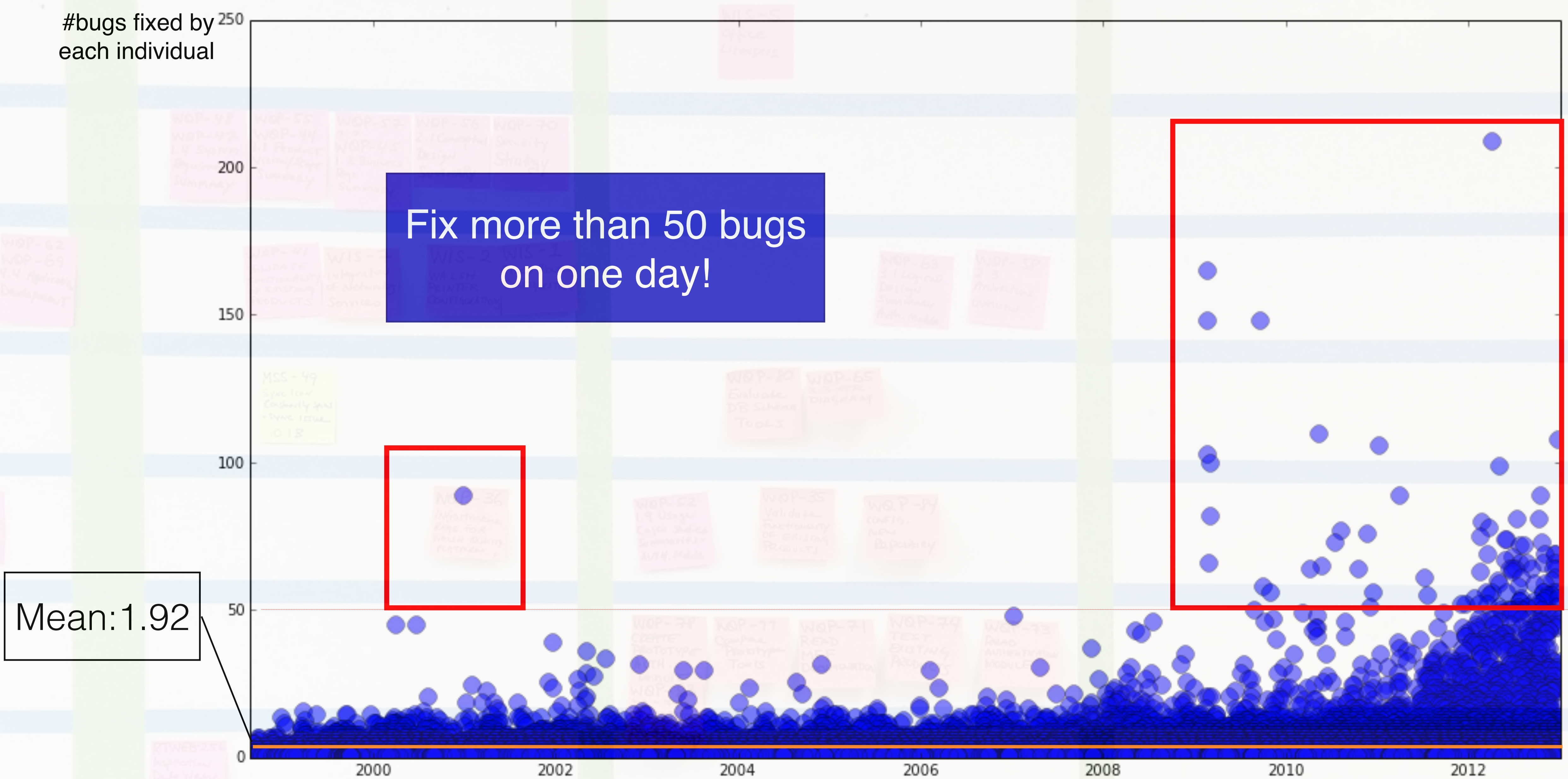
We do find some real issues



Some real issues

- **Task completion time** is important
  - For both research and practical development
- Bug fixing time recorded in ITS is often used
- Count #bugs fixed by each dev on each day
- Experiment on official data from Mozilla

Some real issues



Research on data quality?

Limited amount of work can be found.

Research mentioning data quality?

Data quality consideration is a minority practice [1][2].

This laundry-list of potentially serious data quality issues is often either ignored or treated on a case-by-case basis in much of the extant work.[3]

[1] Michael Franklin Bosu and Stephen G. MacDonell. 2013. Data quality in empirical software engineering: a targeted review.

[2] Gernot A. Liebchen and Martin Shepperd. 2008. Data sets and data quality in software engineering.

[3] Audris Mockus. 2014. Engineering big data solutions.

Research on Data Quality?

Limited amount of work can be found.

Researchers love data.

Yet few care about their quality.

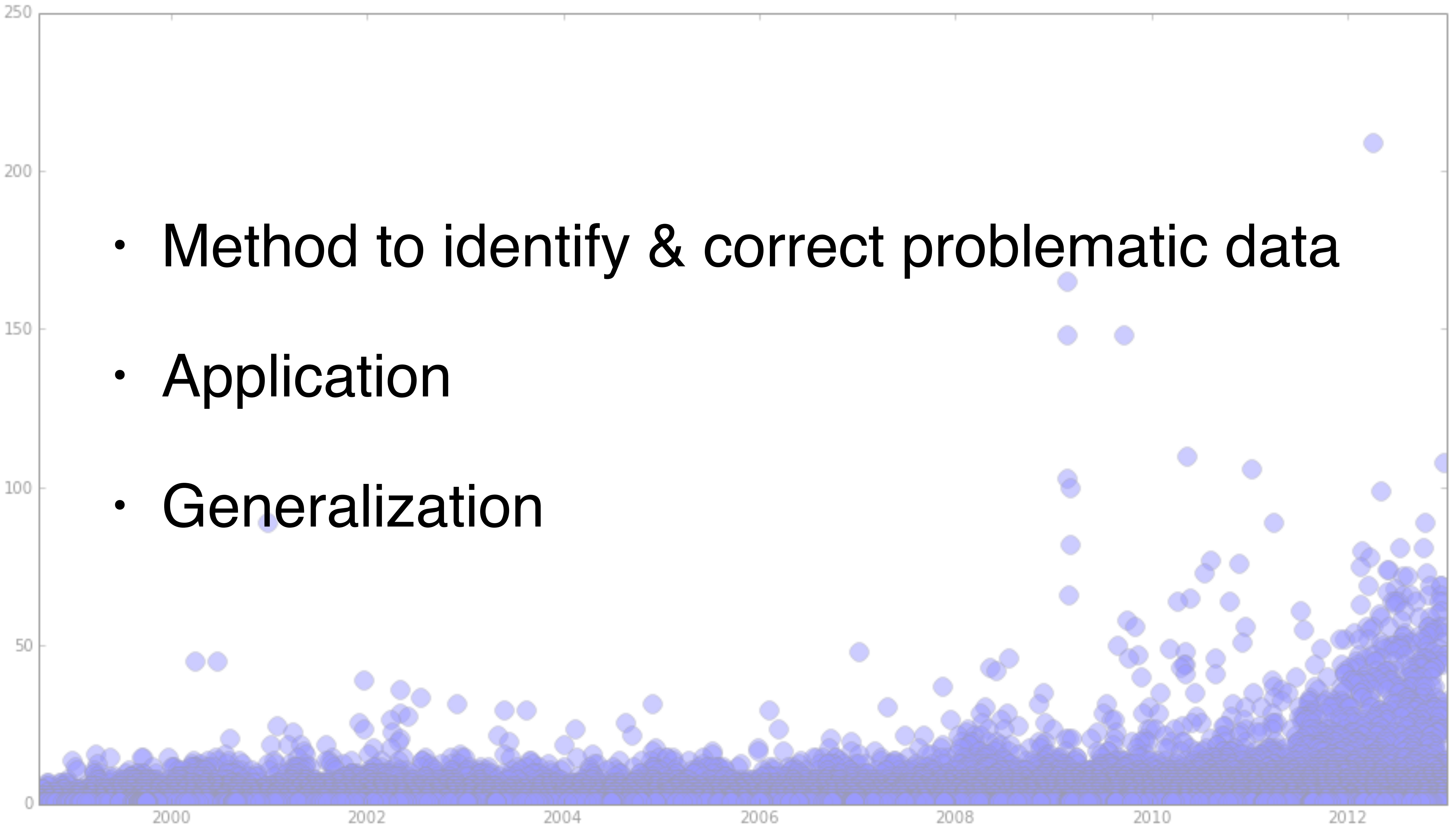
Data quality consideration is a minority practice [1][2].

[1] Michael Franklin Bosu and Stephen G. MacDonell. 2013. Data quality in empirical software engineering: a targeted review.

[2] Gernot A. Liebchen and Martin Shepperd. 2008. Data sets and data quality in software engineering.

So we try to fix the problem

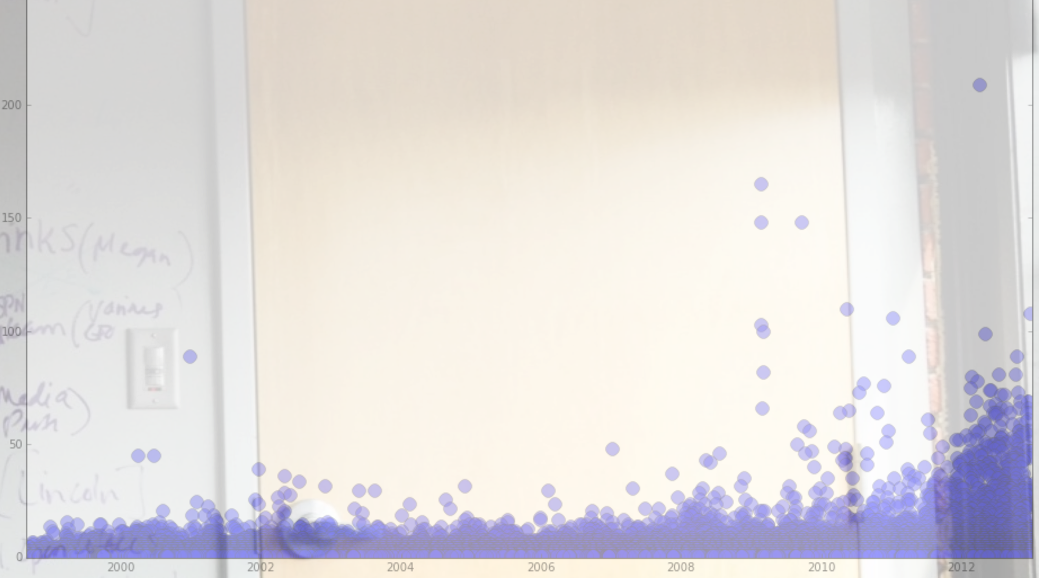
- Method to identify & correct problematic data
- Application
- Generalization



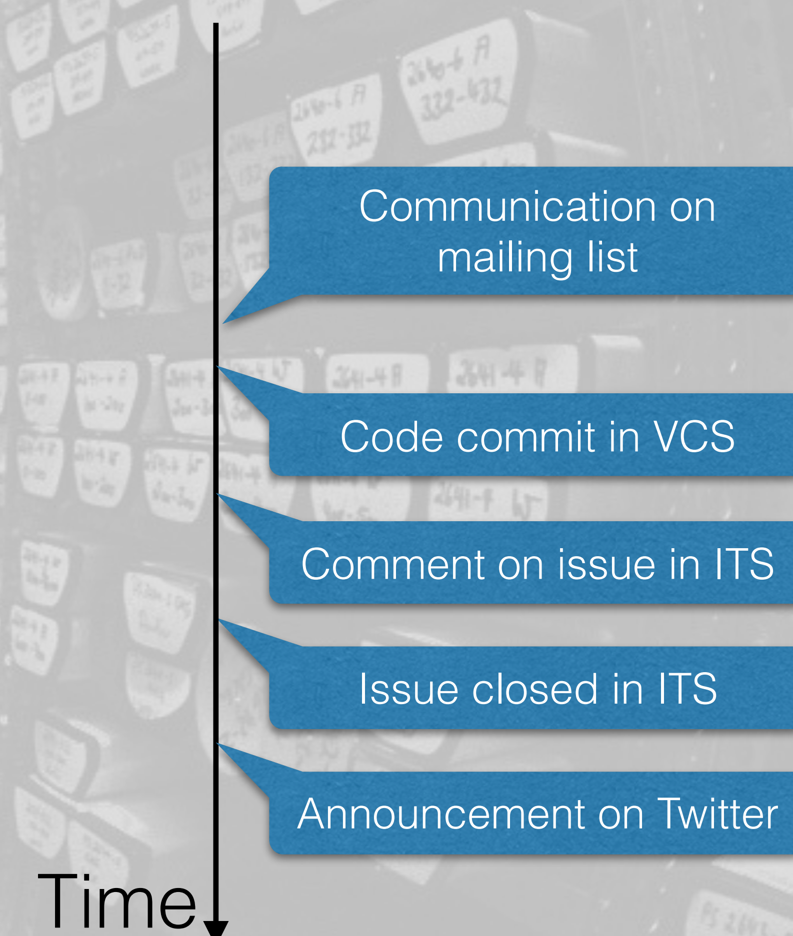
Before that…

Two observations about software activity data

- Capacity constraints



- Data redundancies



Method

for identifying and correcting
problematic data

- **Gather data**

- Choose **primary event type** (default choice)

Data Redundancies

- Choose a set of **redundant event types** (some approximation)

- Obtain event times t_{ik} for task i and event k.

Capacity Constraints

- Use the distribution of t_{ik} to **identify problematic data**.

$$isProblematic(t_{ik}) = \text{the likelihood that } t_{ik} \text{ being incorrect.}$$

- Obtain $isProblematic(t_{ik})$ for each redundant observation type k.

- **Correct problematic data**. Choose observations via:

$$correct(t_i) = \begin{cases} \arg\min_{k>1}(isProblematic(t_{ik})) & if\ isProblematic(t_{i1}) \\ t_{i1} & if\ !isProblematic(t_{i1}) \end{cases}$$

Shorter Version

- **Gather data**
- **Choose primary event type** (default choice)
- **Choose redundant event types** (some approximation)
- **Identify problematic data**

$$isProblematic(t_{ik}) = \text{the likelihood that } t_{ik} \text{ being incorrect.}$$

- **Correct problematic data.**

$$correct(t_i) = \begin{cases} \arg\min_{k>1}(isProblematic(t_{ik})) & if\ isProblematic(t_{i1}) \\ t_{i1} & if\ !isProblematic(t_{i1}) \end{cases}$$

Even Shorter

- **Data Gathering**
- **Primary Event Type**
- **Redundant Event Types**
- **Problematic Data Identification**
- **Problematic data Correction**

- Data Gathering
- Primary Event Type
- Redundant Event Types
- Problematic Data Identification
- Problematic Data Correction

Application

of the proposed method

Data Gathering

- Official Bugzilla dump from Mozilla (January 2013)
- All code commits data from Mozilla (February 2014)

Primary Event Type

Bug-fix time recorded in issue tracking system.

cdawson	2012-04-03 08:58:14 PDT	Status	NEW	RESOLVED
		Resolution	---	FIXED
		Last Resolved		2012-04-03 08:58:14

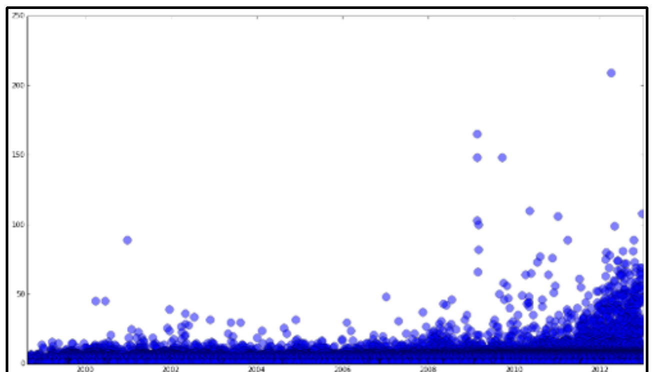
Redundant Event Types?

Choose by understanding error mechanisms!

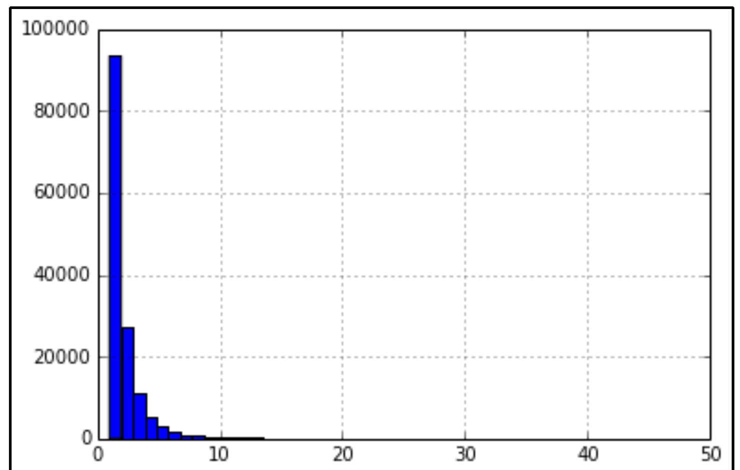
Redundant Event Types

- Error mechanisms
  - Development Process Tracked By Other System
  - Dormant issues
  - Closing issues with committed patches
- Good substitutes:
  - Last comment time
  - Last code commit time

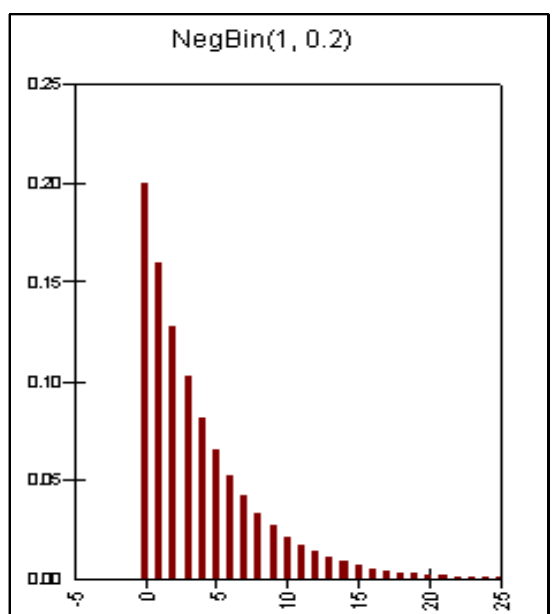
Problematic Data Identification



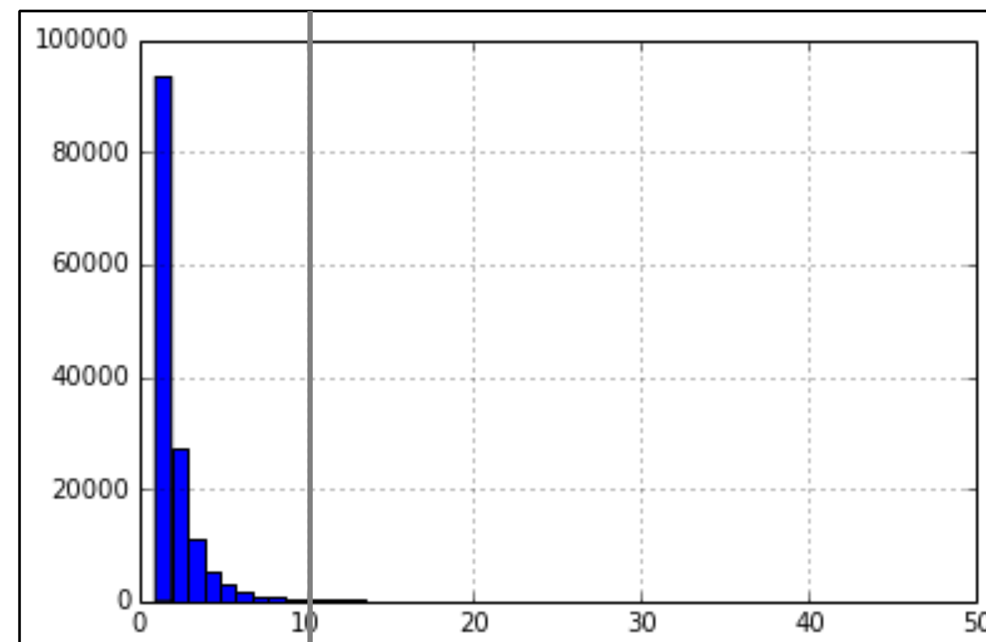
Data



Histogram



0-truncated negative binomial distribution



10 as cut-off

Problematic Data Correction

- Available options:
  - Last comment time
  - Last commit time
- Since last commit time will be used for evaluation, we use **last comment time** for correction:

$$correct(t) = \begin{cases} \text{last comment time} & if \;\; isProblematic(\text{ITS recorded time}) \\ \text{ITS recorded time} & if \; !isProblematic(\text{ITS recorded time}) \end{cases}$$

Does it matter?

¯_(ツ)_/¯


Data Accuracy


Impacts on Research

Data Accuracy

- 16% of the issues are fixed with a link pointing to some commits in version control system (VCS)
- We take the timestamp in VCS as gold standard for evaluation

$$\text{absolute error} = |\text{timestamp} - \text{vcs timestamp}|$$

$$\text{relative error} = \frac{|\text{timestamp} - \text{vcs timestamp}|}{\text{vcs timestamp} - \text{issue creation time}}$$

Absolute Error

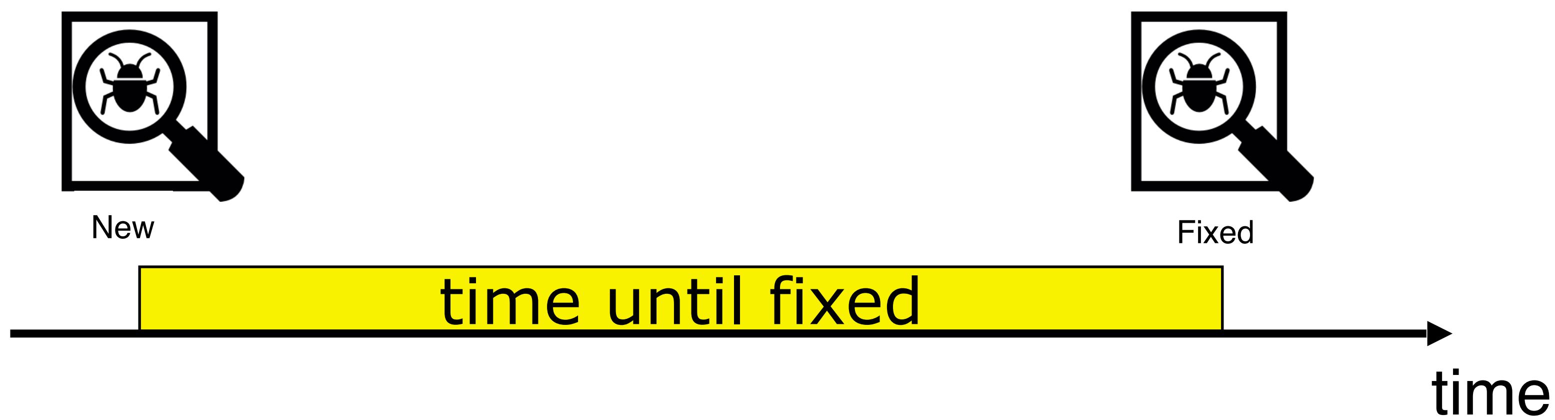
Quantile	Uncorrected	Corrected
0.50	0d 07:17:13	0d 01:08:17
0.75	1d 00:16:33	1d 11:03:00
0.80	1d 08:52:50	0d 21:21:03
0.90	5d 21:59:42	4d 12:40:42
0.99	75d 03:43:39	72d 11:18:15

Relative Error

Quantile	Uncorrected	Corrected
0.50	0.0205	0.0073
0.75	0.2105	0.0777
0.80	0.3700	0.1544
0.90	1.6504	0.8502
0.99	148.2818	73.3260

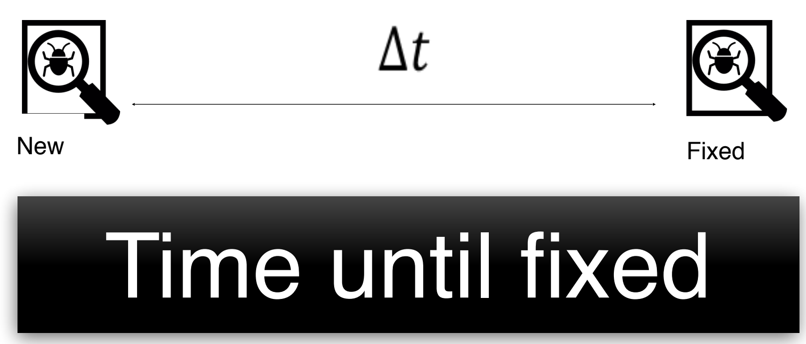
Impacts on Research

Time until fix



Existing research

- Summary
- Severity
- Priority
- Product
- Description



Impacts on Research

$$\ln(days+1) \sim \text{severity} + \ln(attachments+1) + reputation + \ln(assignee+1)$$
$$+\ln(depends+1) + priority + late + \ln(comments+1) + resolver + last\_commenter$$

	Estimate	p-value
(Intercept)	4.91	0.00
Critical	0.39	0.00
Major	0.64	0.00
Normal	0.80	0.00
Minor	1.02	0.00
Trivial	0.75	0.00
Enhancement	1.23	0.00
ln(attachments+1)	-0.16	0.00
ln(depends+1)	0.62	0.00
ln(assignee+1)	0.32	0.00
Reputation	-1.04	0.00
P1	-0.22	0.00
P2	0.08	0.11
P3	0.32	0.00
P4	0.52	0.00
P5	1.33	0.00
ln(comments+1)	0.54	0.00
Resolver	-0.22	0.00
Late	-0.72	0.00

	Estimate	p-value
(Intercept)	-2.23	0.02
Critical	0.28	0.01
Major	0.43	0.00
Normal	0.60	0.00
Minor	0.75	0.00
Trivial	0.75	0.00
Enhancement	1.12	0.00
ln(attachments+1)	-0.12	0.00
ln(depends+1)	0.41	0.00
ln(assignee+1)	0.45	0.00
Reputation	-0.52	0.00
P1	-0.09	0.05
P2	0.20	0.00
P3	0.43	0.00
P4	0.49	0.00
P5	0.85	0.00
ln(comments+1)	1.08	0.00
Resolver	-0.21	0.00
Late	-0.20	0.00

Impacts on Research

$$\ln(days+1) \sim \text{severity} + \ln(attachments+1) + reputation + \ln(assignee+1)$$
$$+ \ln(depends+1) + priority + late + \ln(comments+1) + resolver + last\_commenter$$

R2: 0.381 => 0.452

Predictors: 4 significancy changes

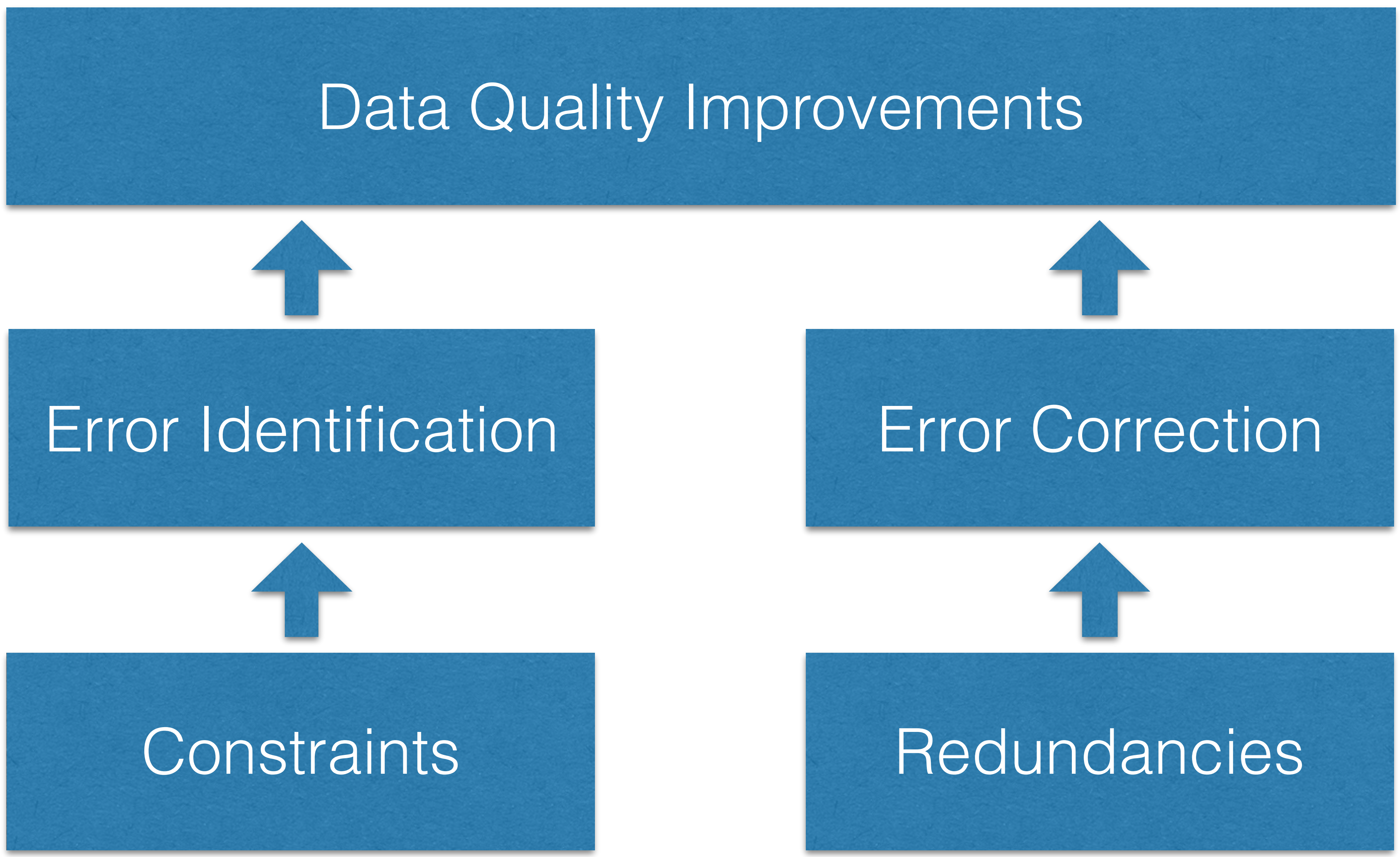
	Estimate	p-value
(Intercept)	4.91	[illegible]
Critical	0.39	0.00
Major	0.64	0.00
Normal	0.80	0.00
Minor	1.02	0.00
Trivial	0.75	0.00
Enhancement	1.23	0.00
ln(attachments+1)	[illegible]	0.00
ln(depends+1)	0.62	0.00
ln(assignee+1)	0.32	0.00
Reputation	-1.04	0.00
P1	-0.22	0.00
P2	0.08	0.11
P3	0.32	0.00
P4	0.52	0.00
P5	1.33	0.00
ln(comments+1)	0.54	0.00
Resolver	-0.22	0.00
Late	-0.72	0.00

	Estimate	p-value
(Intercept)	-2.23	0.02
Critical	0.28	0.01
Major	0.43	0.00
Normal	0.60	0.00
Minor	0.75	0.00
Trivial	0.75	0.00
Enhancement	1.12	0.00
ln(attachments+1)	0.12	0.00
ln(depends+1)	0.41	0.00
ln(assignee+1)	0.45	0.00
Reputation	-0.52	0.00
P1	-0.09	0.05
P2	0.20	0.00
P3	0.43	0.00
P4	0.49	0.00
P5	0.85	0.00
ln(comments+1)	1.08	0.00
Resolver	-0.21	0.00
Late	-0.20	0.00

Correction of data makes a substantial difference.

Generalization

Generalization



Generalization

Exceptionally “Productive” Individuals
(Based on Issue Report Events)

Date	User ID	Count
2012-10-01	452624	542
1999-11-22	4415	277
2011-06-24	12809	116
2009-12-16	24572	110
2012-01-27	148348	93
2012-10-12	384312	90
2011-12-14	24572	87
2010-10-13	164048	87
2012-06-01	24572	86
2000-07-08	41	86

Exceptionally “Productive” Individuals
(Based on Code Commit Events)

Date	User ID	Count
2013-03-21	Bobby Holley	1160
2013-08-22	Ms2ger	1029
2013-02-25	Gregory Szorc	1024
2014-01-27	B2G Bumper Bot	998
2012-08-04	Ms2ger	991
2013-07-24	Ms2ger	986
2013-01-08	ffxbld	981
2011-07-21	ffxbld	964
2013-08-06	ffxbld	945
2013-02-20	ffxbld	907

Thank you!