Mining GitHub for fun and profit

Georgios Gousios // **@gousiosg**
TU Delft

GitHub

- Project forge (Git, Issues)
- Project wiki (Git based)
- Social network (star repository, follow developer)
- Forks and Pull requests
- Comments on commits, issues and pull requests

GitHub is cool because

- api.github.com
- process and product data
- interconnection

Mining the GitHub API:

Only current state

Missing data schema

5k reqs/hr

GHTorrent: a scalable, queriable, offline mirror of GitHub data

api.github.com

Entities

- static view
- interlinked
- current state

Events

- dynamic view
- generated by user actions
- affect current entity state
- can be browsing roots

Events

CreateEvent

PushEvent

WatchEvent

.
.
.

ForkEvent

```
"type": "WatchEvent",
"payload": {...},
"public": true,
"repo": {...},
"created_at": "2012-05-28T12:42:30Z"
"id": "1556481024",
"actor": {"login": "Sarukhan"}
```

Entities

users /users/:user

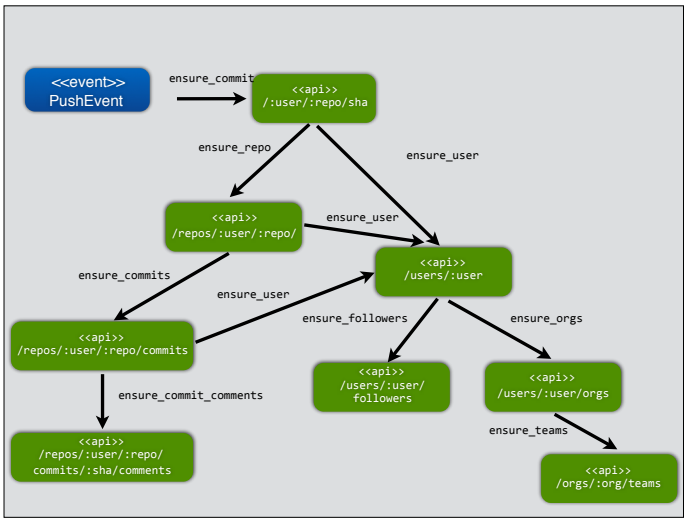
repositories /user/repos

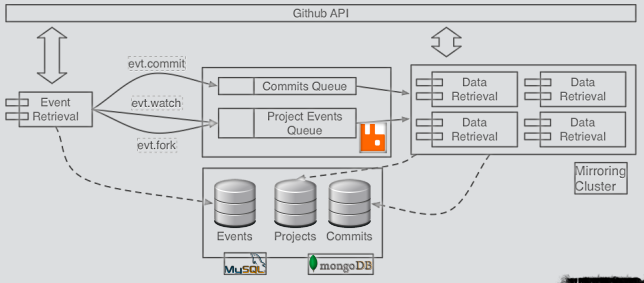
organizations /orgs/:org

.
.
.

issues /repos/:user/:repo/issues

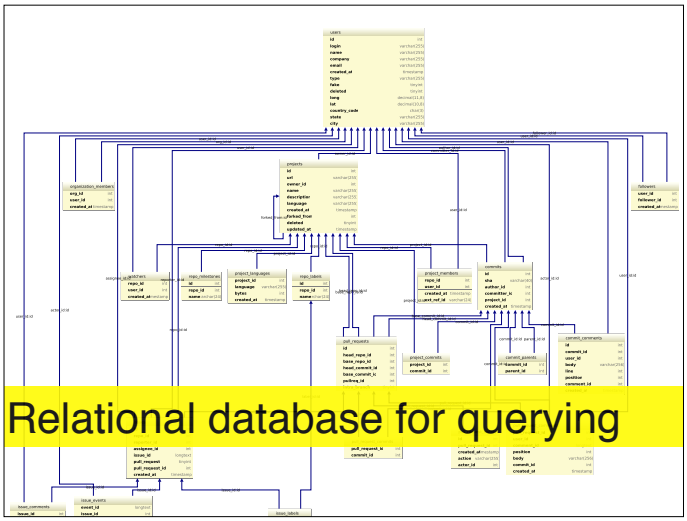
```
{
  "type": "User",
  "public_gists": 10,
  "login": "gousiosg",
  "followers": 64,
  "name": "Georgios Gousios",
  "public_repos": 20,
  "created_at": ...,
  "id": 386172,
  "following": 16,
}
```





Architecture

Demo!



Relational database for querying

users /users/:user

repositories /user/repos

organizations /orgs/:org

.
.
.

issues /repos/:user/:repo/issues

```
{
  "type": "User",
  "public_gists": 10,
  "login": "gousiosg",
  "followers": 64,
  "name": "Georgios Gousios",
  "public_repos": 20,
  "created_at": ...,
  "id": 386172,
  "following": 16,
}
```

MongoDB as query-able cache

mongoDB

```
select country_code, count(*)
    from users
    where fake is false
        and deleted is false
        and country_code is not null
    group by country_code
    order by count(*) desc
```

Conn: rs0 | DB: ghtorrent | Table Structure | History | 1. R

Export | Show SQL | Refresh | Auto Refresh

country_code	count(*)
us	283211
gb	57550
cn	50819
in	46204
de	45473
fr	33003
ru	24441

User geolocation

```
select u.login, p.name, p.language, pl.language as main_language,
    pl.bytes, pl.created_at as date_measured
    from project_languages pl, projects p, users u
    where u.id = p.owner_id
    and pl.project_id = p.id
    and p.name = 'scala'
    and u.login = 'scala';
```

Connection | DB: ghtorrent | Table: project_langu | History | 1. Result

Export | Show SQL | Refresh | Auto Refresh

login	name	language	main_language	bytes	date_measured
scala	scala	Scala	scala	15934585	2015-10-29 10:46:57
scala	scala	Scala	java	1023702	2015-10-29 10:46:57
scala	scala	Scala	javascript	241109	2015-10-29 10:46:57
scala	scala	Scala	shell	57442	2015-10-29 10:46:57
scala	scala	Scala	python	44027	2015-10-29 10:46:57
scala	scala	Scala	css	37216	2015-10-29 10:46:57
scala	scala	Scala	html	6396	2015-10-29 10:46:57
scala	scala	Scala	c	141	2015-10-29 10:46:57

Full language retrieval

Operation modes

- **Normal operation:** Follow the event timeline, apply dependency-based retrieval
- **Bi-monthly updates:** Refresh the state of all repos/users (cater for deleted repos, changed user locations etc)
- **Full retrievals:** Get all info for a repo/user, in case stuff is missing

Open from the beginning

MySQL database dumps

New! As of MySQL dump `mysql-2015-09-25`, we are distributing CSV files (one file per table) instead of `mysqldump` based backups. The provided ar
directory including a restore script and instructions on how to do the restore. See more information here.

You can also query MySQL. It is always loaded with the latest dump.

- 2016-02-01 (34969 MB)
- 2016-01-16 (34178 MB)
- 2016-01-08 (33942 MB)
- 2015-09-25 (32273 MB)
- 2015-08-07 (31537 MB)
- 2015-06-18 (33476 MB)
- 2015-04-01 (25075 MB)
- 2015-01-04 (16583 MB)
- 2014-11-10 (14418 MB)
- 2014-08-18 (11485 MB)
- 2014-04-02 (7013 MB)
- 2014-01-02 (5646 MB)
- 2013-10-12 (4312 MB)

MongoDB database dumps

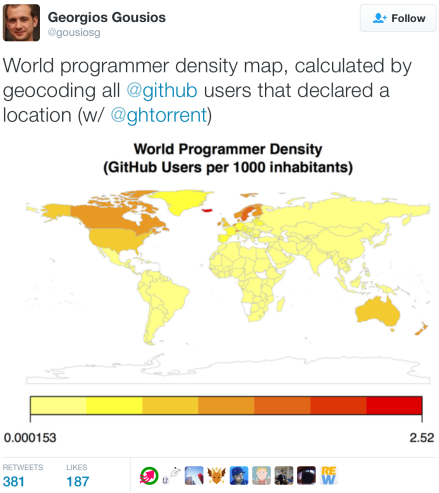
The MongoDB backups come in different formats:

- Bi-montly and by collection, from August 2015 up to Nov 2015
- Daily from 2015-12-01, includes all collections

To restore the full database you need to download, extract and restore all of them.

2016-02-08, 2016-02-07, 2016-02-06, 2016-02-05, 2016-02-04, 2016-02-02, 2016-02-01, 2016-01-31, 2016-01-30, 2016-01-29, 2016-01-28, 2016-01-27, 201
01-24, 2016-01-23, 2016-01-22, 2016-01-21, 2016-01-20, 2016-01-19, 2016-01-18, 2016-01-17, 2016-01-16, 2016-01-15, 2016-01-14, 2016-01-13, 2016-01-
2016-01-09, 2016-01-08, 2016-01-07, 2016-01-06, 2016-01-05, 2016-01-04, 2016-01-03, 2016-01-02, 2016-01-01, 2015-12-31, 2015-12-30, 2015-12-29, 201
12-26, 2015-12-25, 2015-12-24, 2015-12-23, 2015-12-22, 2015-12-21, 2015-12-20, 2015-12-19, 2015-12-18, 2015-12-17, 2015-12-16, 2015-12-15, 2015-12-
2015-12-09, 2015-12-08, 2015-12-07, 2015-12-06, 2015-12-05, 2015-12-04, 2015-12-03, 2015-12-02, 2015-12-01,

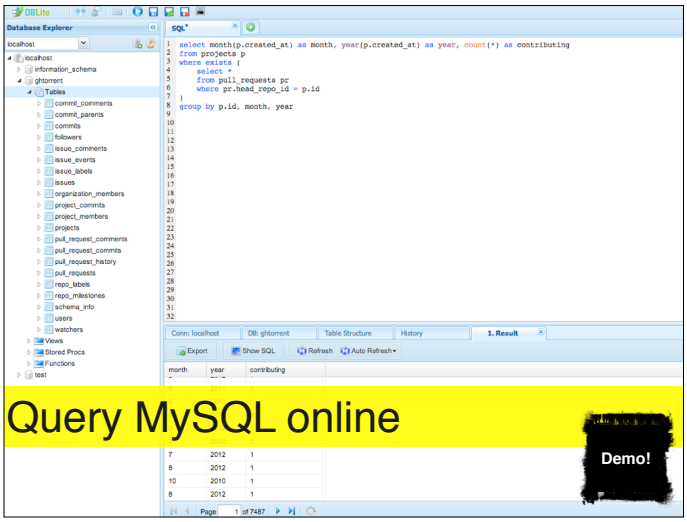
Periodic dumps of DBs

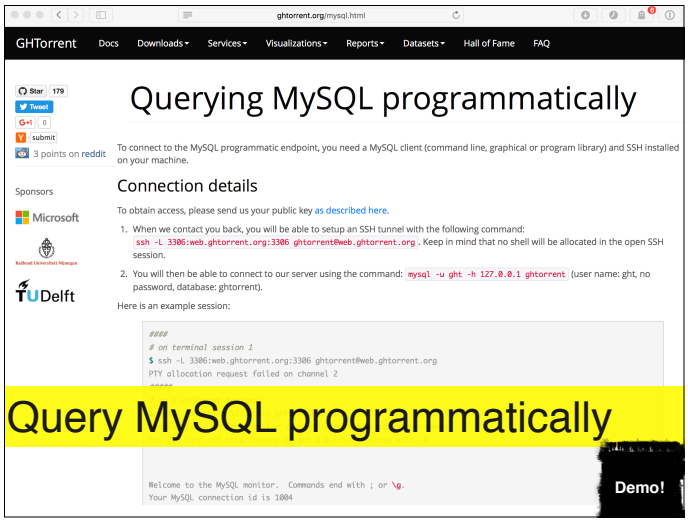


Georgios Gousios @gousiosg — Follow

World programmer density map, calculated by geocoding all @github users that declared a location (w/ @ghtorrent)

RETWEETS 381 LIKES 187





Latest commits

```
select c.*
from commits c, project_commits pc, projects p, users u
where u.login = 'rails'
  and p.name = 'rails'
  and p.id = pc.project_id
  and c.id = pc.commit_id
order by c.created_at desc
limit 20
```

id	sha	author_id	committer_id	project_id	created_at
489407993	f41ae3718ee2ed955ed89b64fd8d4194653c32bd	980867	980867	31997534	2016-08-10 22:48:50
489520659	d61aec316ab1b884f7d95621b4c6057c45830f4f	254676	254676	1464573	2016-08-10 20:25:45
489311017	780a059ac49820b0fb263247a2ac3ce9ffd3b2ebf	254676	30351173	1464573	2016-08-10 20:23:23
489311019	b1b64f86fdaaf4b2022392ed0639a5962a2de4f	254676	254676	1464573	2016-08-10 20:22:32
489307351	94ace8eea9da183aee38cec861b3c5dd0ba22997	51609	51609	9675340	2016-08-10 20:07:34
489202810	1aef0f11627598dc12bb8d7a1a100899c13beb57	476881	476881	42855797	2016-08-10 17:39:22
489202808	a16237fc66788e46b459a8f01a8603a757aff11bf	64341	30351173	1334	2016-08-10 17:03:22
489099710	c82417e2549490790a99fd5b176944816475 8be1	11028754	11028754	42624164	2016-08-10 15:03:38
489095985	c60e1586283119e0079bbf76af1dda2de9903581	3916600	3916600	43143623	2016-08-10 15:02:33
489047938	00f457b0b1c589b9ddc0bbad01062a0703432686	1127496	1127496	10330522	2016-08-10 13:54:12
489047936	dfb246118afbc6b59f6517eaed00208aa6c7e4b5	45381	30351173	1334	2016-08-10 13:01:40
489047936	dfb246118afbc6b59f6517eaed00208aa6c7e4b5	45381	30351173	1334	2016-08-10 13:01:40
488906168	da3827f7f15ec145064517d8fd622022afe424538	36609	36609	11349496	2016-08-10 10:04:22
488795567	11cbe12aebc09053b06b93e6fe53062f424674a0	8941	8941	1334	2016-08-10 07:45:54
488795571	01fbdb311d0661b4db89024e5b1c9fafacaceaafd	5394	5394	3904583	2016-08-10 07:44:08
489447437	16a958f88693ff551674f96801fbe9343b55f483	30437559	30437559	43190079	2016-08-10 07:27:42
489447438	07d02a79d9a5cd59816c5a7ac8bc29b599029bdb	30437559	30437559	43190079	2016-08-10 06:41:56
489447441	9180caa329411e3b131a259fd268602d344f7a2743	30437559	30437559	43190079	2016-08-10 06:01:08
488653339	8b984161d62d2a5167b3a94e2ff d9873b52ee0f6	64341	64341	1045327	2016-08-10 04:02:22
488453463	04870606b630733dc8df1e180cf0ede8185c1946	143208	143208	6512881	2016-08-09 23:11:47

Ruby on Rails

```
select year(w.created_at) as year,
       count(*) as num_starts
from watchers w
where w.repo_id = 1334
group by year(w.created_at);
```

year	num_starts
2008	3779
2009	3125
2010	3709
2011	3909
2012	6486
2013	1207
2014	3135
2015	5300
2016	4083

9 rows in set (0.27 sec)

Django

```
select year(w.created_at) as year,
       count(*) as num_starts
from watchers w
where w.repo_id = 1126
group by year(w.created_at);
```

year	num_starts
2012	9044
2013	1343
2014	3146
2015	5466
2016	3880

5 rows in set (0.22 sec)

Project forks: Total & Contributing per month

```
select last_day(p.created_at) as month,
       count(*) as created
    from projects p
    where p.forked_from = (
      select p.id
      from projects p
      where p.id = 1334)
    group by month
```

```
select last_day(p.created_at) as month,
       count(*) as contributing
    from projects p
    where p.forked_from = (
      select p.id
      from projects p
      where p.id = 1334)
    and exists (
      select *
      from pull_requests pr
      where pr.head_repo_id = p.id)
    group by month
```

month	created
2008-02-29	1
2008-04-30	19
2008-05-31	21
2008-06-30	19
2008-07-31	14
2008-08-31	14
2008-09-30	17
2008-10-31	20
2008-11-30	16
2008-12-31	19
2009-01-31	22
2009-02-28	26
2009-04-30	18
2009-05-31	29
2009-06-30	34
2009-07-31	24
2009-08-31	21
2009-09-30	27
2009-10-31	19
2009-11-30	24
2009-12-31	25
2010-01-31	17
2010-02-28	22
2010-03-31	23
[…]	
2015-07-31	83
2015-08-31	72
2015-09-30	65
2015-10-31	99
2015-11-30	63
2015-12-31	76
2016-01-31	88
2016-02-29	75
2016-03-31	59
2016-04-30	71
2016-05-31	98
2016-06-30	251
2016-07-31	275
2016-08-31	77

102 rows in set (0.19 sec)

month	contributing
2008-02-29	1
2008-04-30	7
2008-05-31	1
2008-06-30	1
2008-07-31	2
2008-08-31	1
2008-09-30	1
2008-10-31	1
2008-11-30	1
2008-12-31	1
2009-01-31	2
2009-02-28	4
2009-04-30	1
2009-05-31	2
2009-06-30	1
2009-07-31	1
2009-08-31	3
2009-09-30	2
2009-10-31	2
2009-11-30	2
2009-12-31	6
2010-01-31	4
2010-02-28	4
2010-03-31	2
[…]	
2015-07-31	51
2015-08-31	42
2015-09-30	45
2015-10-31	52
2015-11-30	45
2015-12-31	53
2016-01-31	60
2016-02-29	53
2016-03-31	33
2016-04-30	46
2016-05-31	45
2016-06-30	17
2016-07-31	20
2016-08-31	7

101 rows in set (26.35 sec)

Core team members last 3 months

```
select distinct(u.login) as login
    from commits c, users u, project_commits pc,
         users u1, projects p
    where u.id = c.committer_id
      and u.fake is false
      and pc.commit_id = c.id
      and pc.project_id = p.id
      and p.owner_id = u1.id
      and p.name = 'rails'
      and u1.login = 'rails'
      and c.created_at > DATE_SUB(NOW(), INTERVAL 3 MONTH)
union
select distinct(u.login) as login
  from pull_requests pr, projects p, users u,
       users u1, pull_request_history prh
  where u.id = prh.actor_id
    and prh.action = 'merged'
    and u1.id = p.owner_id
    and prh.pull_request_id = pr.id
    and pr.base_repo_id = p.id
    and prh.created_at > DATE_SUB(NOW(), INTERVAL 3 MONTH)
    and p.name = 'rails'
    and u1.login = 'rails'
```

login
sgrif
akinomaeni
Neodelf
schneems
maclover7
kangkyu
y-yagi
jahfer
joemasilotti
eileencodes
rafaelfranca
lv
pixeltrix
[…]
fidelisrafael
chumakoff
darkbaby123
lorint
yevhene
mrageh
mubashirhanif
travisoneill
kerrizor
guillelguaran

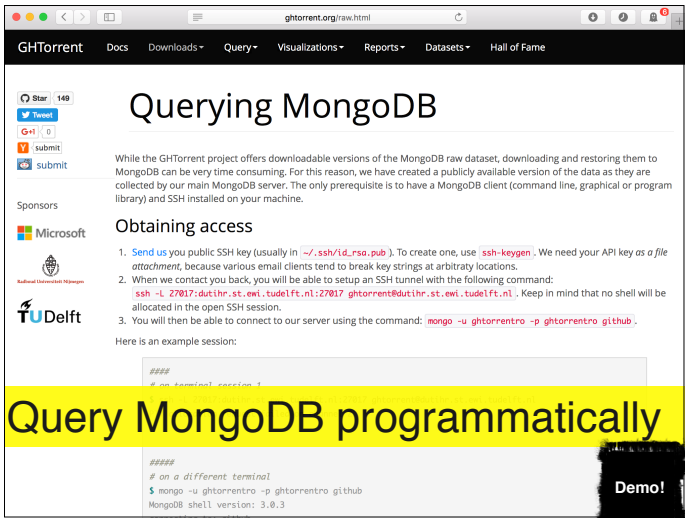
141 rows in set (1.44 sec)

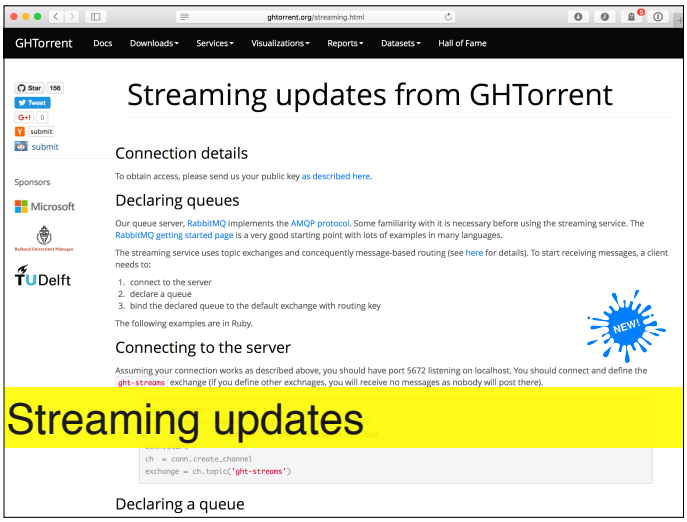
Python developers in Estonia

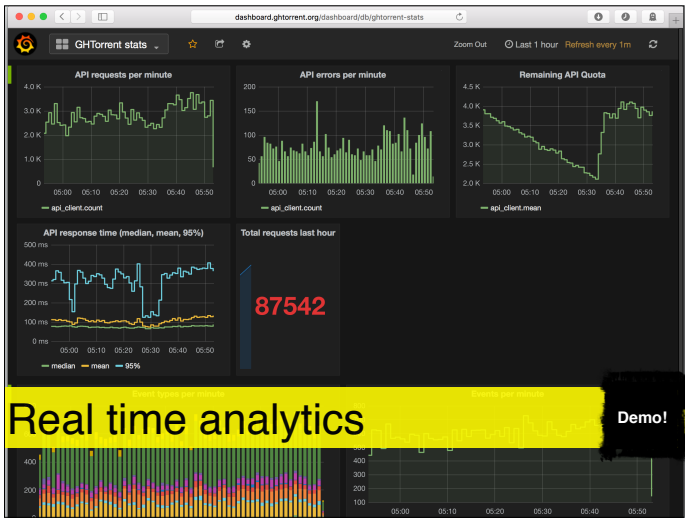
```
select distinct(u.login), u.location
from users u, commits c,
     projects p, project_commits pc
where date(c.created_at) between date('2016-06-01')
                             and date('2016-07-01')
  and pc.commit_id = c.id
  and p.id = pc.project_id
  and c.author_id = u.id
  and u.country_code = 'ee'
  and p.language = 'Python';
```

login	location
plaes	Estonia
sim642	Estonia
rival	Tallinn, Estonia
livenson	Estonia
mittya	Moon
ribozz	Tallinn
emddom	tallinn
argoroots	Estonia
Newbiz	Tallinn, Estonia
mitselek	Tallinn, Estonia
asoltsev	Tallinn, Estonia
3yrno42	Estonia, Tallinn
hudolejev	Estonia
Villane	Estonia
syko	Estonia
raidoz	Estonia
vruusmann	Estonia
amPerl	Estonia
lsaffre	Estonia
mikk351	Tallinn
Sunnepah	Estonia
ropod7	Estonia
siims	Estonia
jaakerisalu	Tallinn, Estonia
kendas	Estonia
artizirk	Estonia
avalikarvamus	Tallinn, Estonia

27 rows in set (0.00 sec)



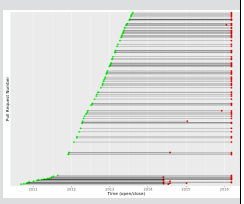


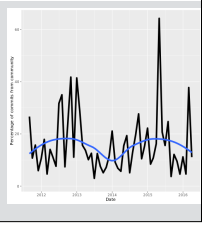


Openess reports

How open is your project to community contributions?

- 5k projects
- every 15 days





Demo! http://ghtorrent.org/pullreq-perf

```
$ gem install sqlite3 bundler
$ git clone https://github.com/gousiosg/github-mirror
$ cd github-mirror
$ bundle install
$ mv config.yaml.standalone > config.yaml
$ ruby -Ilib bin/ght-retrieve-repo -t token rails rails
```

Roll your own dataset

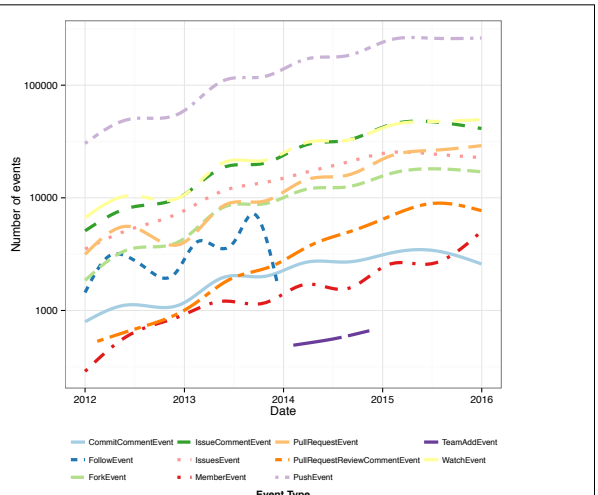

```
$ apt-get install vagrant
$ git clone https://github.com/ghtorrent/ghtorrent-vagrant.git
$ cd ghtorrent-vagrant
$ vagrant ssh
$ ruby -Ilib bin/ght-retrieve-repo -t token rails rails
```

Even simpler with Vagrant

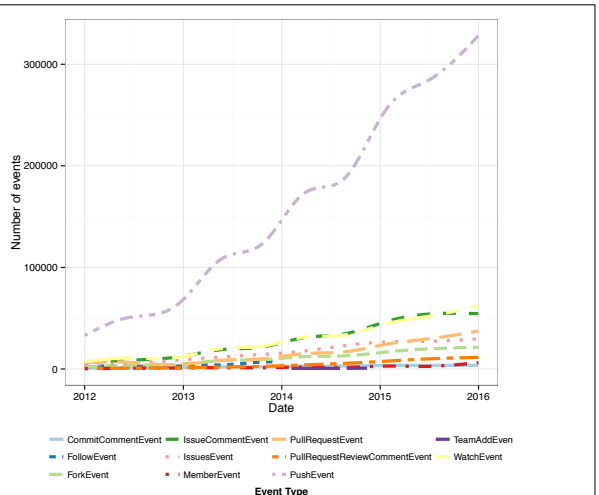
Growth

	MongoDB	MySQL	Diff 2016/2013
Events	476		**11.1x**
Users	6,7	9,2	**8.4x**
Repos	28	25,5	**21.8x**
Commits	367	362	**12.3x**
Issues	24,1	25,3	**10.3x**
Pull requests	11,9	11,1	**9.7x**
Issue comments	42	43	**14.6x**
Watchers	51	37	**6.6x**

G. Gousios, *The Evolution of GHTorrent: Growing an Open Access Dataset 10x*



G. Gousios, *The Evolution of GHTorrent: Growing an Open Access Dataset 10x*



G. Gousios, *The Evolution of GHTorrent: Growing an Open Access Dataset 10x*

Stats and Impact

Since Feb 2012

12TB in MongoDB

4.5B rows in MySQL

2GB per hour

170k API reqs/hour

50 user donated API keys

180+ users, **100+** institutions

100+ papers

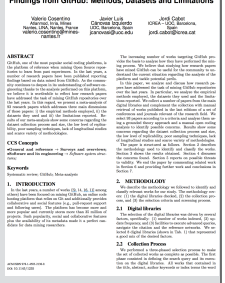
3 data mining challenges

3 best paper awards

G. Gousios, "The GHTorrent dataset and tool suite," in MSR, 2013: 233-236
G. Gousios and D. Spinellis, " GHTorrent: GitHub's Data from a Firehose," in MSR, 2012, 12–21

Impact

MSR 2016: *35% of all papers on Github are done with GHTorrent*



Findings from GitHub: Methods, Datasets and Limitations

Valerio Cosentino, Javier Luis, Jordi Cabot

Valerio Cosentino
Nantes, LINA, Nantes, France
valerio.cosentino@inria-rainbow.fr

Javier Luis
GESSI, UPC-Barcelona Tech
UOC, Barcelona, Spain
jcanovas@uoc.edu

Jordi Cabot
ICREA - UOC - Barcelona, Spain
jordi.cabot@icrea.cat

ABSTRACT

CCS Concepts

Keywords

1. INTRODUCTION

2. METHODOLOGY

2.1 Digital libraries

2.2 Collection Process

Valerio Cosentino, Javier Luis, and Jordi Cabot. 2016. *Findings from GitHub: methods, datasets and limitations.* MSR '16, pp 137-141

MSR 2014 challenge

- Pull requests
- Project developer attraction
- Sentiment analysis of commit comments
- Biodiversity of project ecosystems
- Project openess
- Design discussions
- Co-evolution of documentation and popularity

looking for include code (and instructions to use it) in a GitHub repository, an academic write-up of your analysis, or an informal prose write-up. If you're not linking to a repository, you should submit a Gist with your documentation.

After the submission deadline on August 25th, GitHub employees will review and vote on all entries to pick the three top winners. We'll send out notifications to those top three by mid-September.

Data Sources

GitHub activity data is available from several publicly-available sources. Here are a few links to get you started:

- Our very own API.
- The GitHub Archive, providing historical archives of our public timeline data.
- Google BigQuery, where GitHub's public timeline is a featured public dataset; see the GitHub Archive home page for getting started instructions.
- GHTorrent, which maintains a relational model of GitHub activity data and offers archives for download.

ProTips

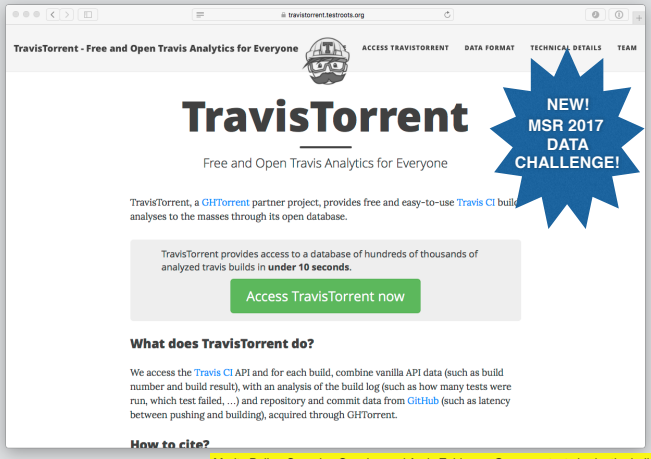
There are a few things we're looking for when we score your entry:

- **Innovation/Story:** Does your entry tell a good, data-driven story? Does it reveal interesting insights about GitHub activity? We love it when we're surprised by new insights hidden in our own data.
- **Accuracy:** Is your analysis accurate? Do accompanying visualizations clearly and unambiguously convey your conclusions?
- **Completeness:** Is your entry a code submission? If so, is your code well-organized and documented? Can others easily understand and reproduce your analysis from the materials you've submitted?

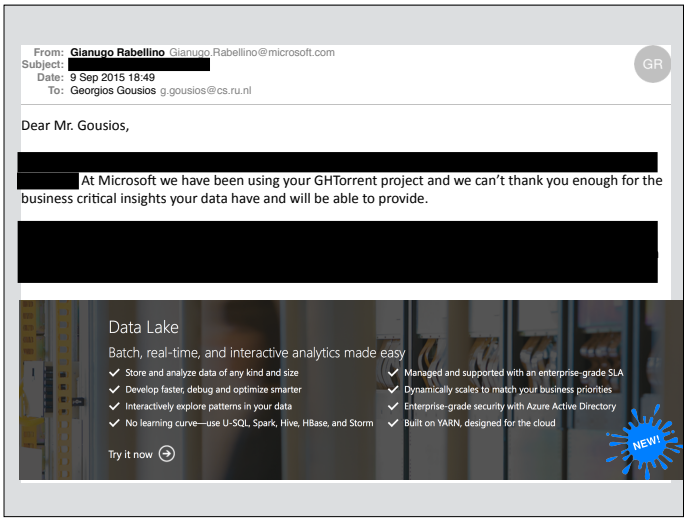
The Prizes

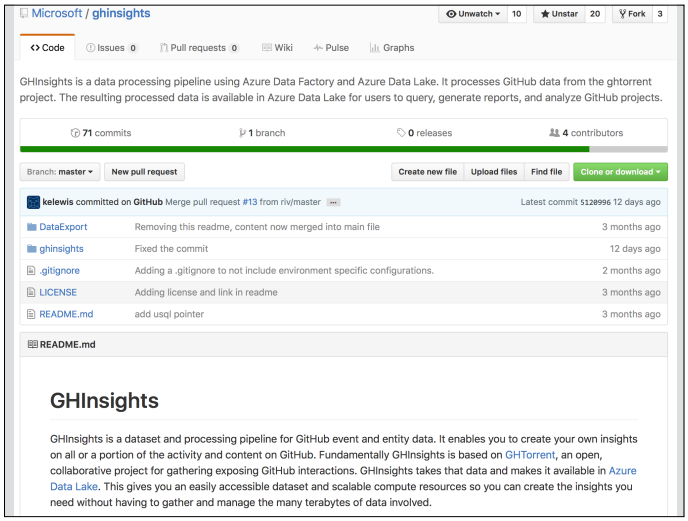
GitHub API rate limit and network latency. The mining program was able to check about one repository ID per second, which was slowed by network latencies, or, once 5000 API calls had been made in one hour, was throttled by GitHub. The apparent solution was to create multiple accounts, each providing 5000 API calls per hour. After contacting GitHub to request help with this issue, they indicated that they do not want users to create multiple accounts for mining projects because it can put a strain on their servers, slowing the service for regular users. An alternative strategy was proposed by GitHub of using ghtorrent[3] to find Python projects without using the API. However, at this point in the project, enough data had been acquired to begin analysis and a determination was made to stop development of this mining program and focus on analysis. Future mining efforts are encouraged to obtain repository information from this database instead of crawling through all projects using the GitHub API, like tour_de_source did.

Chapman, Carl Allen, *Usage and refactoring studies of python regular expressions.* Graduate Theses and Dissertations, Paper 15139. Iowa State University, 2016



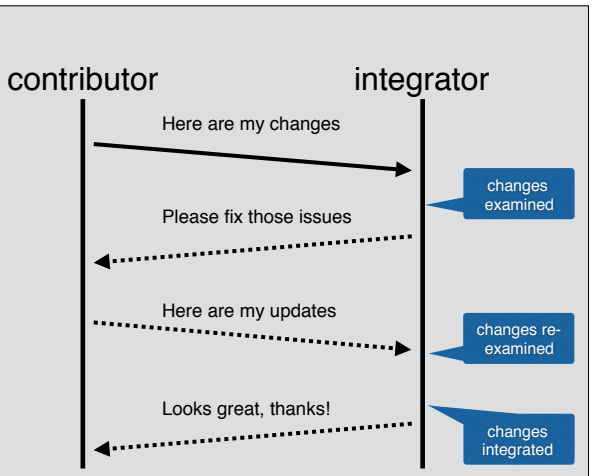
Moritz Beller, Georgios Gousios and Andy Zaidman. *Oops, my tests broke the build: An analysis of Travis CI builds with GitHub.* No. e1984v1. PeerJ Preprints, 2016.

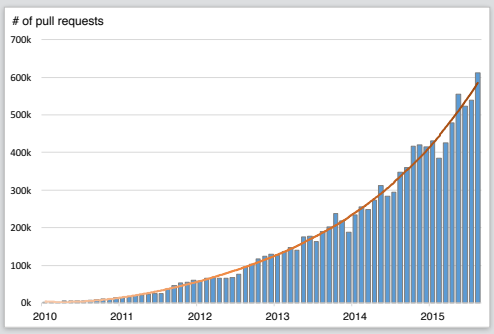




https://octodex.github.com

fork-edit pull request merge





projects with > 1 integrators
55% shared repository
45% use pull requests

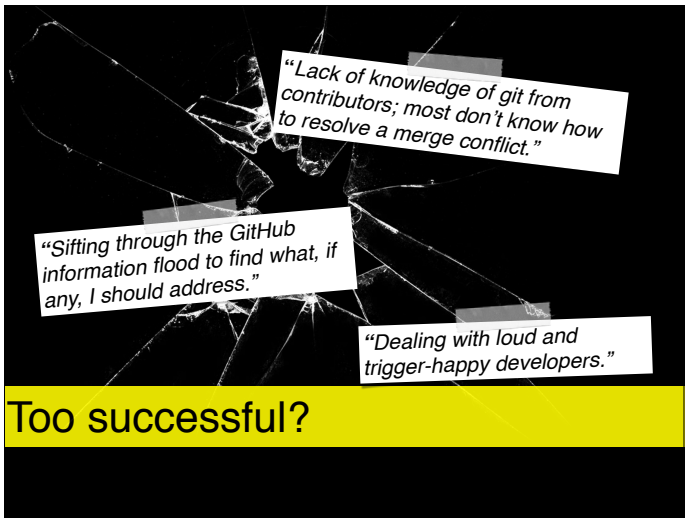
2016
135k repos/month
600k pull-requests/month

Large scale collaboration

Repository	Commits	Pull requests	Code reviews	Issues	Issue comments	Total
isaacs/npm	100	167	291	2568	3302	**6428**
torvalds/linux	5968	67	178	0	0	**6213**
symfony/symfony	1021	1261	1752	1844	2160	**8038**
jquery/jquery-mobile	212	431	384	2888	3008	**6923**
joyent/node	657	833	1127	2304	2805	**7726**
CocoaPods/Specs	2658	2584	1364	515	268	**7389**
gitlabhq/gitlabhq	605	871	1142			
angular/angular.js	875	1306	1751			
rails/rails	2699	2315	4090			
mxcl/homebrew	3426	3125	4492			

23,50

http://www.gousios.gr/blog/The-triumph-of-onlin



How does the PR process look like?

Across **5k** repos/ ~**1M** pull requests

85% merged, **70%** with merge button

80% < 150 lines, < 7 files, 3 commits

66% < 1 day to merge

80% 4 comments, 3 participants

Mostly rejected due to observability/awareness issues (not technical!)


An Exploratory Study of the Pull-based Software Development Model

G. Gousios, M. Pinzger, and A. van Deursen, "An Exploratory Study of the Pull-based Software Development Model," ICSE 2014, pp. 345–355

Which factors affect PR acceptance?

?

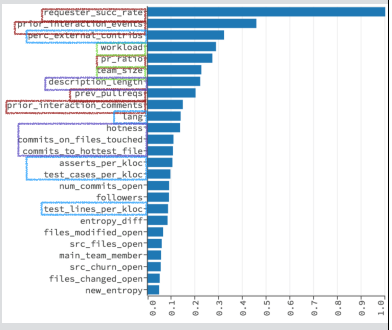
Which factors affect PR acceptance?

Do we know the submitter?

Can we handle the workload?

What does the PR look like?

How ready is our project for PRs?



G. Gousios et al., "Distributed software development with pull requests". Unpublished.

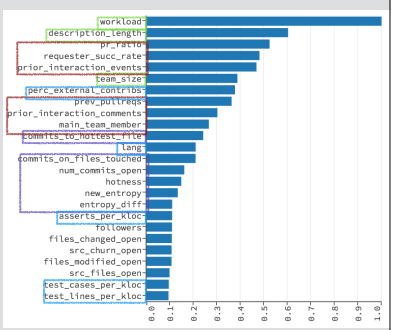
Which factors affect the time to process PRs?

Can we handle the workload?

Do we know the submitter?

How ready is our project for PRs?

What does the PR look like?



G. Gousios et al., "Distributed software development with pull requests". Unpublished.

What do ***integrators*** actually believe?

Survey of **650** integrators

Generally, few complains about the process

Points of pain are mostly social (workload, drive-by PRs, explaining rejection)

Needed tools

Quality analysis

Impact analysis

Work prioritization


Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective

G. Gousios, A. Zaidman, M.-A. Storey, and A. van Deursen, "Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective," ICSE 2015, pp. 358–368.

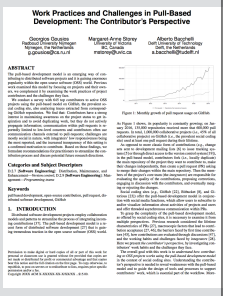
What do ***contributors*** actually believe?

Survey of **640** contributors.
Similar issues, reversed

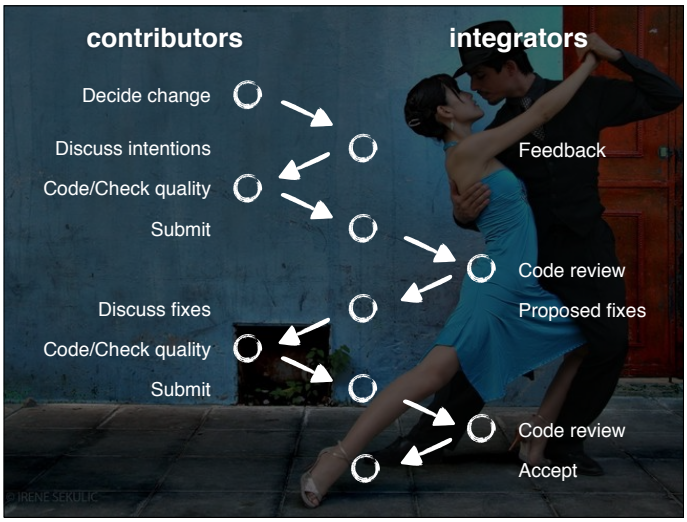
Awareness

Asynchrony

Responsiveness


Work Practices and Challenges in Pull-Based Development: The Contributor's Perspective

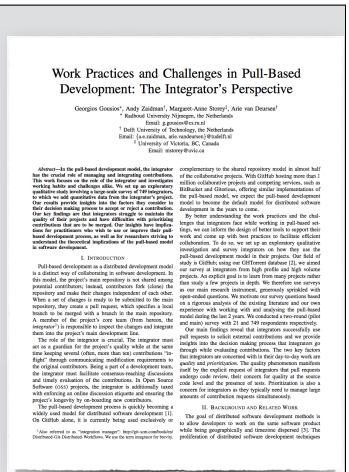
G. Gousios, M.-A. Storey, and A. Bacchelli, "Work Practices and Challenges in Pull-Based Development: The Contributor's Perspective," ICSE, 2016.



What are the integrators' biggest challenges?

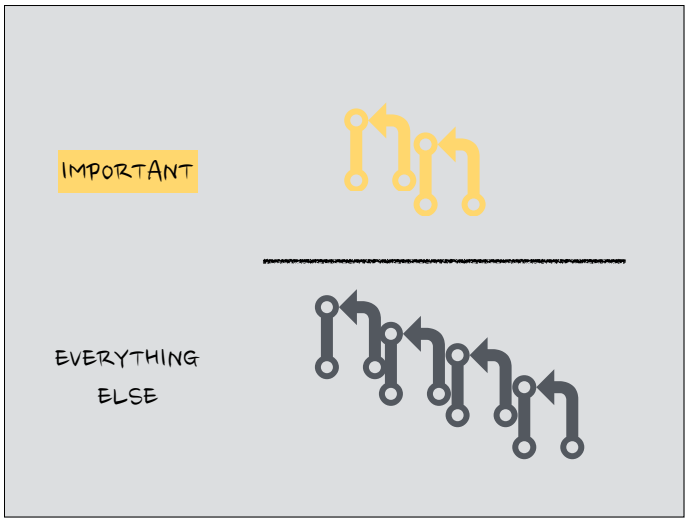
Quality

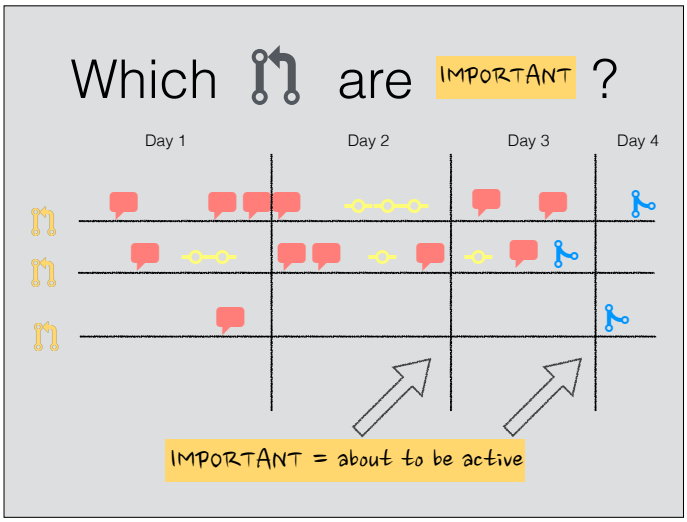
Prioritization

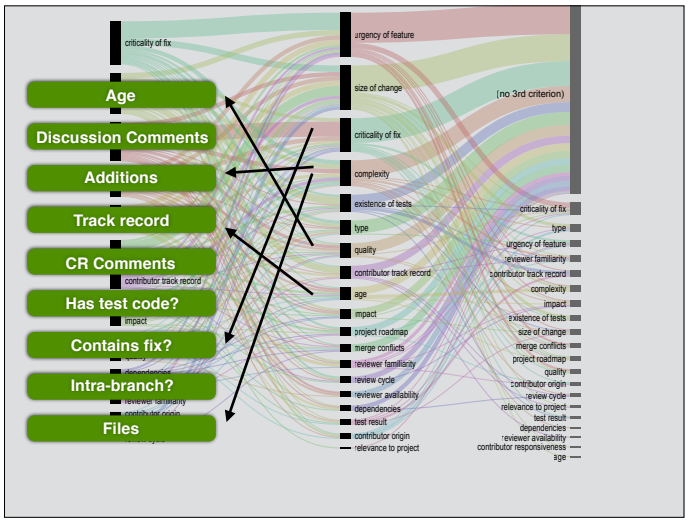

Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective

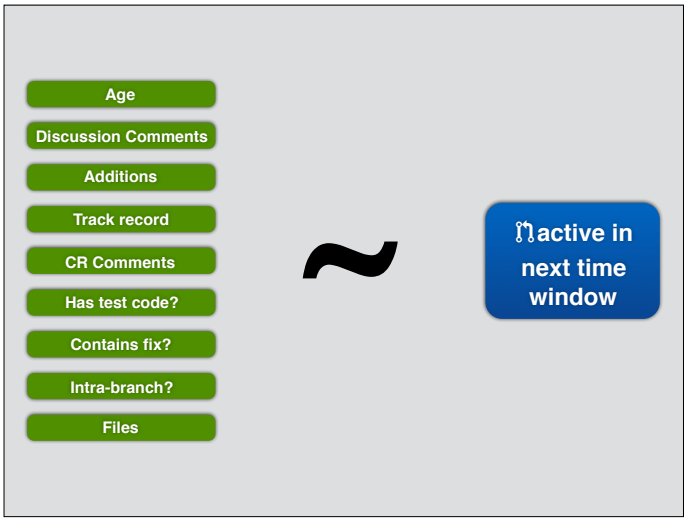
Characteristics of PRs

- They change fast (66% are processed within a day)
- Attention is only needed after an event occurred
- They are integrated with other tools



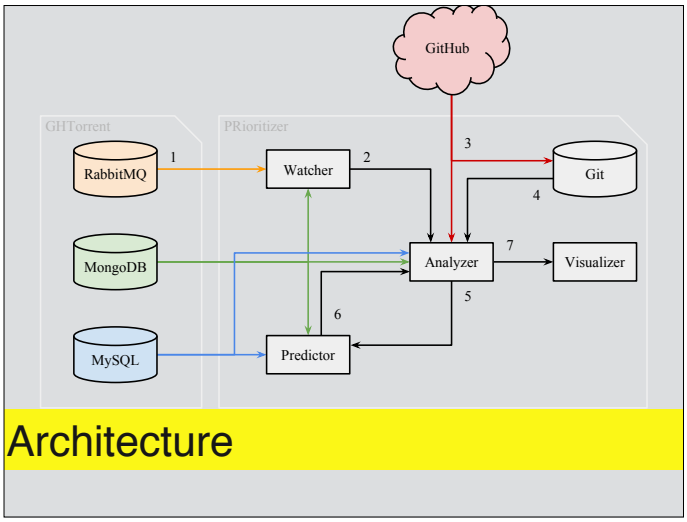




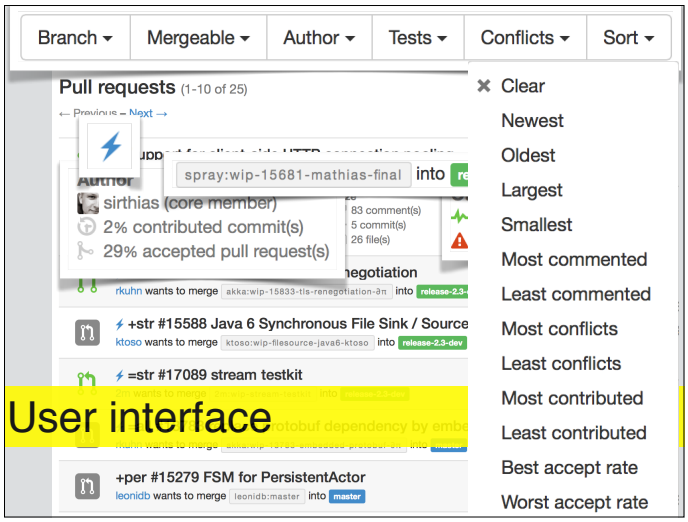


	Precision	Recall	AUC	Accuracy
Random Forests	0.66	0.63	**0.89**	**0.86**
Naive Bayes	0.34	0.79	0.75	0.60
Logistic regression	0.36	0.84	0.81	0.62

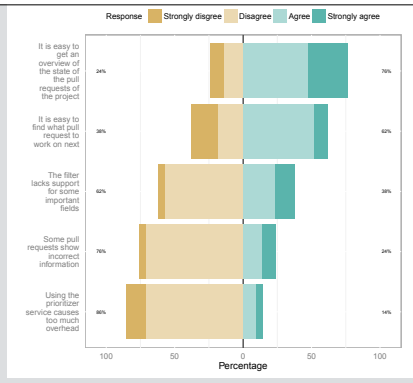
E. van der Veen, G. Gousios, and A. Zaidman, "Automatically Prioritizing Pull Requests," MSR, 2015, pp. 357–361.



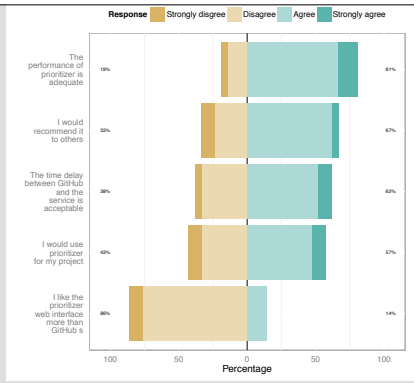
Architecture



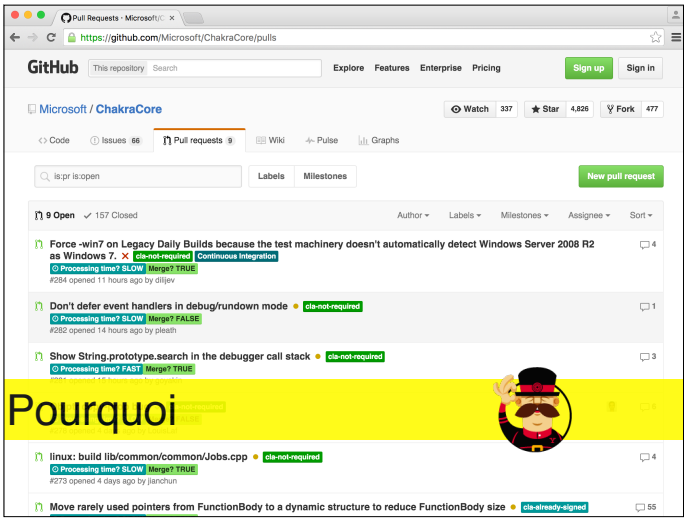
User interface



Prioritizer user evaluation



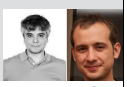
Prioritizer user evaluation



Pourquoi

Based upon…

Promises & Perils



How good is GitHub data for SE research?

- 5 promises
- 13 perils
- Ways to select projects for research
- Examined MSR 2014 papers, all susceptible to perils!

E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, *"An in-depth study of the promises and perils of mining GitHub."* Empirical Software Engineering, pp. 1–37, 2015

Issue lifetime predictions

Work dual to PRs

Need to train in short term time windows

Difficult to generalise across projects

Contextual features more important than generic ones

Riivo Kikas, Marlon Dumas, and Dietmar Pfahl. *Using dynamic and contextual features to predict issue lifetime in GitHub projects.* MSR '16, pp 291—302

Reviewer recommendation

Exploit *@mention* networks to propose top-3 reviewers for incoming pull requests.

Accuracy **~60%** on top-3 recommendation

Yue Yu, Huaimin Wang, Gang Yin, Tao Wang, *Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?*,IST, 2016

Automated code review

Examine how "natural" the PR code is WRT the project's code base.

Accepted PRs are significantly similar to the project

More debated PRs are significantly less similar

Vincent J. Hellendoorn, Premkumar T. Devanbu, and Alberto Bacchelli. 2015. *Will they like this?: evaluating code contributions with language models.* MSR '15, pp157-167

Not only OSS

"…GitHub's transparency and popular workflow can promote open collaboration, allowing organizations to increase code reuse and promote knowledge sharing across their teams."

Eirini Kalliamvakou, Daniela Damian, Kelly Blincoe, Leif Singer, and Daniel M. German. *Open source-style collaborative development practices in commercial projects using GitHub.* ICSE 2015, pp 574-585.

Gender and Tenure diversity

"Our study suggests that, overall, when forming or recruiting a software team, increased gender and tenure diversity are associated with greater productivity."

Vasilescu, Posnett, Ray, van den Brand, Serebrenik, Devanbu, and Filkov. *Gender and Tenure Diversity in GitHub Teams.* CHI. 2015

Geographical diversity

Traces of bias on contributions from certain countries

- Contributors perceive it
- Integrator's do not

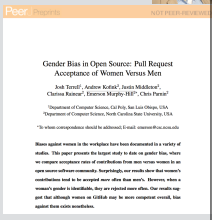

All Contributors are Equal; Some Contributors are More Equal than Others

A. Rastogi, N. Nagappan, and G. Gousios, "All contributors are equal; some contributors are more equal than others,", TR, 2016

Gender & Contributions

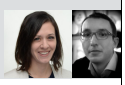
When gender is identifiable: women rejected more often

When gender is *not* identifiable: women accepted more often


Gender Bias in Open Source: Pull Request Acceptance of Women Versus Men

Terrell J, Kofink A, Middleton J, Rainear C, Murphy-Hill E, Parnin C. (2016) Gender bias in open source: Pull request acceptance of women versus men. *PeerJ PrePrints* 4:e1733v1

Productivity



Study how context-switching between repos affects productivity. If

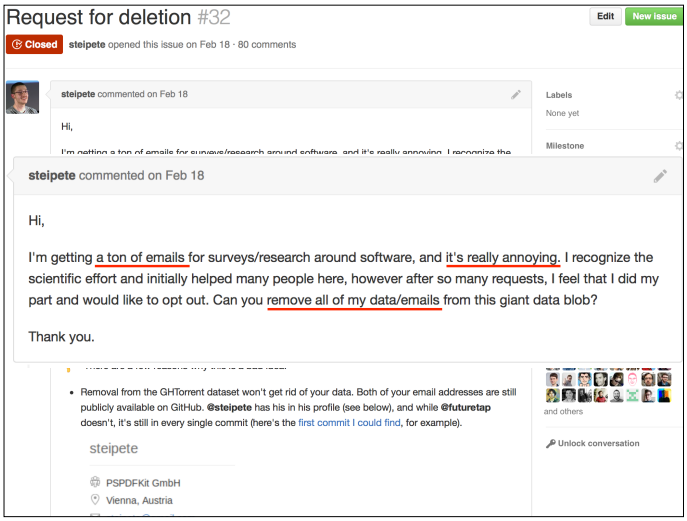
- rate of switching
- number of projects involved in

are high, productivity goes down


The Sky Is Not the Limit: Multitasking Across GitHub Projects

Bogdan Vasilescu, Kelly Blincoe, Qi Xuan, Casey Casalnuovo, Daniela Damian, Premkumar Devanbu, and Vladimir Filkov. *The sky is not the limit: multitasking across GitHub projects.* ICSE 2016, pp 99—1005

The #issue32 incident



I am not a lawyer!

- Other commenters are no lawyers either
- The law is complicated and open to interpretation

Two important issues

- *Copyright:* Who owns the data?
- *Privacy:* How does GHTorrent protect users from personal data misuse?

Copyright —General terms

- For *original content*, the *publisher* maintains full copyright by default
  - Licenses *restrict* the effect of copyright
- Events (e.g. the fact that an issue comment was created) are *not* copyrightable, but their content may be

Copyright — GitHub's POV

- GitHub: *We claim no intellectual property rights over the material you provide to the Service.* (TOS F.1)
- Structure of API responses is GitHub's IP
- Several fields in API responses may contain copyrighted material

Copyright situation example


```
{
    "id": "4141500869",
    "type": "IssueCommentEvent",
    "actor": {},
    "repo": {},
    "payload": {
      "action": "created",
      "issue": {
        "id": 158442053,
        "number": 138,
        "title": "Issue in CopyrightedProjectName",
        "user": {},
        "labels": [],
        "state": "closed",
        "body": "Added data holding classes and a
map manager. Will add a system soon"
      },
      "comment": {
        "created_at": "2016-06-14T05:51:16Z",
        "updated_at": "2016-06-14T05:51:16Z",
        "body": "continuing in #141 \r\n"
      }
    }
}
```


©GitHub

©Project Nar

©Issue initiat

©Issue cor

Privacy is the ability of an individual or group to seclude themselves, or information about themselves, and thereby express themselves selectively.

Privacy provisions — EU

- *Personal data* identify a person uniquely
  - *Facts* are not personal data
- GHTorrent processes personal data, therefore is a *controller*
- Controllers must
  - get consent for processing (except in the case of *legitimate interest*)
  - include mechanisms for opting out

Privacy provisions — USA

- No single law/directive
- Consent only required for specific types of data storage (e.g. social security numbers)
- Offering an opting out mechanism

What did GHTorrent do?

- Stopped distributing user names and emails in MySQL data dumps
  - Researchers can “sign” a form to get access to private data
- Created an opt-out process
- In the process of creating Terms of Fair Use

Research ethics

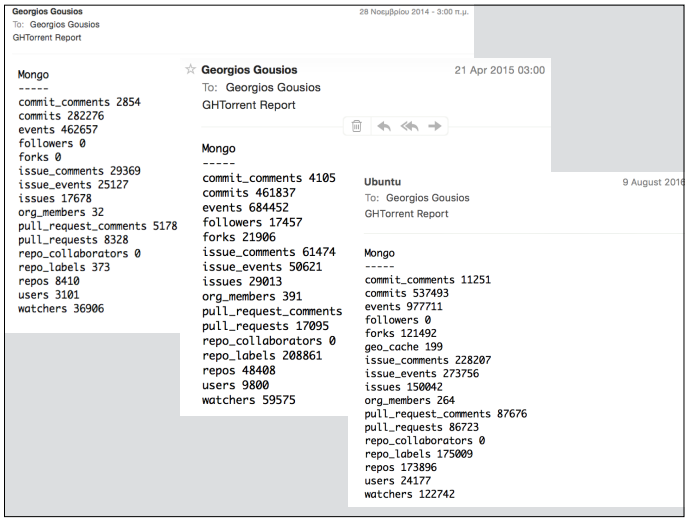
Can we, in the name of science,

- send emails to developers?
- create developer profiles?
- recommend work to developers?
- rank developers based on contributions?
- compare project characteristics?
- characterise community practices?

Where is GHTorrent going?

Challenges

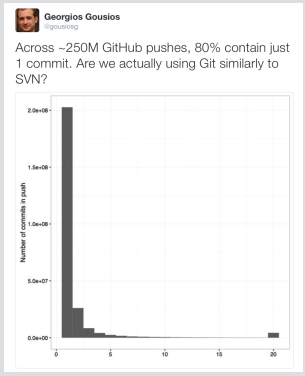
- 3Vs of big data: **V**olume, **V**elocity and **V**ariety
- Hardware not enough
- Software not optimal
- Services not always up an running



What can we do?

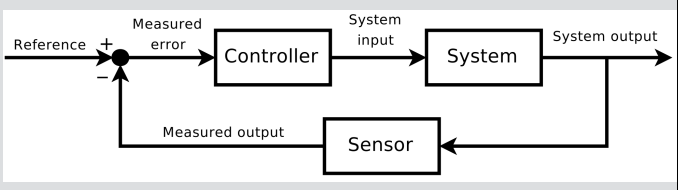
- Proper storage subsystem
  - MongoDB sharding? HDFS? Hyperdex?
- Proper querying infrastructure
  - Spark?
  - Datalake? <— Working with Microsoft
- Distributed operation and eventual consistency?

DataLake: sneak preview

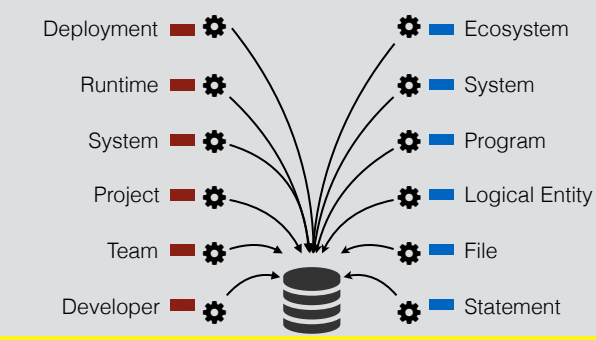


A dataset accessible to anybody who wants to do research on extremely interesting big data

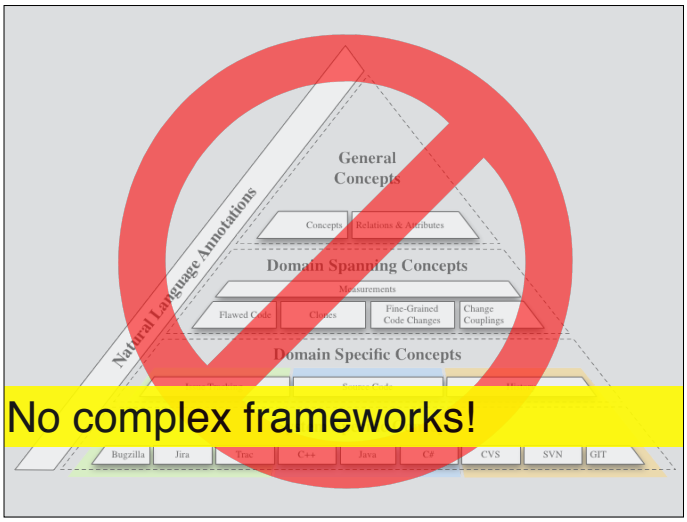
The future: Feedback Driven SE



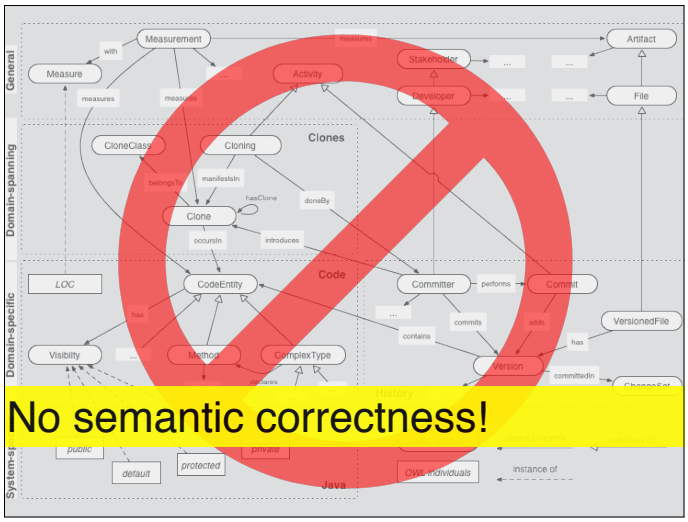
Feedback-driven development



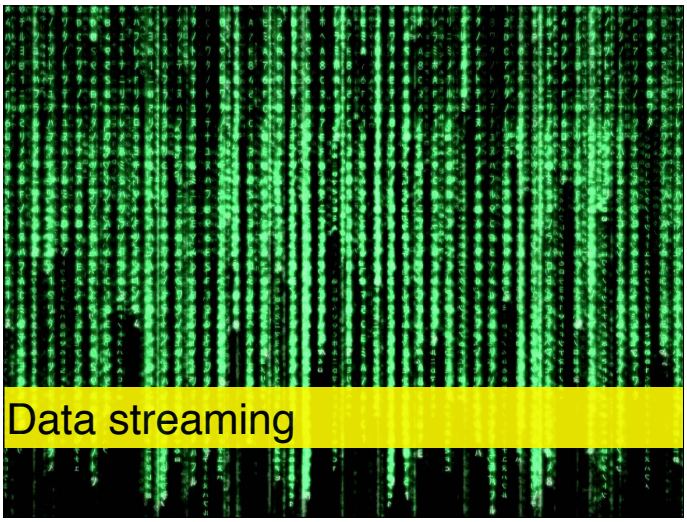
Data silos



No complex frameworks!



No semantic correctness!

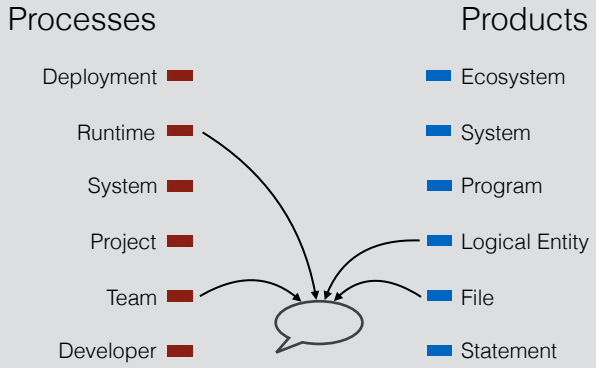


Data streaming

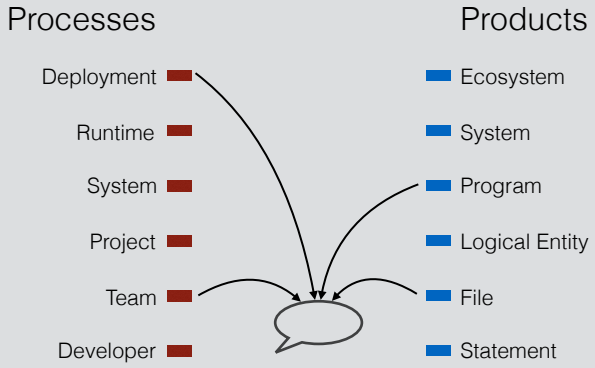
```
Observable.from(stacktraces)
  .flatMap{s => findDev(s) match {
      case Some(s) => Observable.just(s)
      case None => Observable.error(new Exception("Not found"))
  }}
  .groupBy(x => x)
  .map { case (dev, xs) =>
    (dev, xs.scan(0)((count, _) => count + 1))
  }
  .flatMap { case (dev, xs) =>
    xs.map(c => (dev, c))
  }
```

Queries on streams

How does my team test?



Did switching to agile work?



How is my incident respon