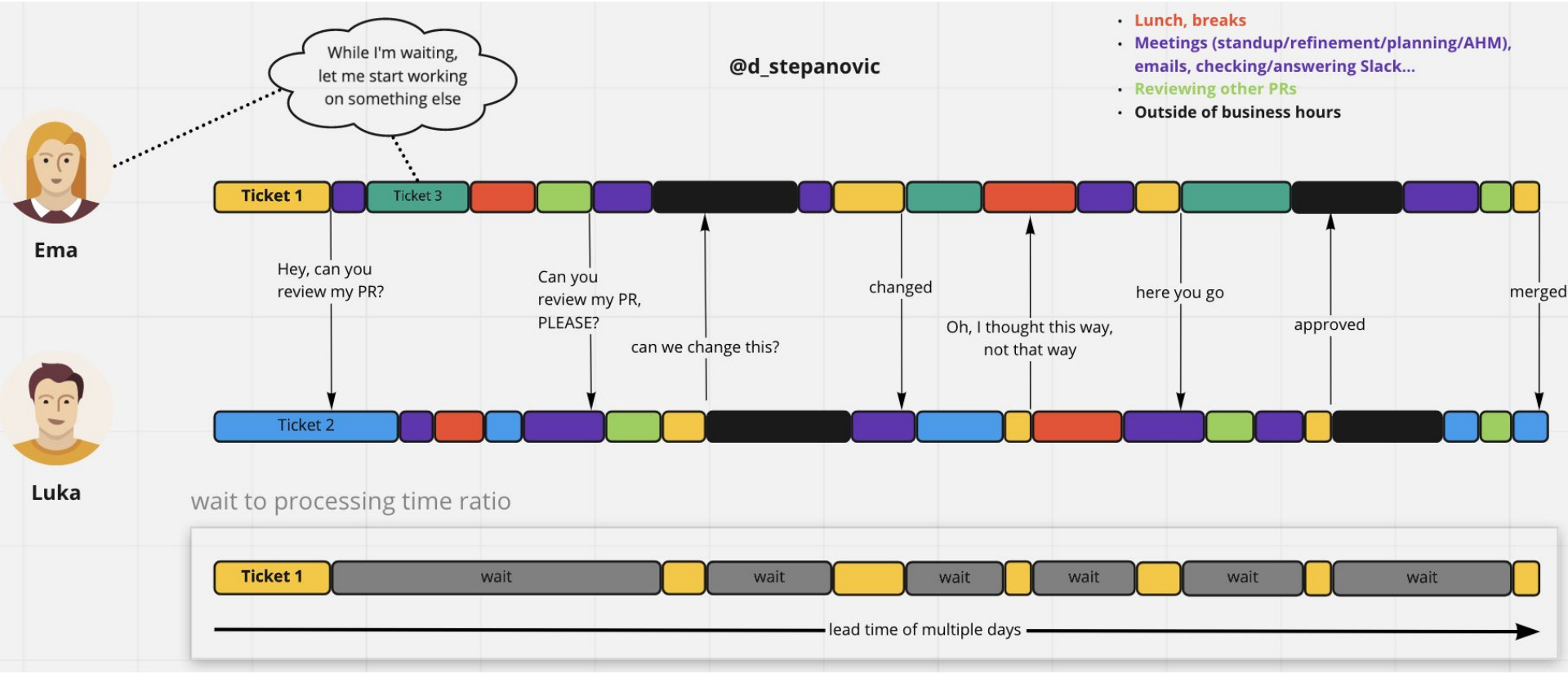
Async Code Reviews Are Killing Your Company’s Throughput

- Ex: Principal @Careem/Uber, HelloFresh, GetYourGuide
- XP, ToC, Lean, Systems Thinking
- Rants on
  - *draganstepanovic.com*
  - *@d_stepanovic*
- I put mayo on 🍕 😱

justsharing.dev


Nobody ever got fired for
buying IBM.

PR-based async code review



Meet people where they are

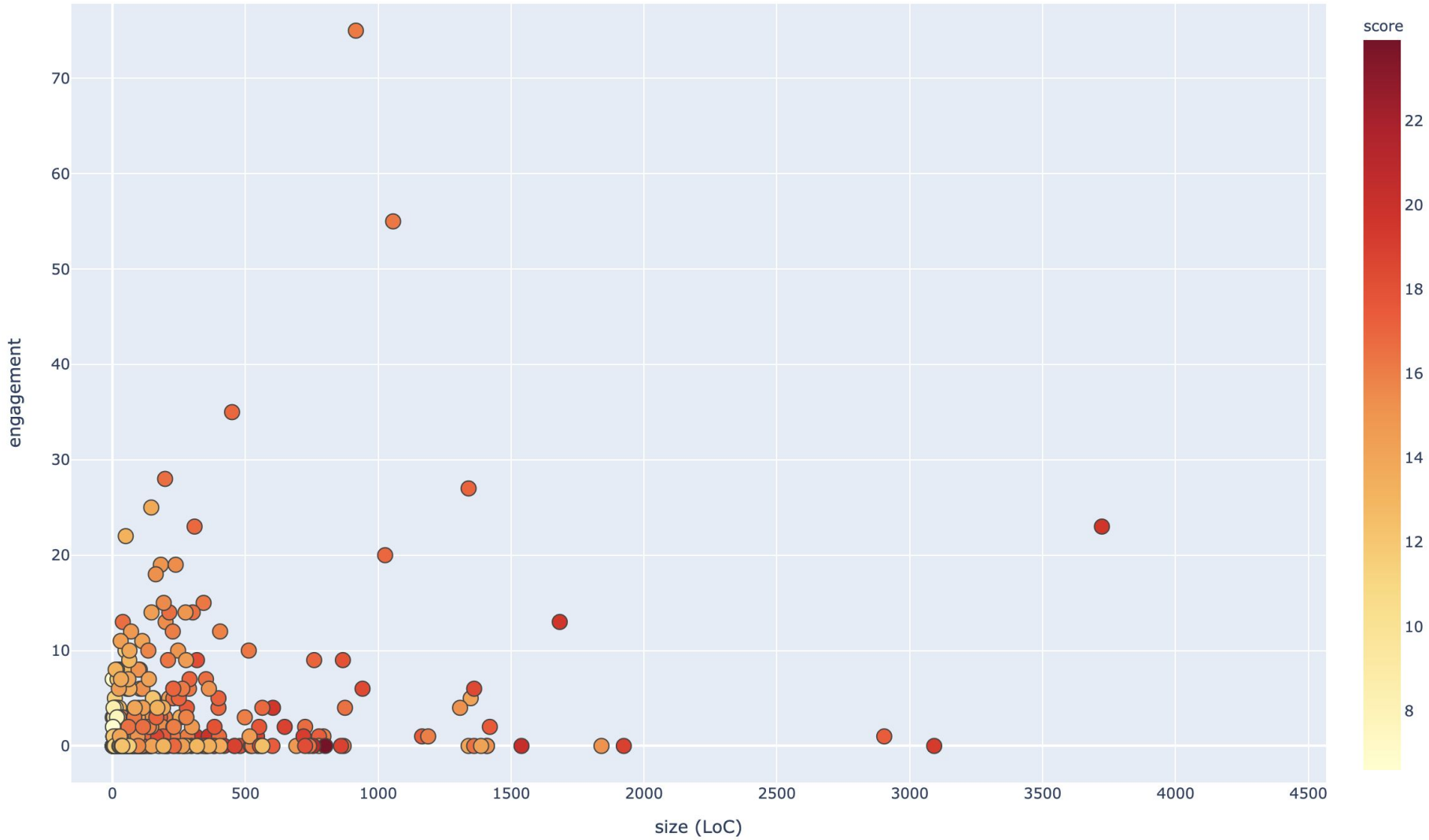
What was I curious to see?

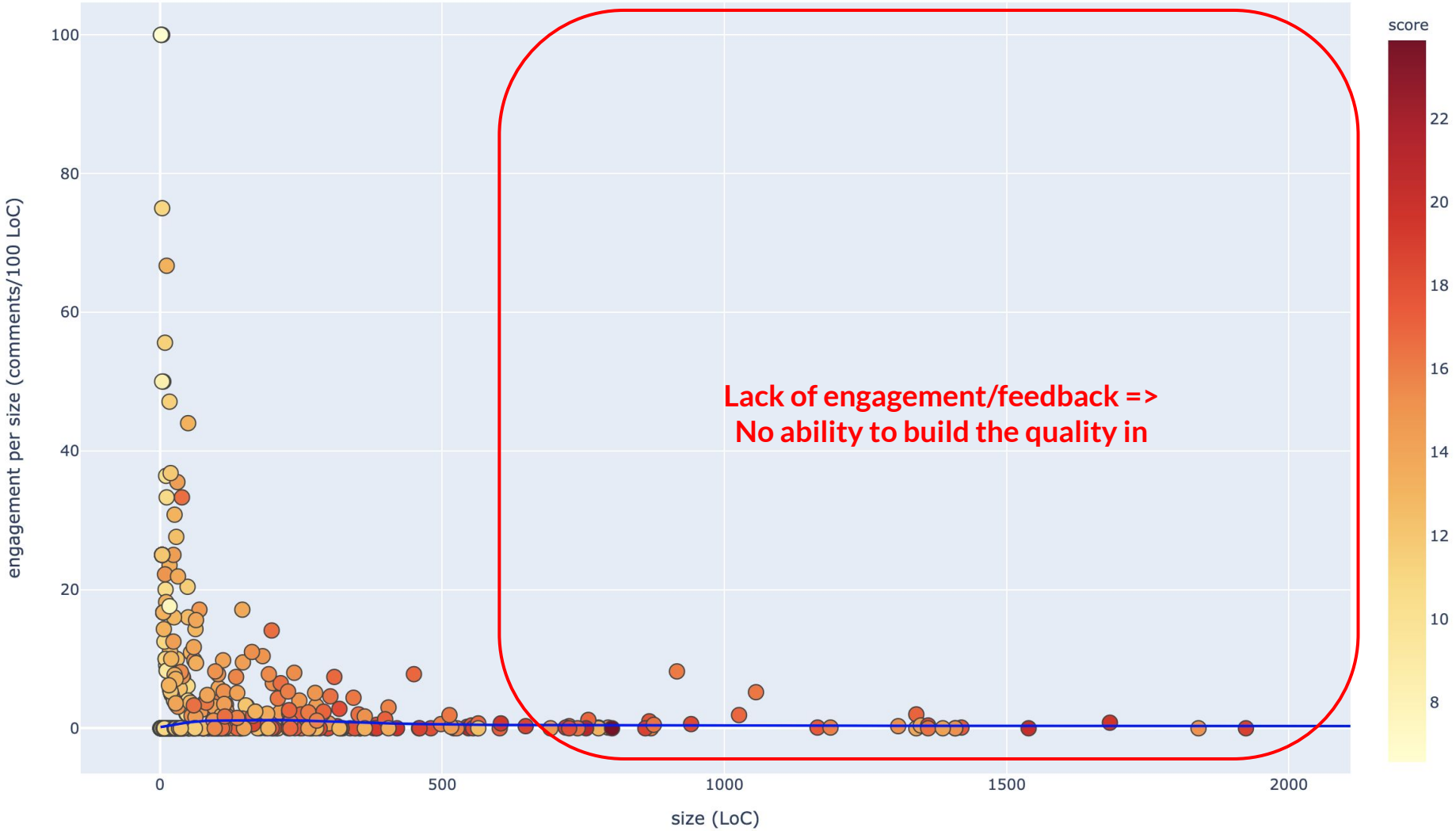
- Engagement
- Wait Time
- Size

Engagement

Why was I curious about the engagement?

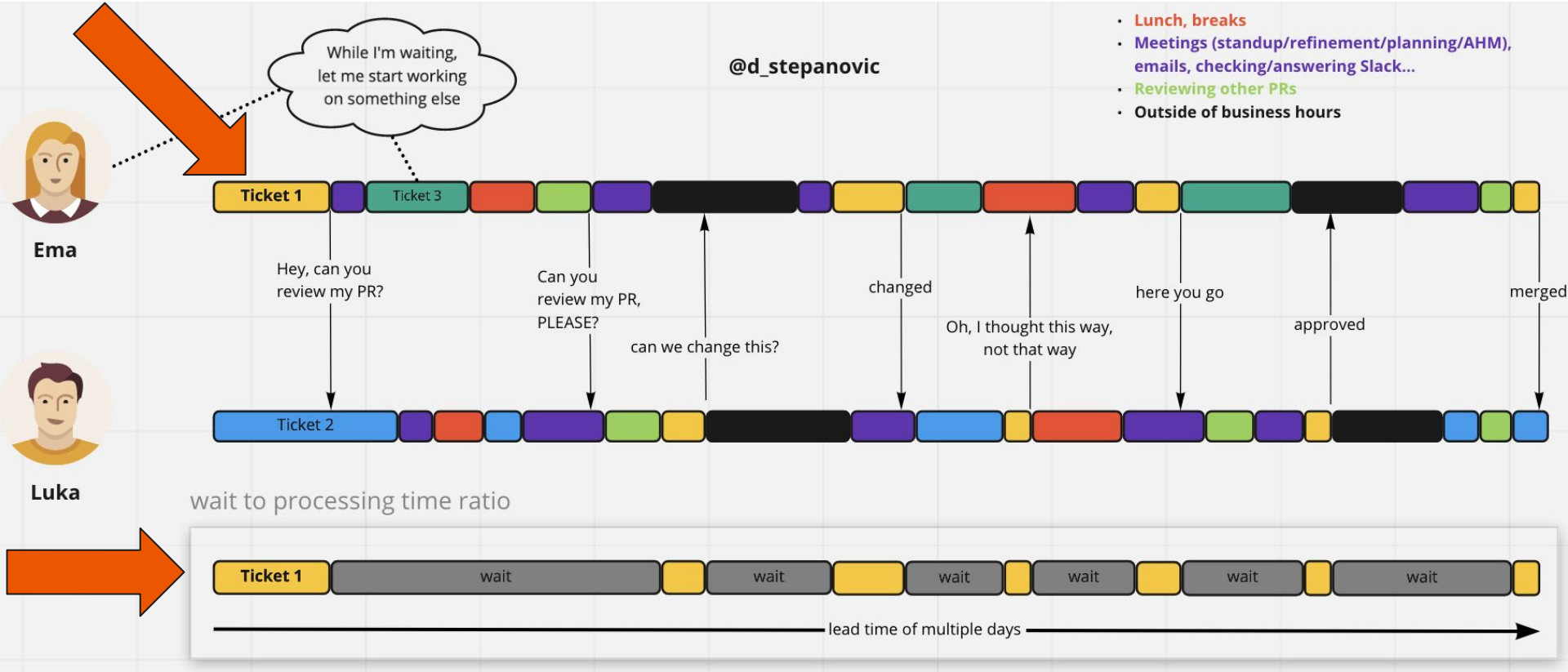
- Systemic effects of delayed and ‘choked’ feedback
  - High-latency, low-throughput feedback
- Engagement by size

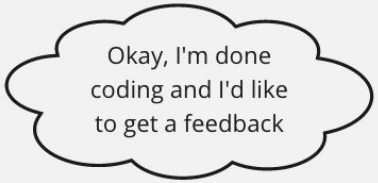

score
22
20
18
16
14
12
10
8
engagement
70
60
50
40
30
20
10
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
size (LoC)


Lack of engagement/feedback =>
No ability to build the quality in
engagement per size (comments/100 LoC)
size (LoC)
score
0
20
40
60
80
100
0
500
1000
1500
2000
8
10
12
14
16
18
20
22

“Never had a huge PR that didn’t look good to me”

Wait time


While I'm waiting, let me start working on something else
@d_stepanovic
Lunch, breaks
Meetings (standup/refinement/planning/AHM), emails, checking/answering Slack...
Reviewing other PRs
Outside of business hours
Ema
Luka
Ticket 1
Ticket 3
Ticket 2
Hey, can you review my PR?
Can you review my PR, PLEASE?
can we change this?
changed
Oh, I thought this way, not that way
here you go
approved
merged
wait to processing time ratio
Ticket 1
wait
lead time of multiple days




```
def start_of_processing_time(self):
    return earliest_of([self.opened_at,
                        self.commits.time_of_first()])
```

```
def end_of_processing_time(self):
    return earliest_of([self.review_requested_at,
                        self.comments.time_of_first_from_human(),
                        self.commits.time_of_last()])
```

```
def start_of_wait_time(self):
    return self.end_of_processing_time()
```

```
def end_of_wait_time(self):
    return self.merged_at
```

Important assumptions and approximations

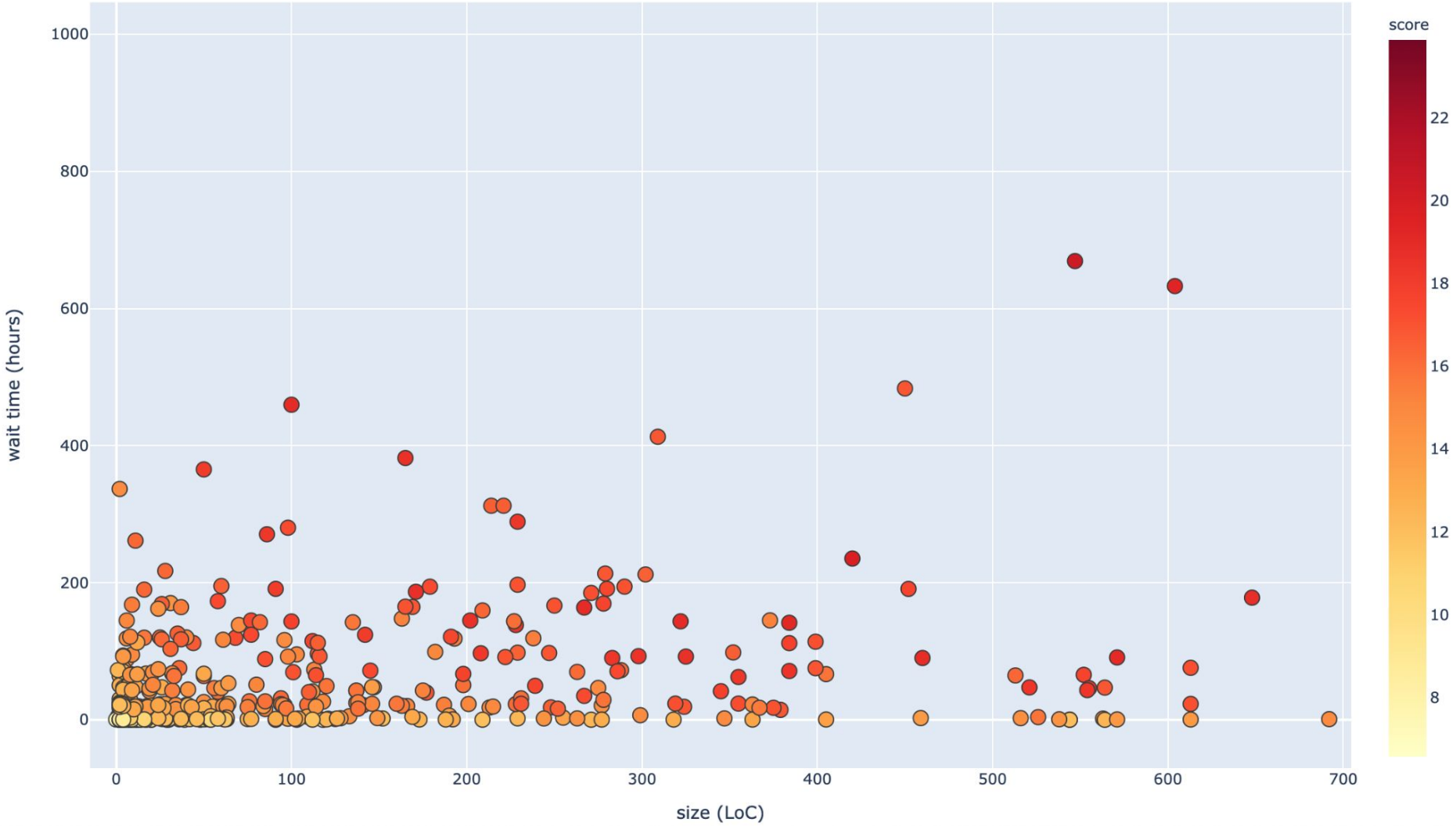
- **Processing time** can have wait time
- **Wait time** can have processing time
- **Processing Time and Flow Efficiency** on the bigger size PRs end of the spectrum inaccurate because of git rewrite practice
- **Wait Time** way more accurate than **Processing Time**
- **PR size** is measured through simple LoC changed

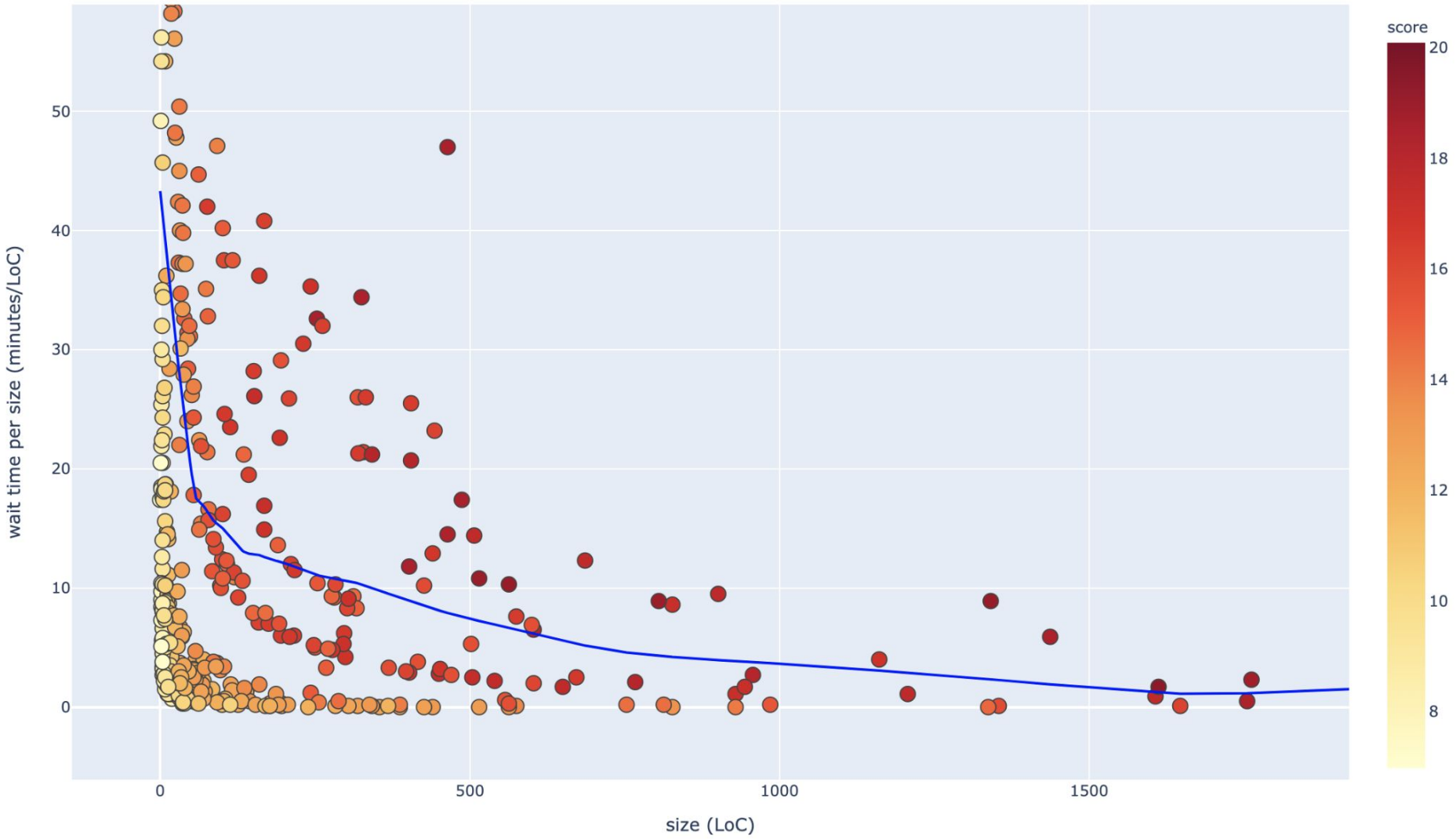
Period

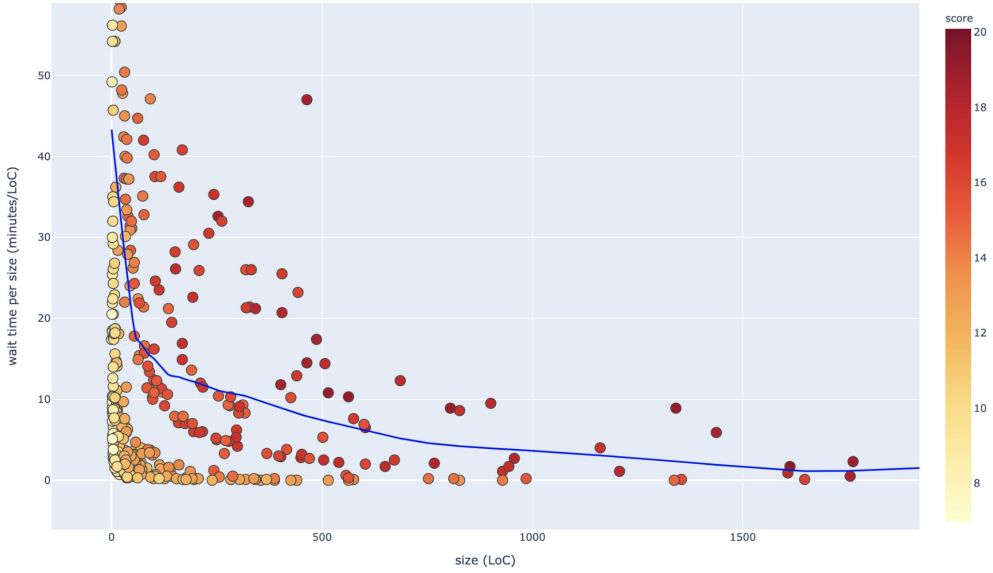
Number of PRs analyzed	500
Period covered (using PR merge time)	['2020-10-13', '2021-04-15']
Period covered (months)	6.1
Period covered (days)	184.0

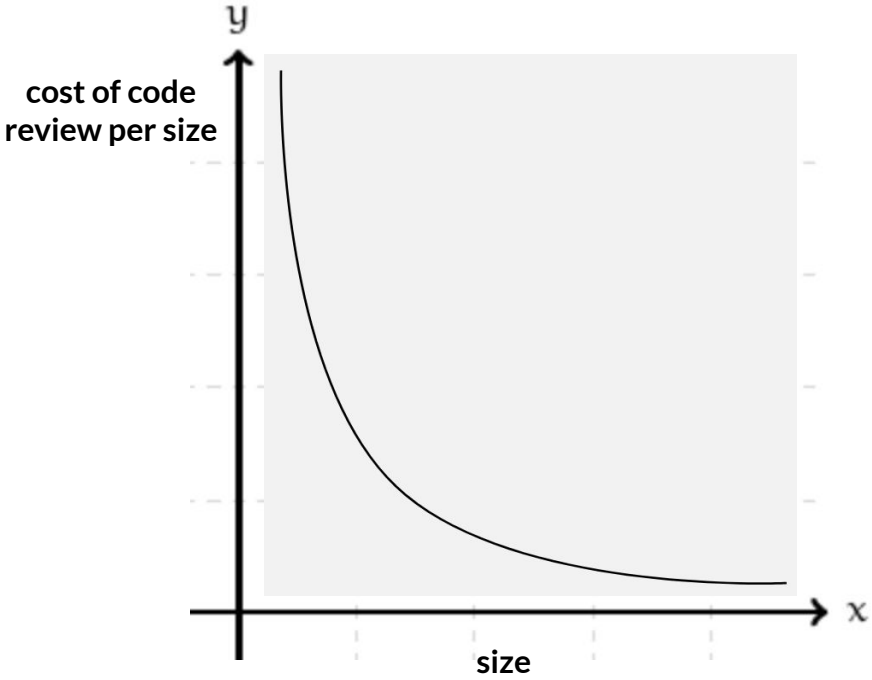
Flow Efficiency

Processing time (cumulative, in days)	298.0
Wait time (cumulative, in days)	830.9
Wait time (cumulative, in months)	27.7
Lead time (cumulative, in days)	1128.9


wait time (hours)
1000
800
600
400
200
0
0
100
200
300
400
500
600
700
size (LoC)
score
22
20
18
16
14
12
10
8


score
20
18
16
14
12
10
8
wait time per size (minutes/LoC)
50
40
30
20
10
0
0
500
1000
1500
size (LoC)


score
20
18
16
14
12
10
8
wait time per size (minutes/LoC)
50
40
30
20
10
0
0
500
1000
1500
size (LoC)

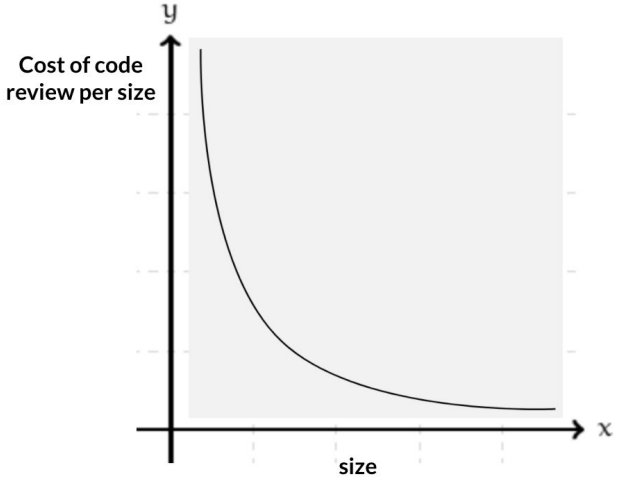

y
cost of code
review per size
x
size

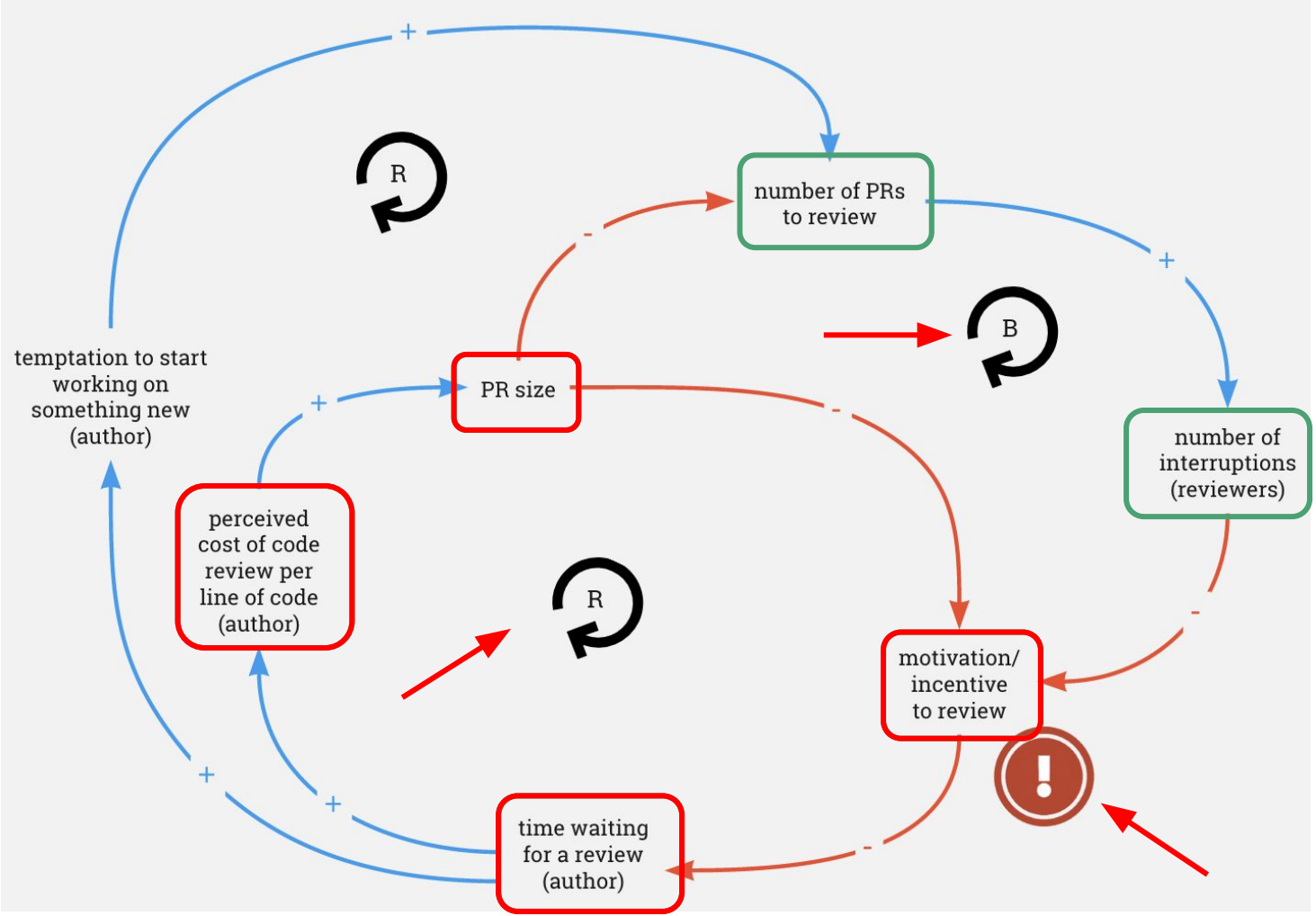
Small PRs

- quicker to write
- quicker to review
- less time allocation for review
- higher engagement
- less risky
- shorter Lead Time to Change and higher Deployment frequency
- etc.

The system that people work in and the interaction with people may account for 90 or 95 percent of performance

W. Edwards Deming

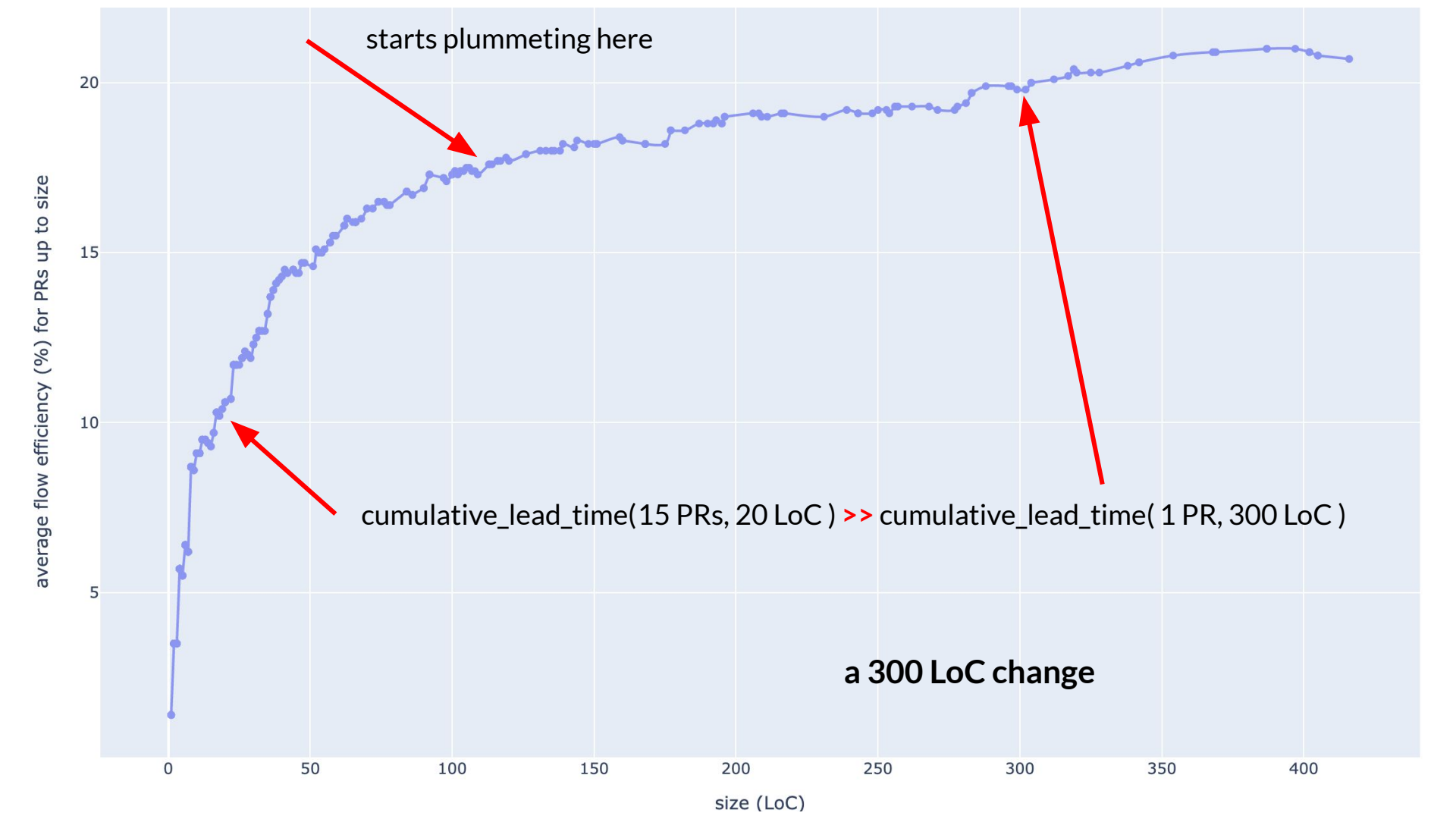

y
Cost of code
review per size
x
size

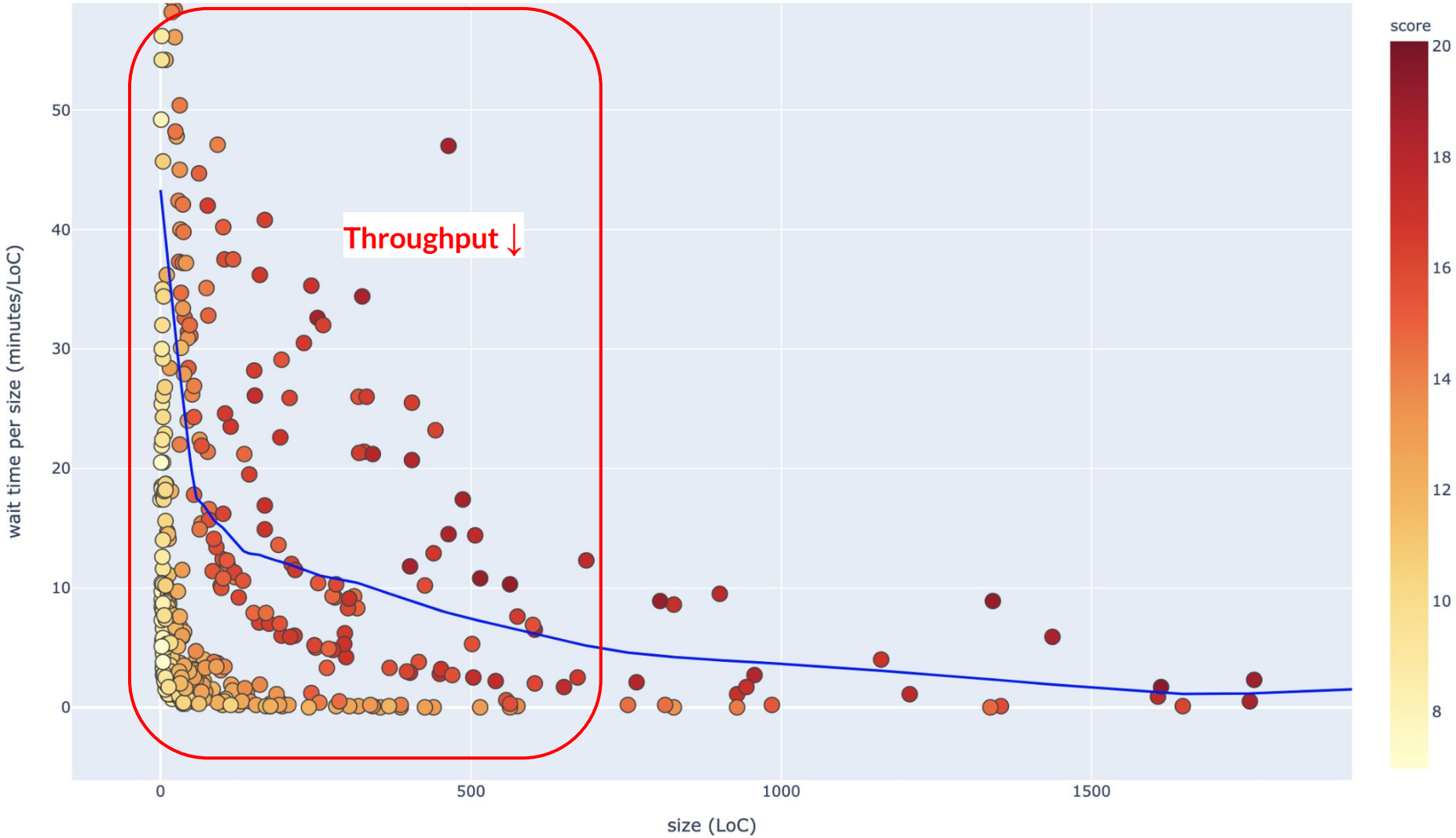


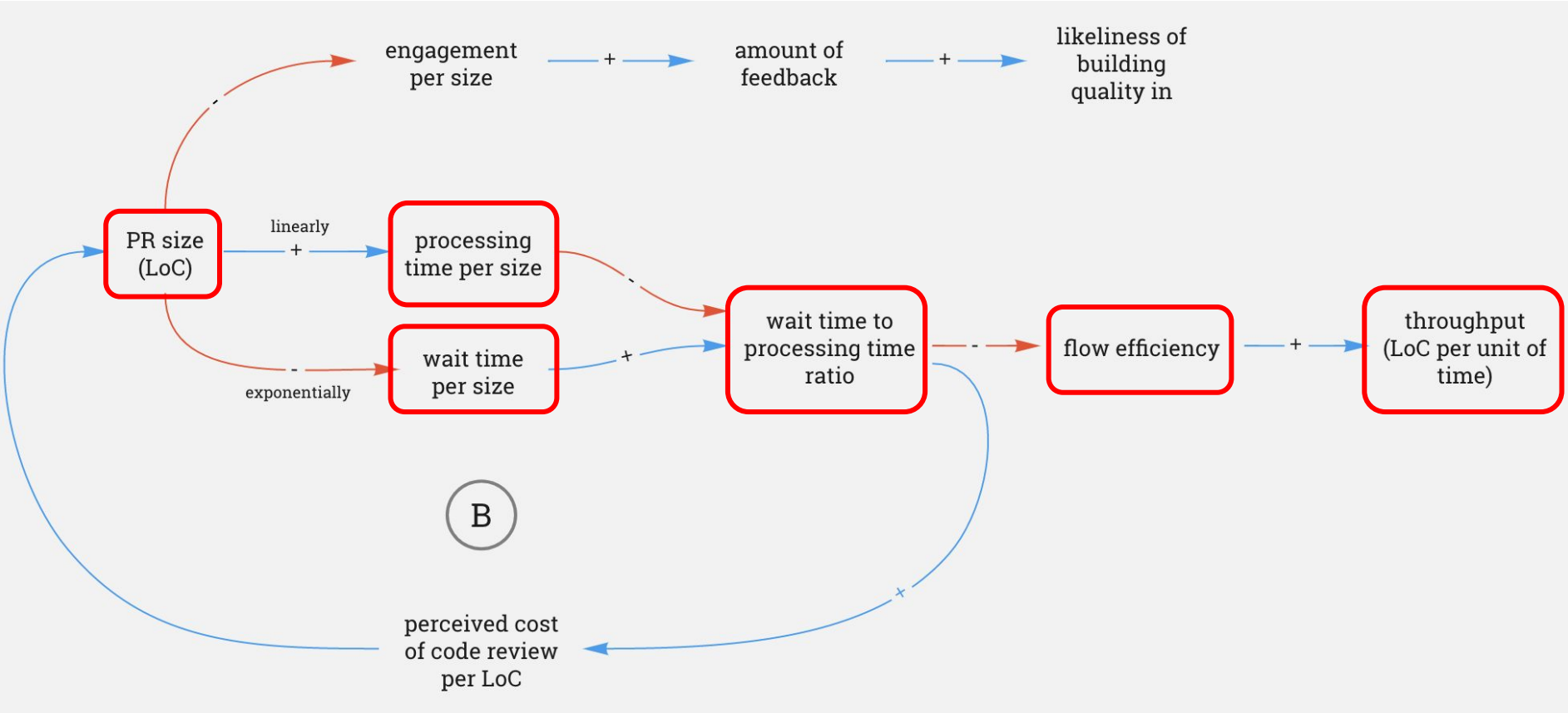
www.draganstepanovic.com/2020/11/16/pr-size-cannot-be-reduced.html

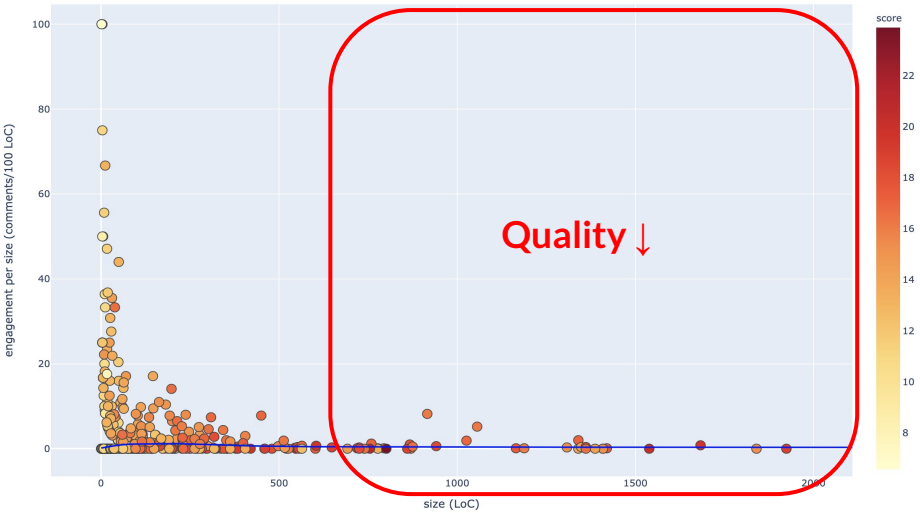
Flow efficiency

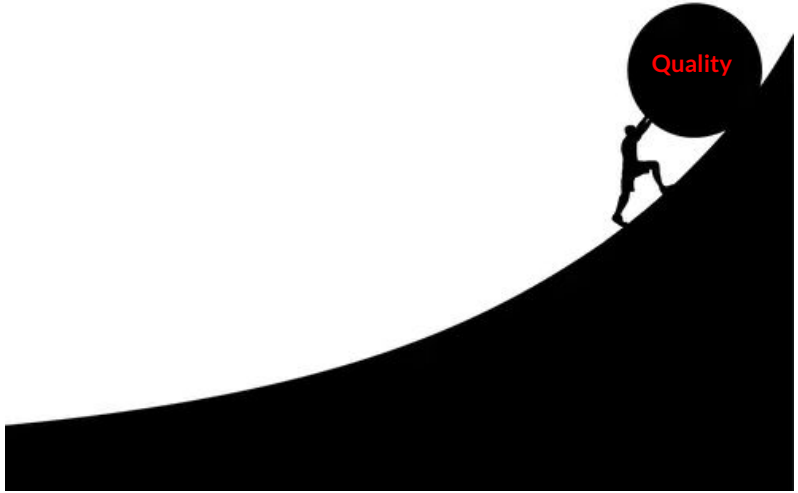
$$\frac{\text{work}}{\text{work} + \text{wait}} \times 100\%$$

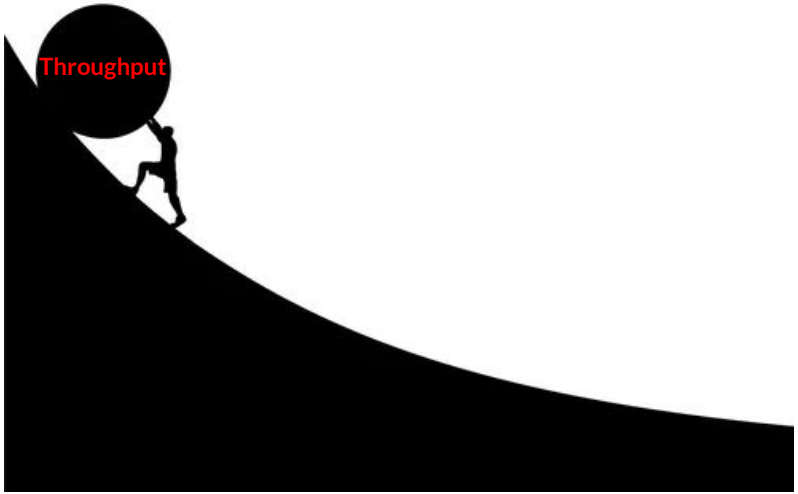

starts plummeting here
cumulative_lead_time(15 PRs, 20 LoC) >> cumulative_lead_time(1 PR, 300 LoC)
a 300 LoC change
average flow efficiency (%) for PRs up to size
size (LoC)
20
15
10
5
0
50
100
150
200
250
300
350
400

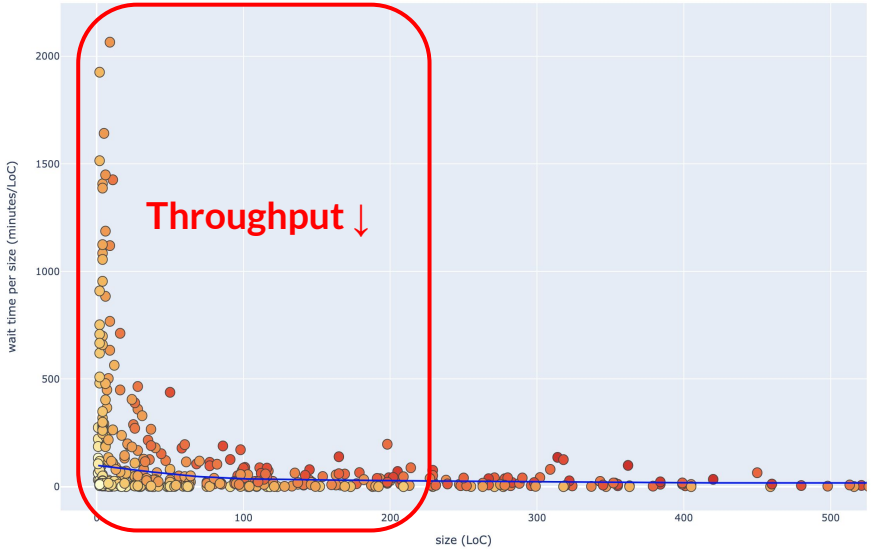

Throughput ↓
wait time per size (minutes/LoC)
size (LoC)
score
0
10
20
30
40
50
500
1000
1500
8
12
14
16
18
20

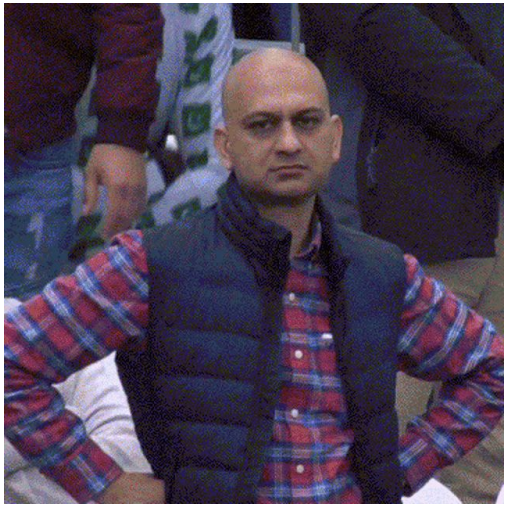

engagement per size
+
amount of feedback
+
likeliness of building quality in
-
PR size (LoC)
linearly
+
processing time per size
-
wait time to processing time ratio
-
flow efficiency
+
throughput (LoC per unit of time)
-
exponentially
wait time per size
+
B
+
perceived cost of code review per LoC


Quality ↓
engagement per size (comments/100 LoC)
size (LoC)
score

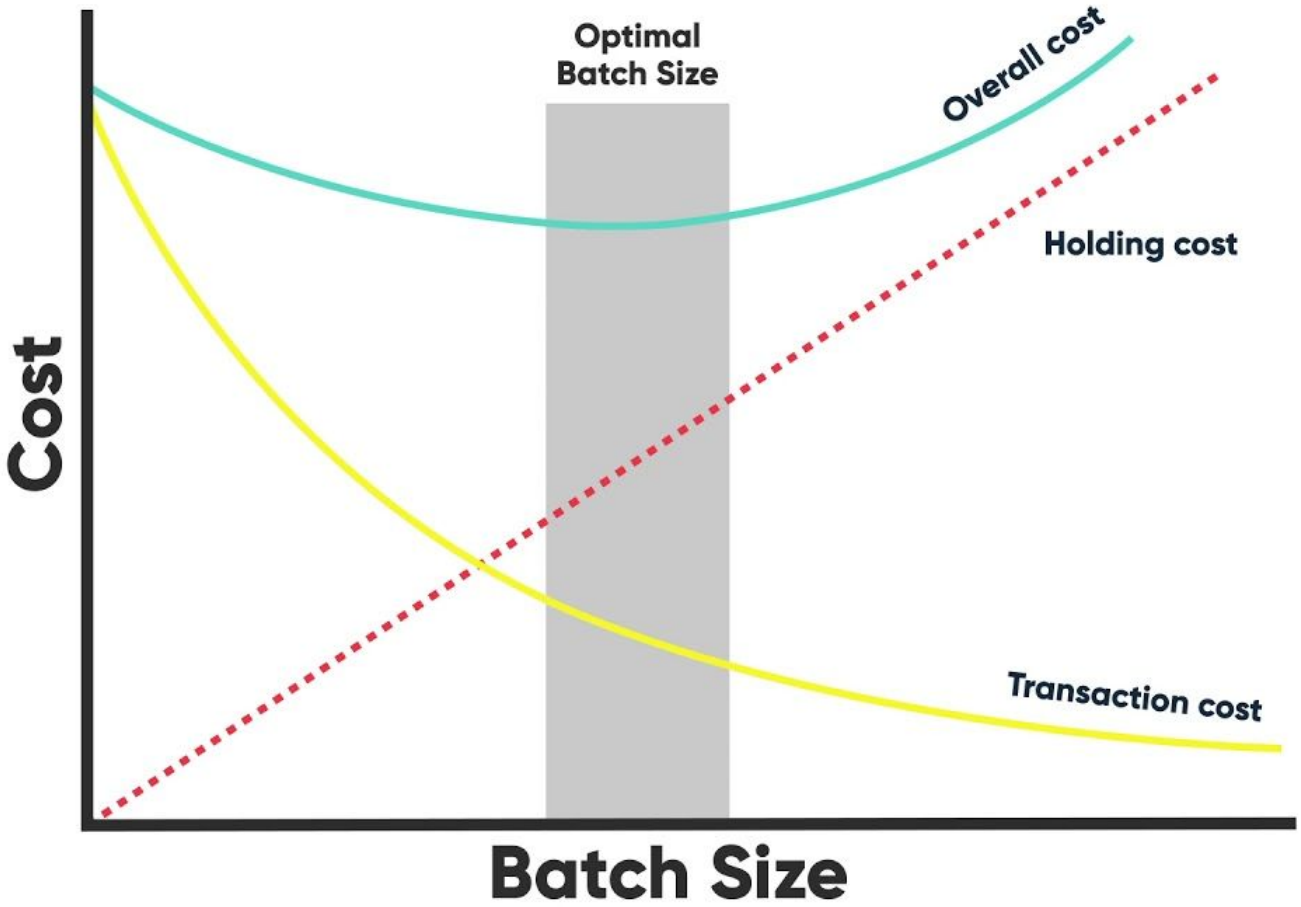

Quality


Throughput


Throughput ↓
wait time per size (minutes/LoC)
size (LoC)

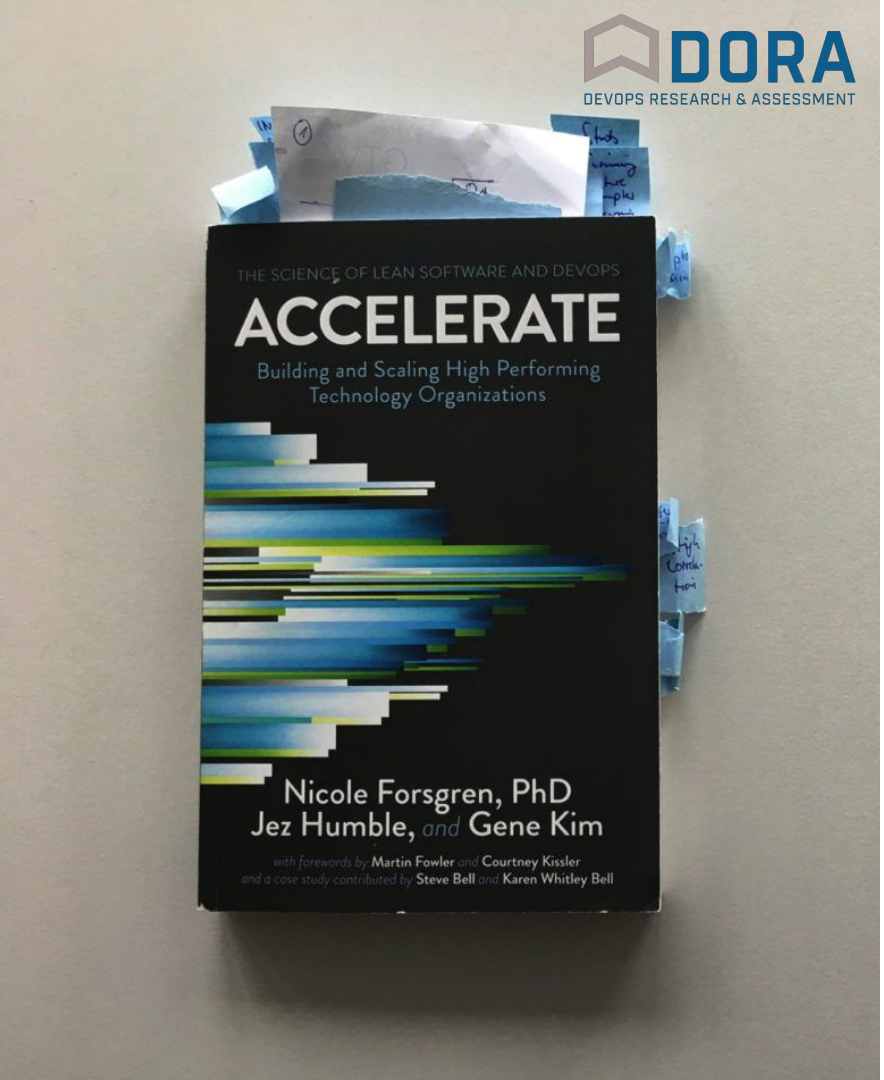



Throughput
Quality



Donald G. Reinertsen: The Principles of Product Development Flow

https://lastcallmedia.com/blog/why-devops-most-important-tech-strategy-today



DORA
DEVOPS RESEARCH & ASSESSMENT

BETTER

Throughput

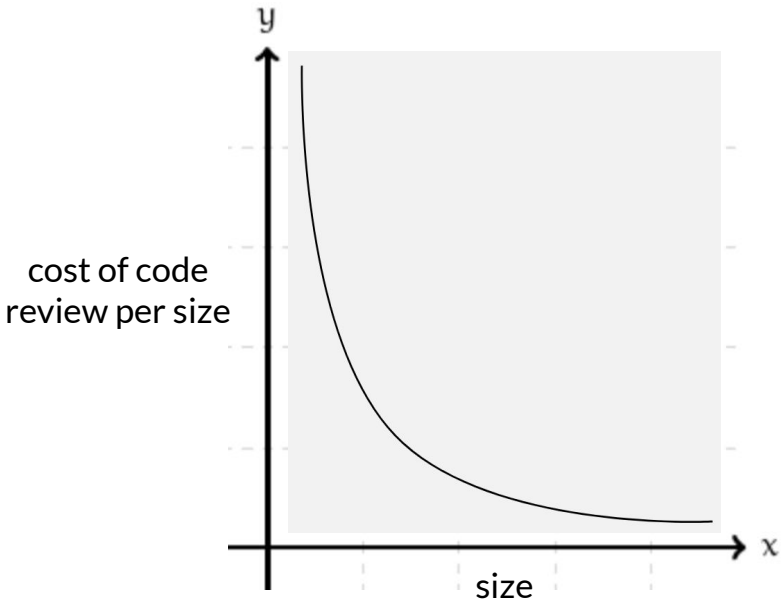
AND

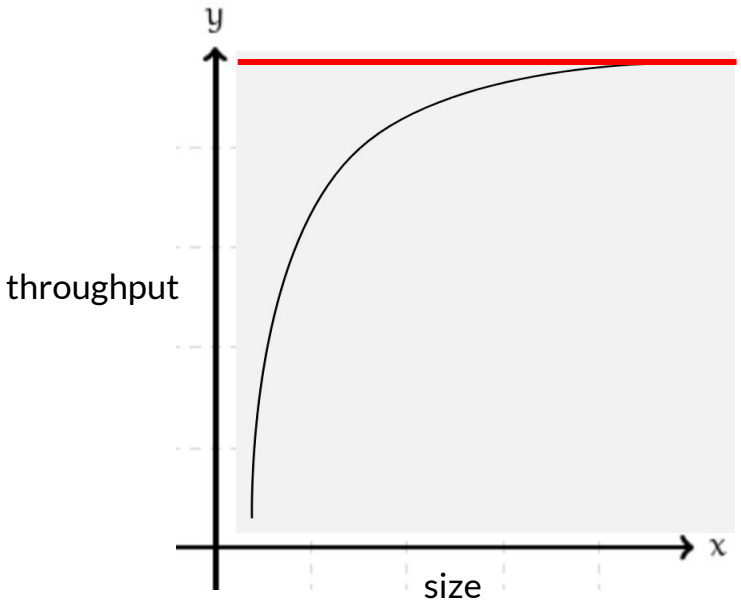
Stability

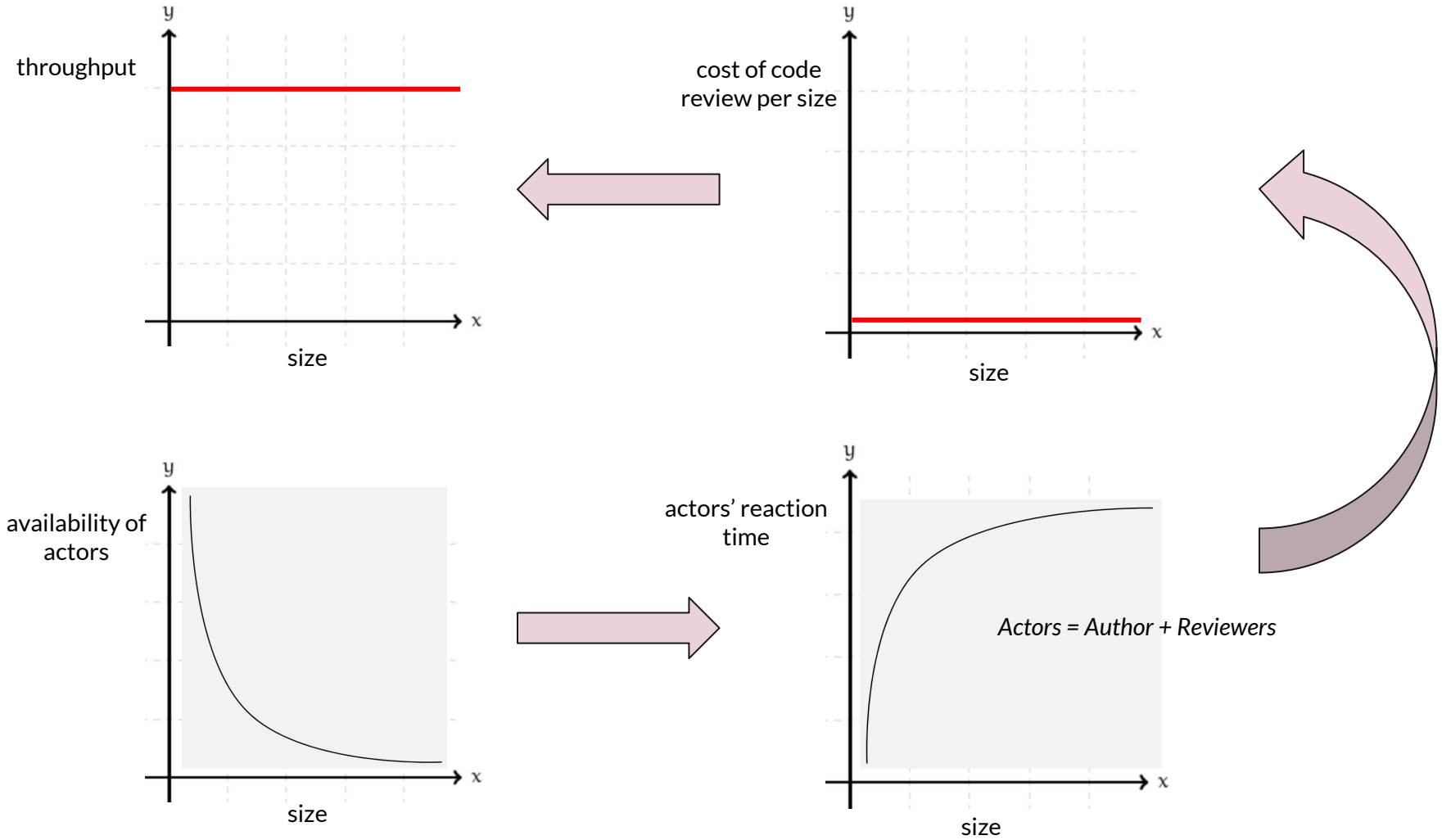
~~“There’s always a trade-off”~~

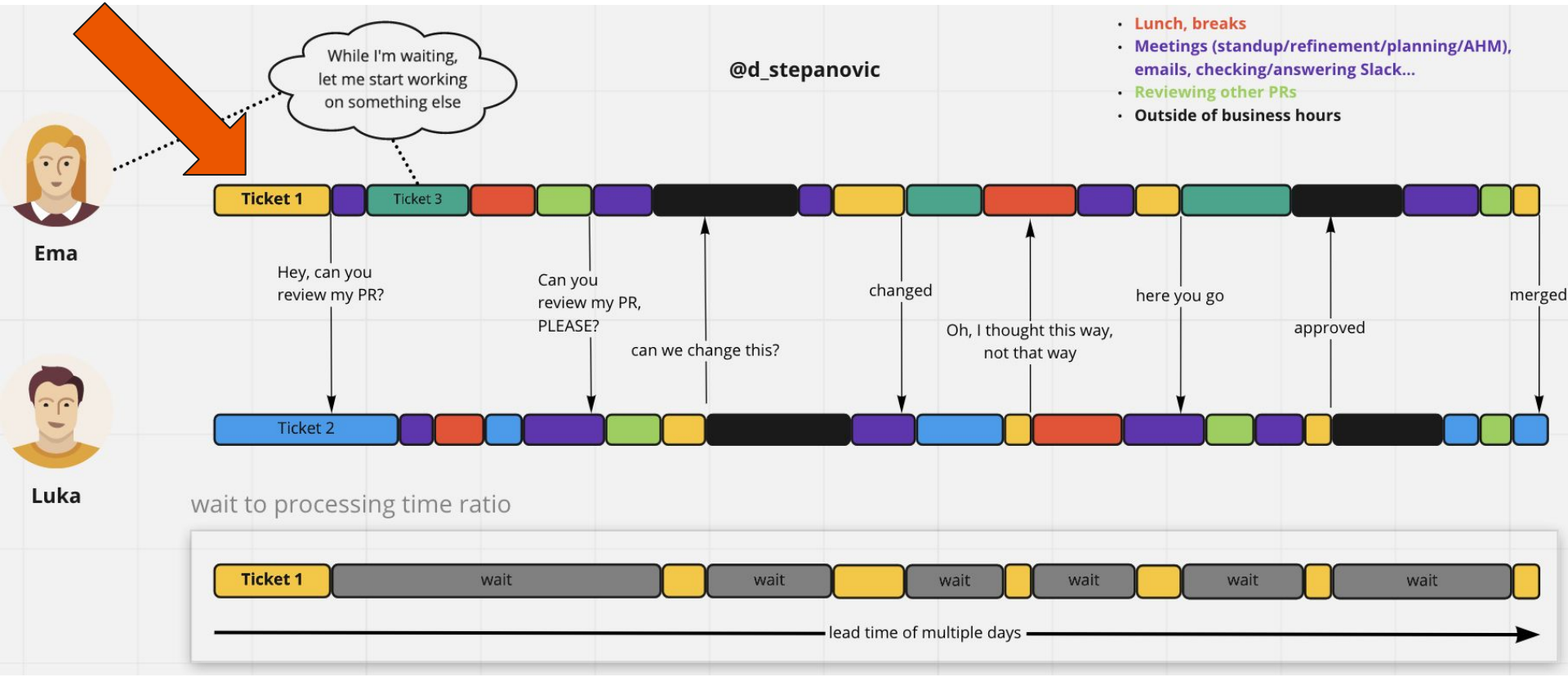
Some trade-offs actually do not exist because underlying assumption is flawed

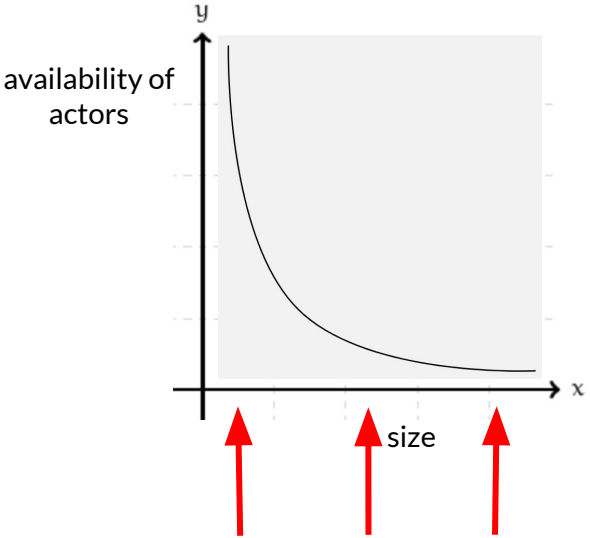
https://sircharlescaryinc.com/having-your-cake-and-eating-it-too/

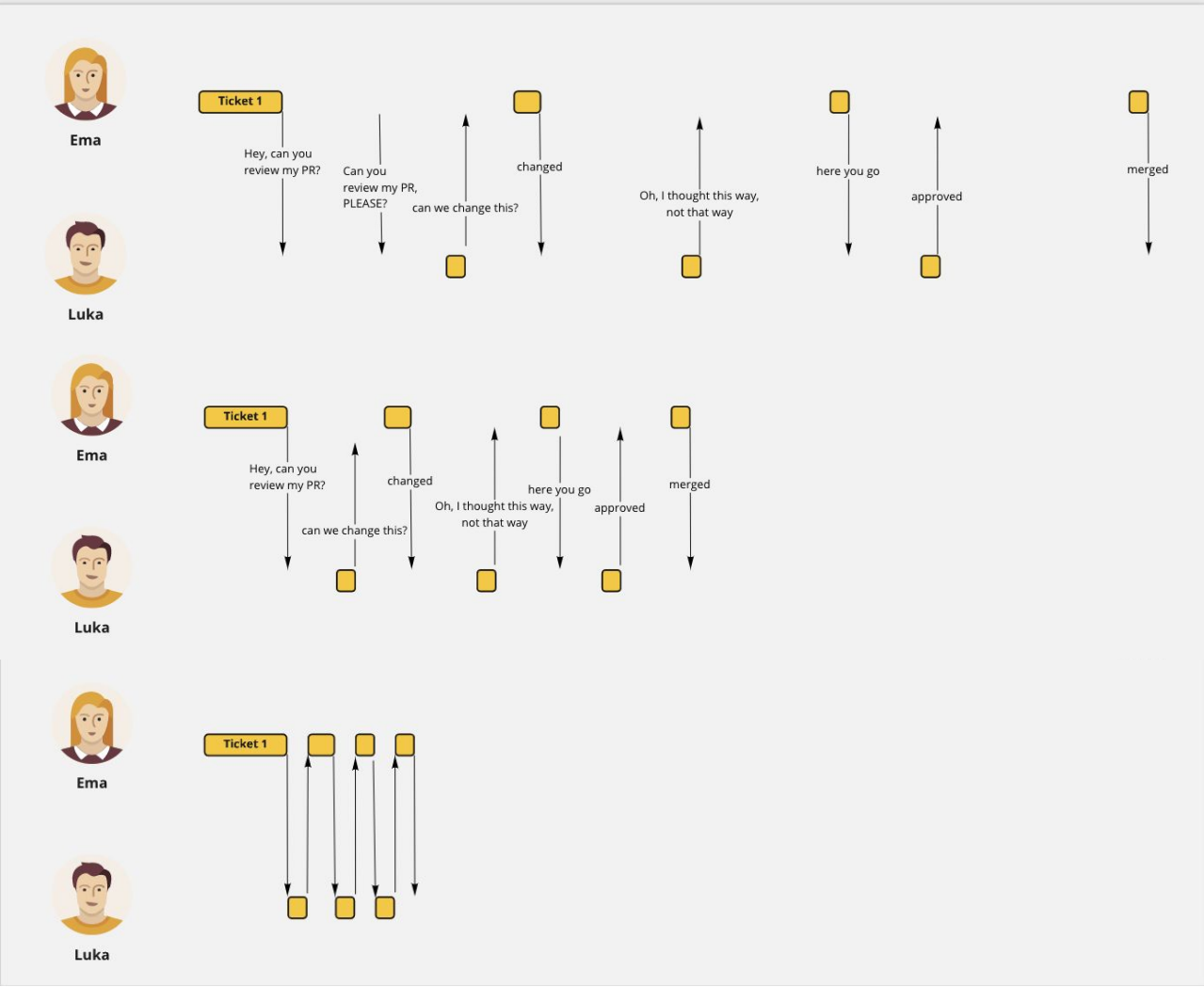

y
cost of code
review per size
x
size


y
throughput
x
size


throughput
y
x
size
cost of code review per size
y
x
size
availability of actors
y
x
size
actors' reaction time
y
x
size
Actors = Author + Reviewers


While I'm waiting, let me start working on something else
@d_stepanovic
Lunch, breaks
Meetings (standup/refinement/planning/AHM), emails, checking/answering Slack...
Reviewing other PRs
Outside of business hours
Ema
Luka
Ticket 1
Ticket 3
Ticket 2
Hey, can you review my PR?
Can you review my PR, PLEASE?
can we change this?
changed
Oh, I thought this way, not that way
here you go
approved
merged
wait to processing time ratio
Ticket 1
wait
lead time of multiple days

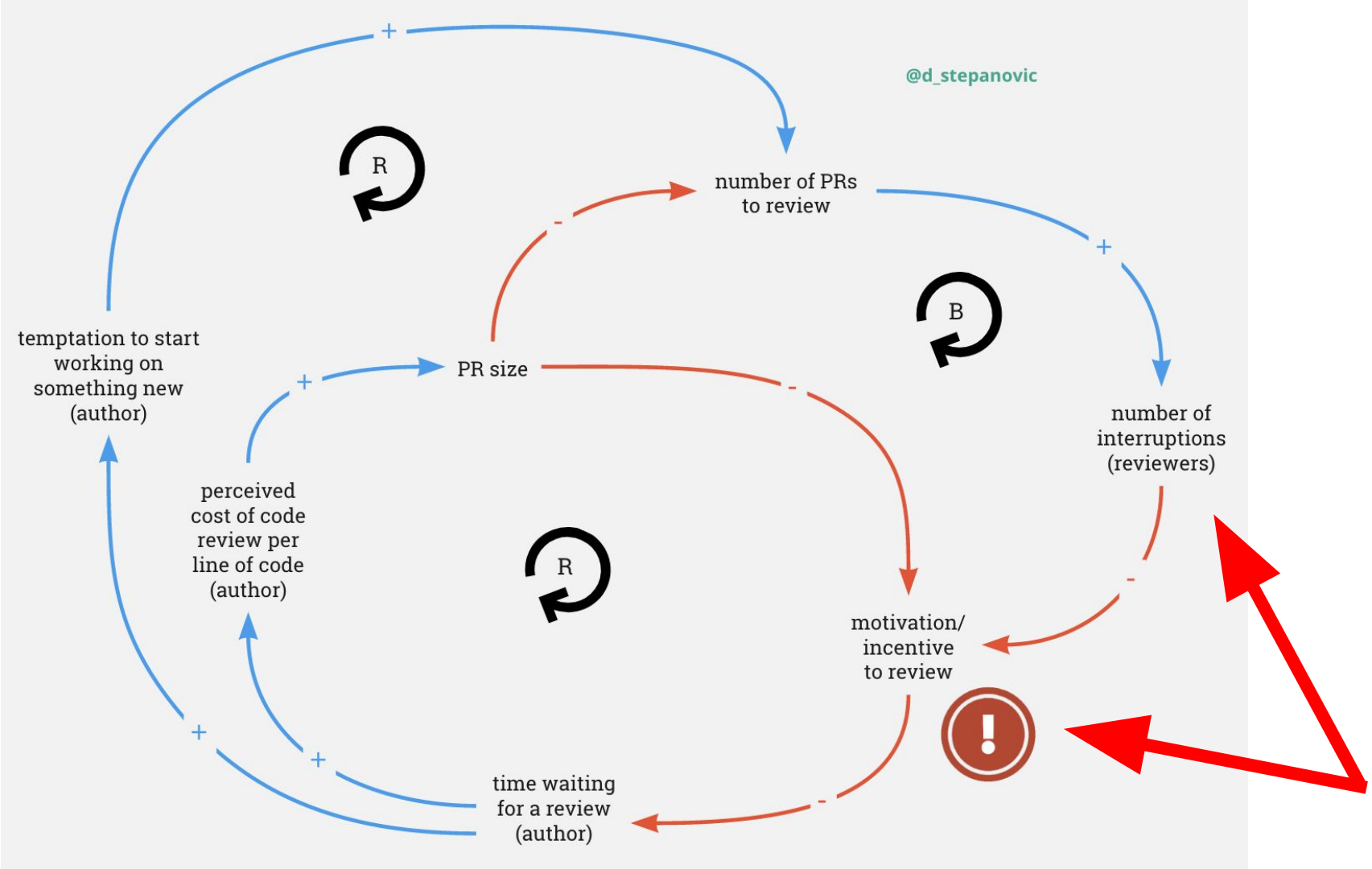

y
availability of actors
x
size


Ema
Luka
Ticket 1
Hey, can you review my PR?
Can you review my PR, PLEASE?
can we change this?
changed
Oh, I thought this way, not that way
here you go
approved
merged
Ema
Luka
Ticket 1
Hey, can you review my PR?
can we change this?
changed
Oh, I thought this way, not that way
here you go
approved
merged
Ema
Luka
Ticket 1

In order to not exponentially lose the throughput while reducing the average size of a PR

people need to get exponentially closer and closer in time

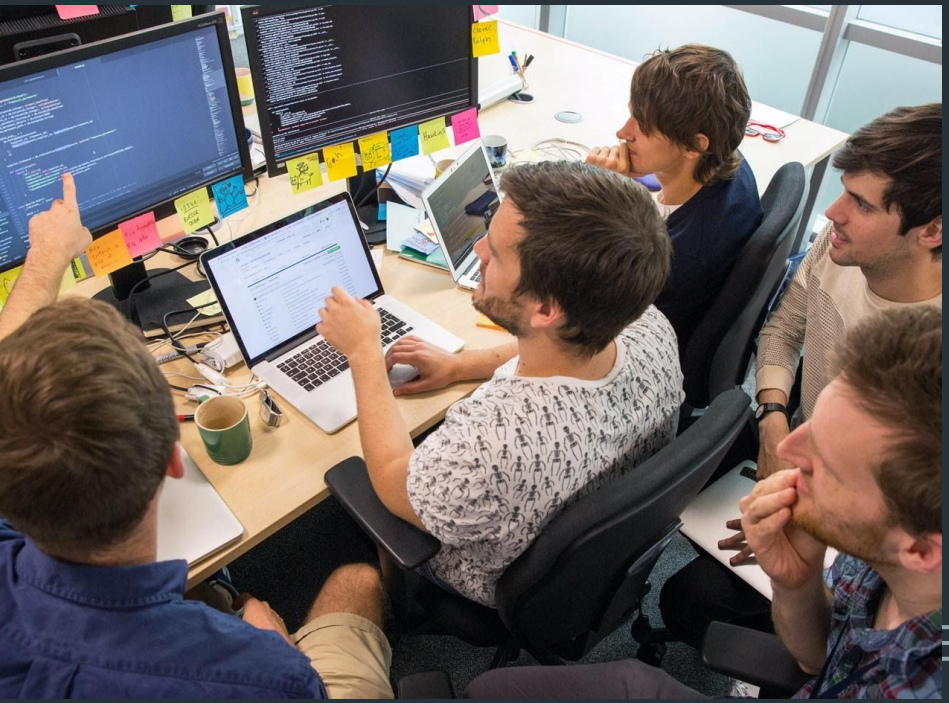
=> *Continuous Code Review*


@d_stepanovic
R
number of PRs
to review
B
temptation to start
working on
something new
(author)
PR size
number of
interruptions
(reviewers)
perceived
cost of code
review per
line of code
(author)
R
motivation/
incentive
to review
time waiting
for a review
(author)


You cannot be interrupted if you're not doing anything else

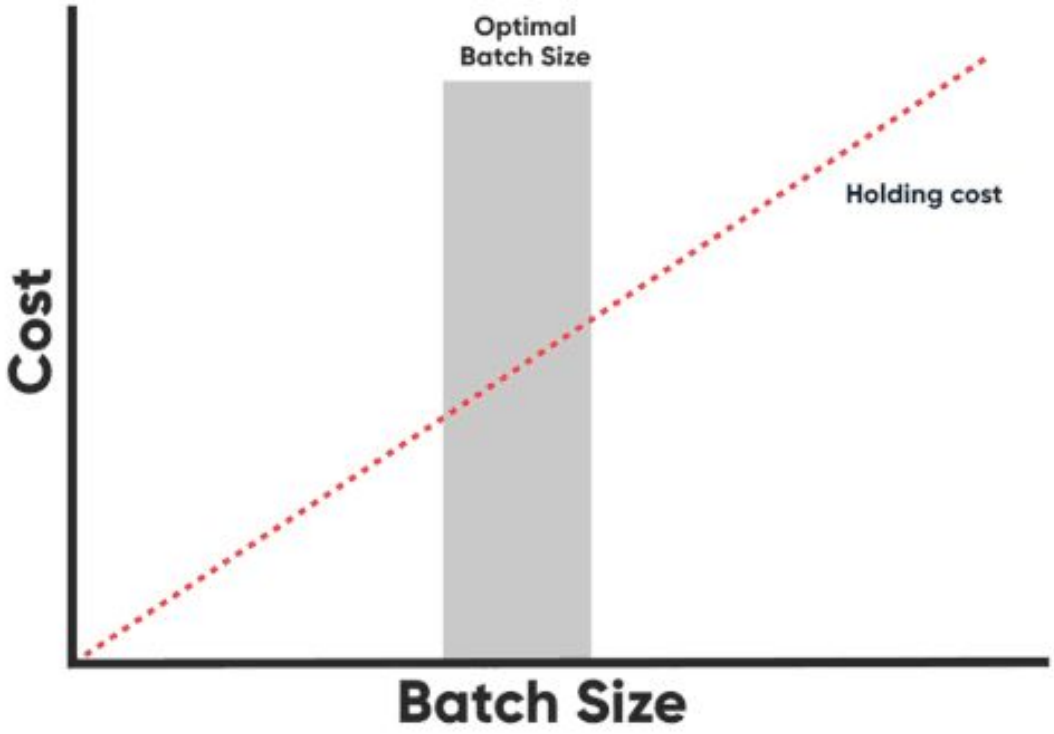
https://i.ytimg.com/vi/2TUza5C2uJ8/maxresdefault.jpg

Enter Co-creation patterns



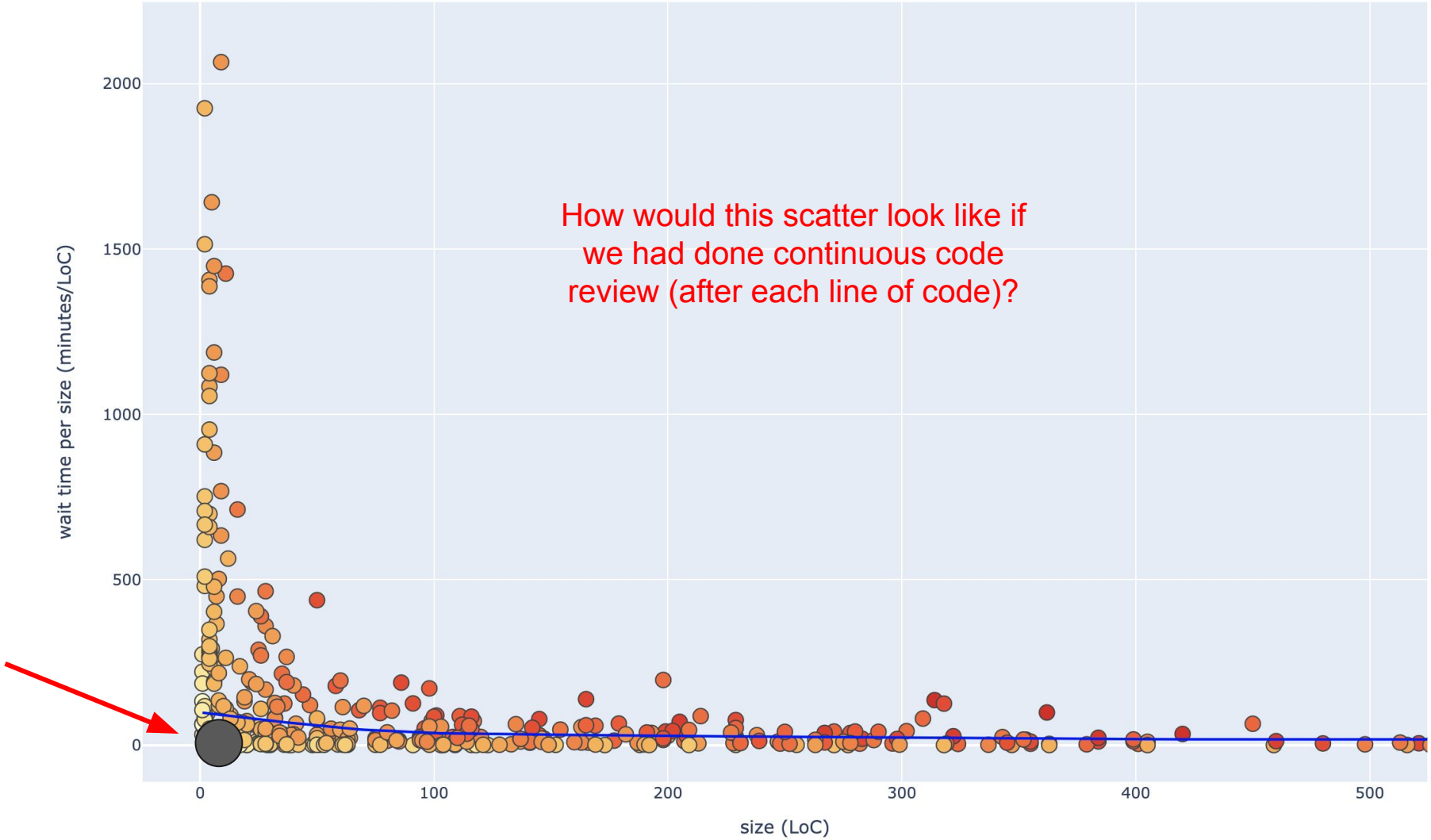
https://en.wikipedia.org/wiki/Mob_programming

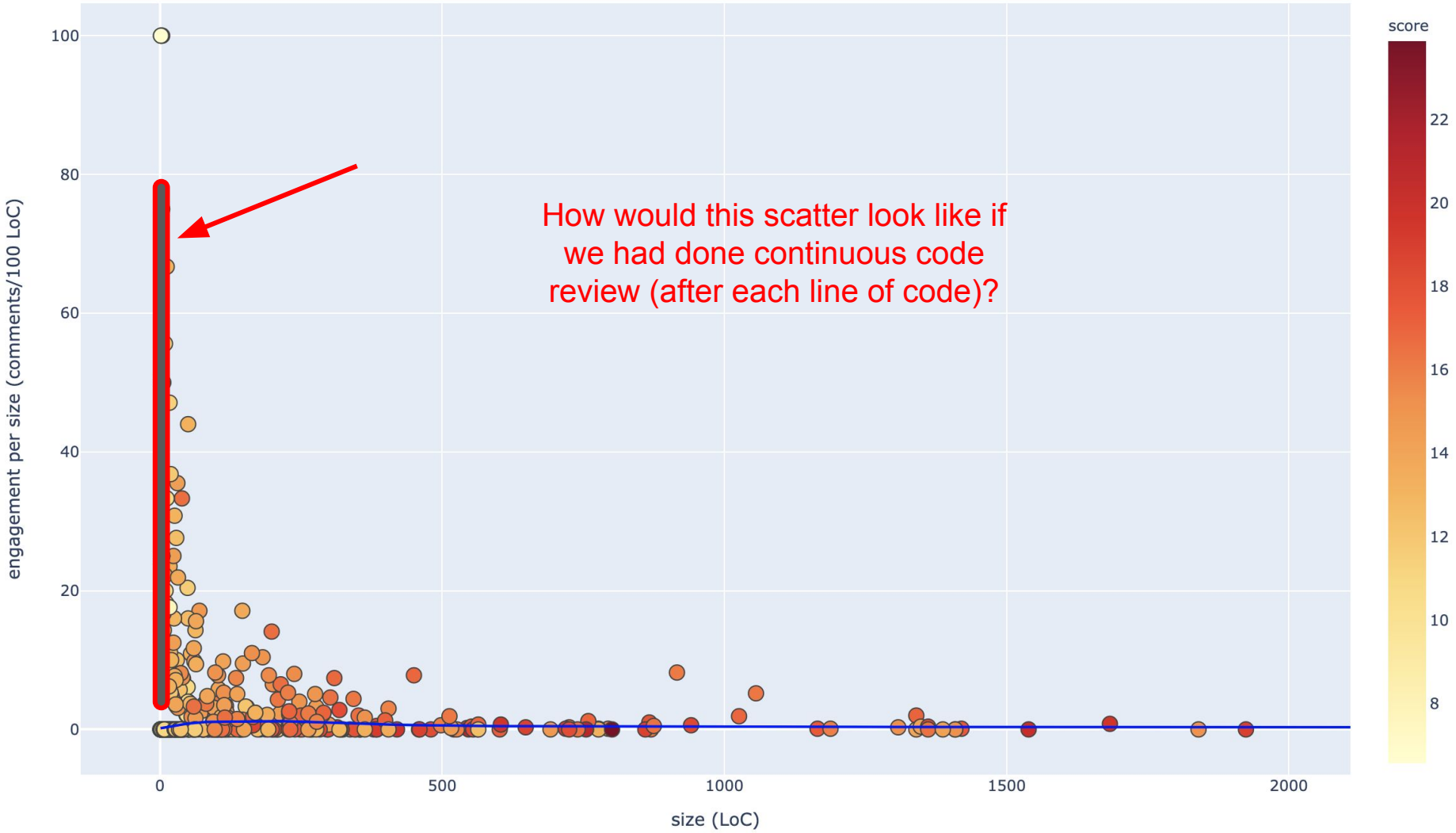
https://www.codefellows.org/blog/6-reasons-for-pair-programming/



-Donald G. Reinertsen: The Principles of Product Development Flow

https://lastcallmedia.com/blog/why-devops-most-important-tech-strategy-today


How would this scatter look like if we had done continuous code review (after each line of code)?
wait time per size (minutes/LoC)
2000
1500
1000
500
0
0
100
200
300
400
500
size (LoC)

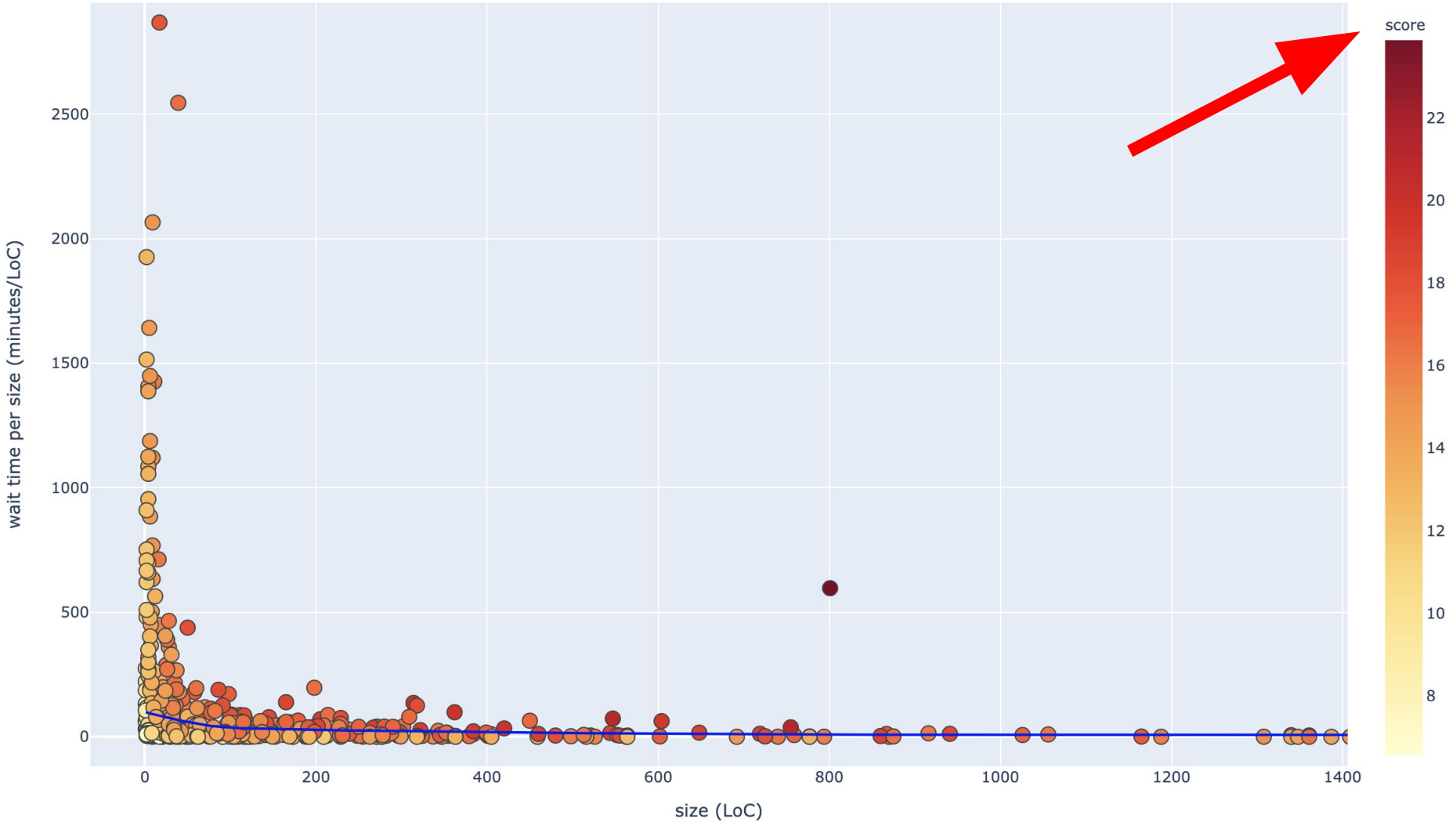

How would this scatter look like if we had done continuous code review (after each line of code)?
engagement per size (comments/100 LoC)
size (LoC)
score
0
20
40
60
80
100
0
500
1000
1500
2000
8
10
12
14
16
18
20
22

Throughput

AND

Quality

PR Score


score
22
20
18
16
14
12
10
8
wait time per size (minutes/LoC)
2500
2000
1500
1000
500
0
0
200
400
600
800
1000
1200
1400
size (LoC)

What are we trying to optimize for?

Size ↓
Wait time per size ↓
Engagement per size↑ (or not ↓)

PR Score

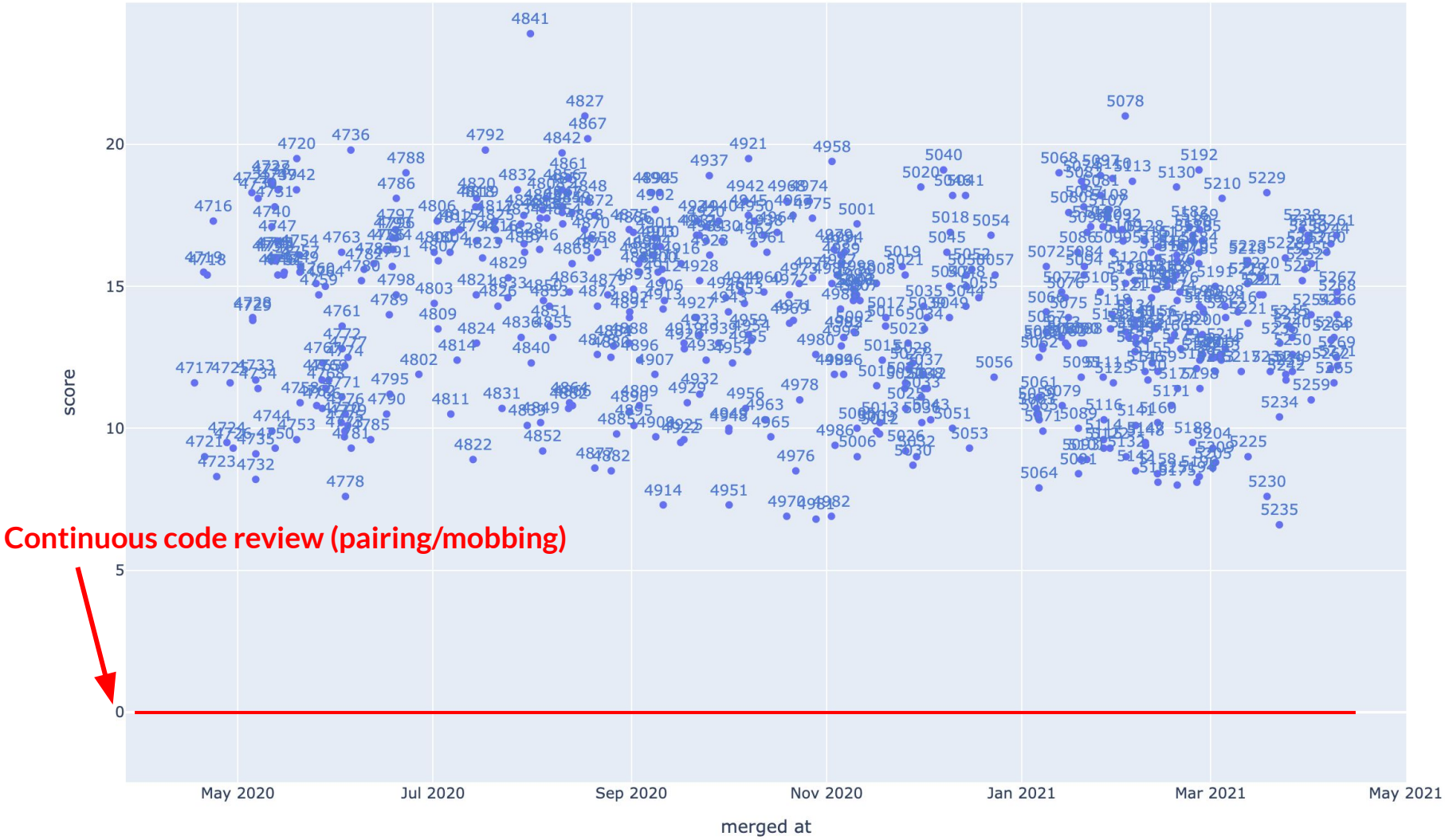
```
def pr_score_for(size, engagement, wait_time):
    return math.log(1 + Score.absolute(engagement, size, wait_time))
```

= 0

```
def absolute_score(size, wait_time, engagement):
    return size * wait_time.in_seconds() / (1 + engagement)
```

= 0

1 0 ↑


Continuous code review (pairing/mobbing)
score
merged at
May 2020
Jul 2020
Sep 2020
Nov 2020
Jan 2021
Mar 2021
May 2021
20
15
10
5
0

The optimal size of Pull Request is one LoC that is reviewed immediately as it's being typed.

And I don't know of a better way to achieve it than by Pair/Mob programming.

How would the world look like had we paired (for PRs up to 100 LoC)?

What would be the cumulative lead time had we paired?	146.1 instead of 486.4 days (-340.3)
How many times sooner would we finish had we paired?	3.3x
When would we finish had we paired?	2020-12-11 instead of 2021-04-15
How long would it take us had we paired?	53.5 instead of 178.0 days (-124.5)
Number of PRs with size up to a median	319
How many more PRs would we finish had we paired?	1053 (+734)
How many more PRs would we finish had we paired (%)?	+230%

~~We've been told all along that we'll achieve more if we limit and delay our interactions~~

Hope you now (also) have a data-informed reason to not believe that

@d_stepanovic