Typesafe

Fighting Bit rot

With your mind

Viktor Klang

Director of Engineering

Typesafe

Typesafe

Bit rot

> The term "bit rot" is often used to refer to dormant code rot, i.e. **the fact that dormant (unused or little-used) code gradually decays in correctness** as a result of interface changes in active code that is called from the dormant code.
>
> - http://en.wikipedia.org/wiki/Bit_rot

Typesafe

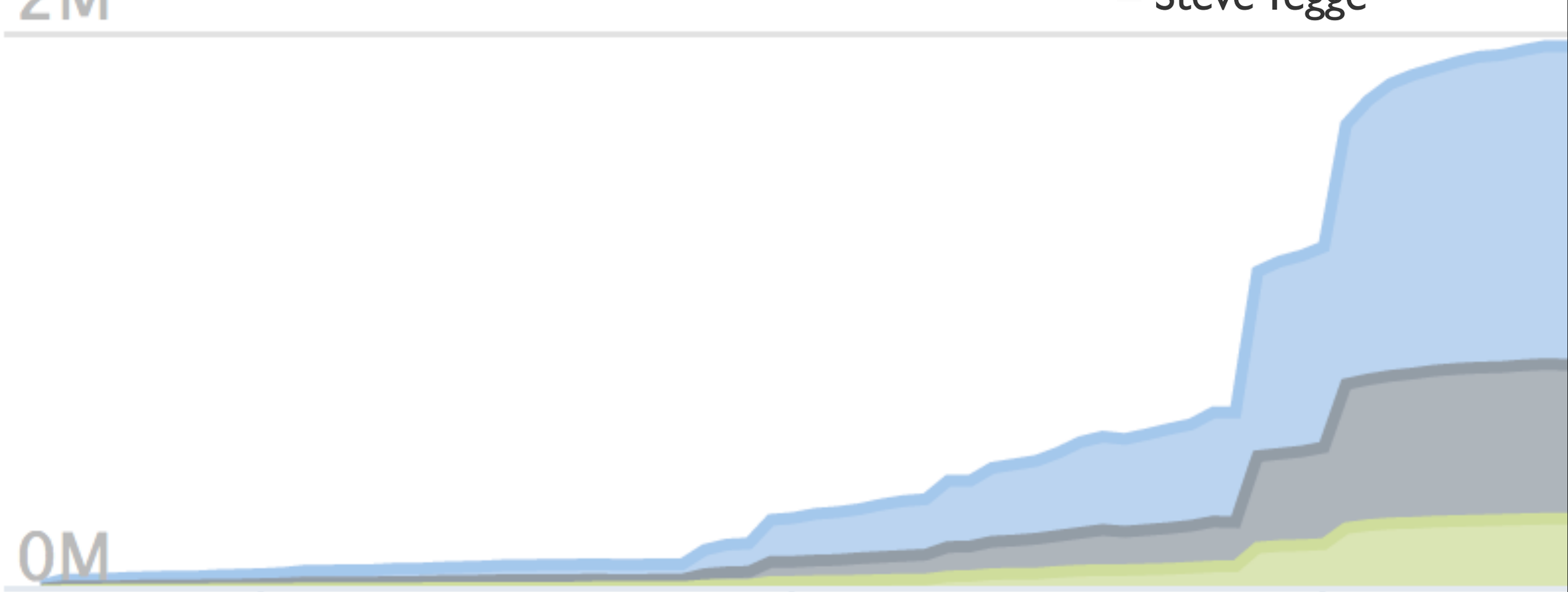
Typesafe

... or how to create maintainable software

Typesafe

Lines of Code

> I happen to hold a hard-won minority opinion about code bases. In particular I believe, quite staunchly I might add, that **the worst thing that can happen to a code base is size.**
>
> – Steve Yegge



Typesafe

Outline

Typesafe

Outline

- Example: Akka

Typesafe

Outline

- Example: Akka
- Code & Maintenance

Typesafe

Outline

- Example: Akka
- Code & Maintenance
- How our minds fight bit rot

Typesafe

Outline

- Example: Akka
- Code & Maintenance
- How our minds fight bit rot
- Tools

Typesafe

Outline

- Example: Akka
- Code & Maintenance
- How our minds fight bit rot
- Tools
- Pro tips

Outline

- Example: Akka
- Code & Maintenance
- How our minds fight bit rot
- Tools
- Pro tips
- Summary

“

Build powerful,
concurrent & distributed
applications more easily.

”

Typesafe

Akka stats

Akka stats

- Number of modules: **19**

Typesafe

Akka stats

- Number of modules: **19**
- First Commit: Feb 2009

Typesafe

Akka stats

- Number of modules: **19**
- First Commit: Feb 2009
- Total Commits: **13,450**

Typesafe

Akka stats

- Number of modules: **19**
- First Commit: Feb 2009
- Total Commits: **13,450**
- Total contributors: 135

Akka stats

- Number of modules: **19**
- First Commit: Feb 2009
- Total Commits: **13,450**
- Total contributors: 135
- Est. effort: 36 years (COCOMO)

Akka stats

- Number of modules: **19**
- First Commit: Feb 2009
- Total Commits: **13,450**
- Total contributors: 135
- Est. effort: 36 years (COCOMO)
- Mostly written in **Scala**

http://www.ohloh.net/p/akka

Typesafe

Akka stats

Typesafe

Akka stats

- Total lines: 203,515

Akka stats

- Total lines: 203,515
- Code lines: **142,894**

Akka stats

- Total lines: 203,515
- Code lines: **142,894**
- Percent Code Lines 70.2%

Akka stats

- Total lines: 203,515
- Code lines: **142,894**
- Percent Code Lines 70.2%
- Number of Languages: 11

Typesafe

Akka stats

- Total lines: 203,515
- Code lines: **142,894**
- Percent Code Lines 70.2%
- Number of Languages: 11
- Total Comment Lines: 36,416

Akka stats

- Total lines: 203,515
- Code lines: **142,894**
- Percent Code Lines 70.2%
- Number of Languages: 11
- Total Comment Lines: 36,416
- Percent Comment Lines: 17.9%

Akka stats

- Total lines: 203,515
- Code lines: **142,894**
- Percent Code Lines 70.2%
- Number of Languages: 11
- Total Comment Lines: 36,416
- Percent Comment Lines: 17.9%
- Total Blank Lines: 24,205

Akka stats

- Total lines: 203,515
- Code lines: **142,894**
- Percent Code Lines 70.2%
- Number of Languages: 11
- Total Comment Lines: 36,416
- Percent Comment Lines: 17.9%
- Total Blank Lines: 24,205
- Percent Blank Lines: 11.9%

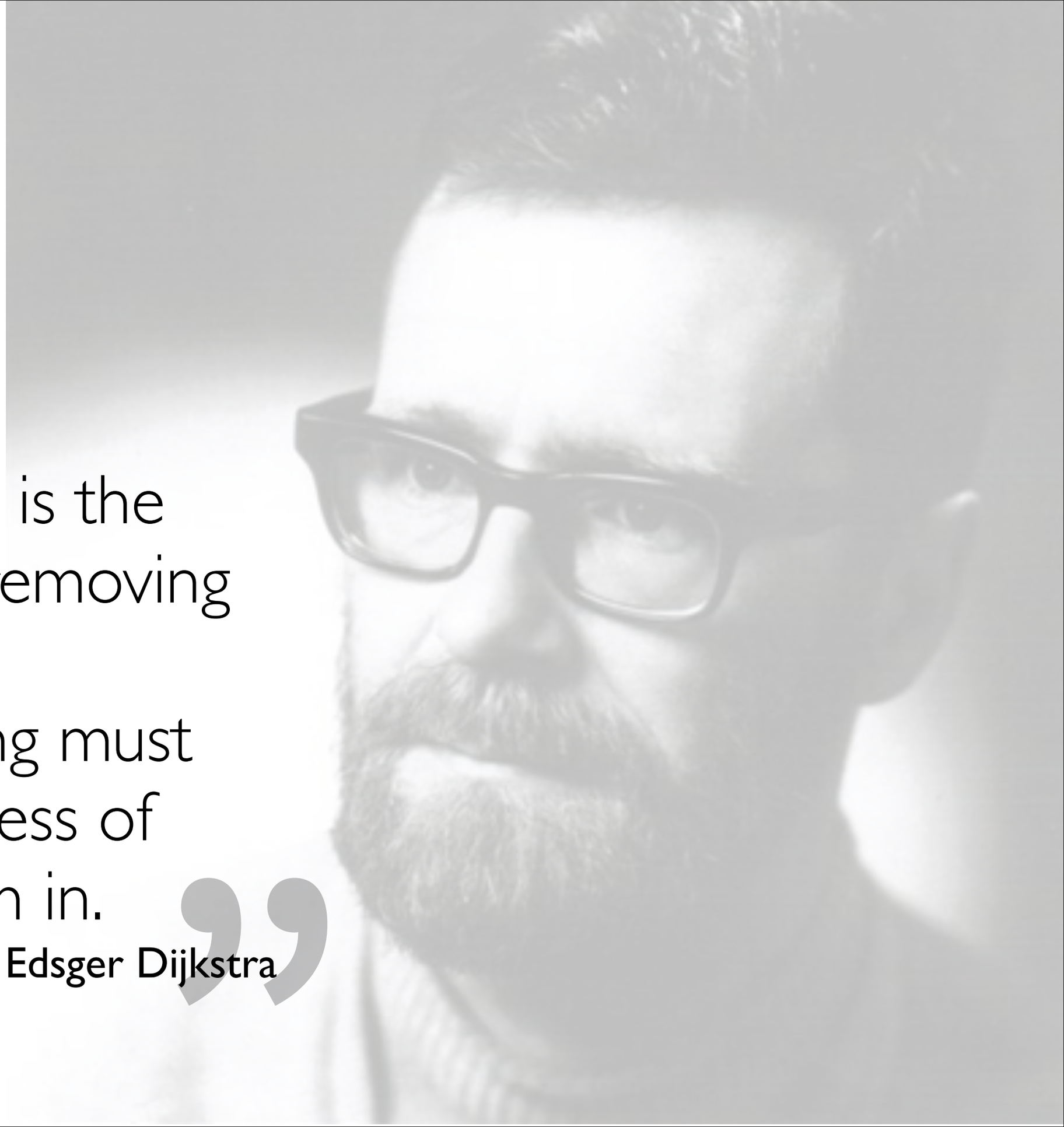
Typesafe

#1 top Github pull award

Typesafe

#1 AWARd gH TDP PILLS

Typesafe



> If debugging is the process of removing bugs, then programming must be the process of putting them in.
>
> – Edsger Dijkstra

Typesafe

Maintenance

Typesafe

Maintenance

- Adaptive – evolve with new surroundings

Typesafe

Maintenance

- Adaptive – evolve with new surroundings
- Perfective – evolve to meet new needs

Typesafe

Maintenance

- Adaptive – evolve with new surroundings
- Perfective – evolve to meet new needs
- Corrective – diagnose and fix defects

Typesafe

Maintenance

- Adaptive – evolve with new surroundings
- Perfective – evolve to meet new needs
- Corrective – diagnose and fix defects
- Preventive – refactor etc

Typesafe

Maintenance

- Adaptive – evolve with new surroundings
- Perfective – evolve to meet new needs
  - Corrective
  - Preventive

Typesafe

[…] a good programmer can reasonably maintain about **20,000** lines of code.

[…] language doesn't matter. It's still 20,000 lines.

If your language requires fewer lines to express the same ideas, you can spend more time on stuff that otherwise would go beyond those 20,000 lines.

– Guido van Rossum
Creator of the Python Programming Language

So ... how do you fight bit rot with your mind?

http://en.wikipedia.org/wiki/Broken_windows_theory

Typesafe

Zero Known Defects

Policy

Typesafe

My point today is that, **if we wish to count lines of code, we should not regard them as "lines produced" but as "lines spent"**:

the current conventional wisdom is so foolish as to book that count on the wrong side of the ledger.

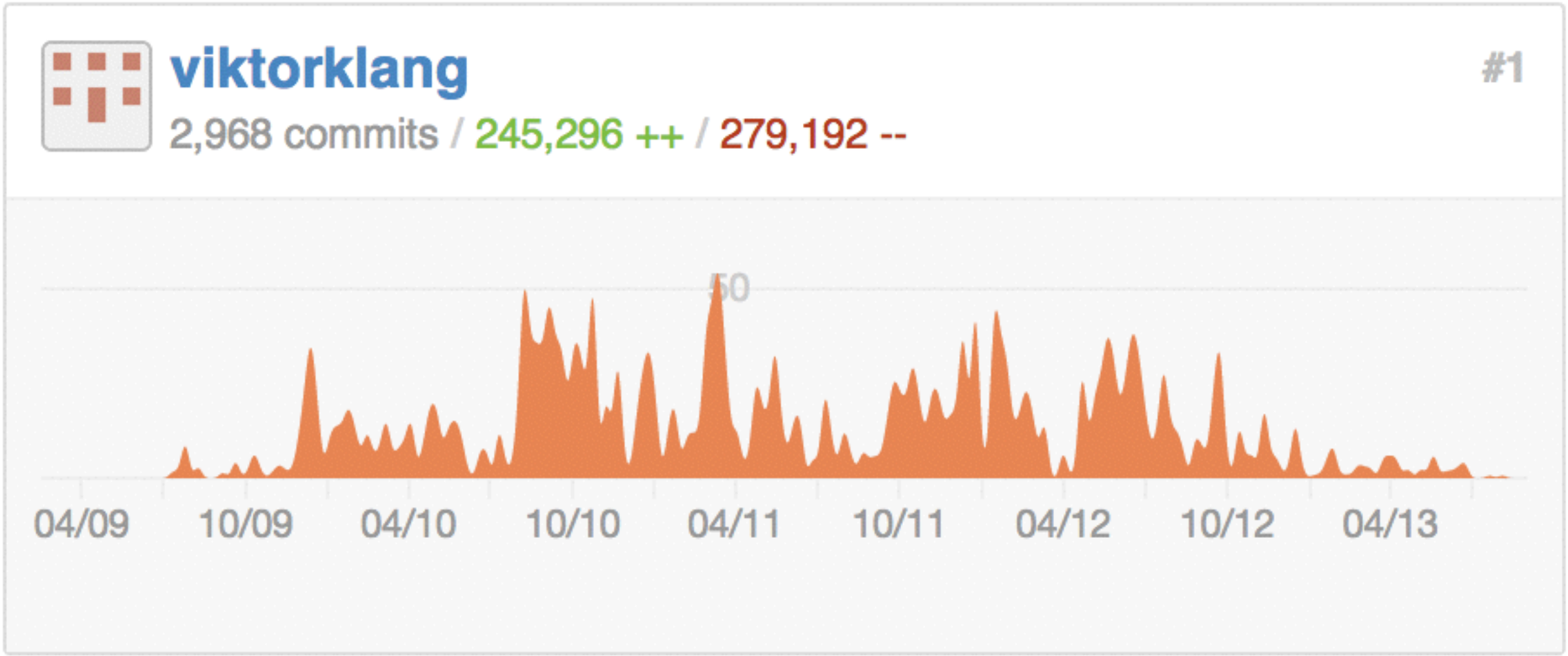
– Edsger Dijkstra

Typesafe

Lines of Code Spent

Typesafe

Less is more!



Typesafe

(My) Test philosophy

Typesafe

(My) Test philosophy

- Feature – **must have**

Typesafe

(My) Test philosophy

- Feature – **must have**
- Integration – **should have**

Typesafe

(My) Test philosophy

- Feature – **must have**
- Integration – **should have**
- Unit – **may have**

Typesafe

Laws

Typesafe

Laws

- DRY

Typesafe

Laws

- DRY
- Boy Scout Rule

Laws

- DRY
- Boy Scout Rule
- Test

Typesafe

Laws

- DRY
- Boy Scout Rule
- Test
- Document

Typesafe

Everyone code reviews

Typesafe

Everyone code reviews

- Collective Ownership

Typesafe

Everyone code reviews

- Collective Ownership
- Harmonization

Typesafe

Everyone code reviews

- Collective Ownership
- Harmonization
- Education

Typesafe

Everyone code reviews

- Collective Ownership
- Harmonization
- Education
- Documentation

Typesafe

Everyone code reviews

- Collective Ownership
- Harmonization
- Education
- Documentation
- Test Coverage

Typesafe

Documentation

Typesafe

Documentation

- Reference documentation

Typesafe

Documentation

- Reference documentation
- API documentation

Typesafe

Documentation

- Reference documentation
- API documentation
- Put "Whys" in the code

Typesafe

Solve everything at least twice

Typesafe

Typesafe

git ~~blame~~ / praise

Typesafe

Rules

Typesafe

Rules

- Code formatting

Rules

- Code formatting
- Tabs vs spaces

Rules

- Code formatting
- Tabs vs spaces
- Features

Typesafe

Rules

- Code formatting
- Tabs vs spaces
- Features
- Commits

Typesafe

Rules

- Code formatting
- Tabs vs spaces
- Features
- Commits
  - Description

Typesafe

Rules

- Code formatting
- Tabs vs spaces
- Features
- Commits
  - Description
  - Message

Typesafe

Auto-enforce

Typesafe

Auto-enforce

- Reformat on save / compile

Typesafe

Auto-enforce

- Reformat on save / compile
- Post-commit hooks

Typesafe

Features

- What features do we use when and why?

Typesafe

Common Dev Tools

Typesafe

Common Dev Tools

- Issue tracker

Typesafe

Common Dev Tools

- Issue tracker
- Code reviewer

Typesafe

Common Dev Tools

- Issue tracker
- Code reviewer
- Continuous Integration

Typesafe

Common Dev Tools

- Issue tracker
- Code reviewer
- Continuous Integration
- Build tool

Typesafe

Common Dev Tools

- Issue tracker
- Code reviewer
- Continuous Integration
- Build tool
- Source Control

Typesafe

Common Dev Tools

- Issue tracker
- Code reviewer
- Continuous Integration
- Build tool
- Source Control
- Documentation Generator

Typesafe

Issue tracking

Typesafe

Issue tracking

- Absolute priority ordering

Typesafe

Issue tracking

- Absolute priority ordering
- Break down into half-day size

Typesafe

Issue tracking

- Absolute priority ordering
- Break down into half-day size
- Follow motivation when choosing

Typesafe

Source Related

Typesafe

Source Related

- Code review tool

Typesafe

Source Related

- Code review tool
  - Use the one that works best

Source Related

- Code review tool
  - Use the one that works best
- Continuous Integration tool

Source Related

- Code review tool
  - Use the one that works best
- Continuous Integration tool
  - Use the one that works best

Typesafe

Source Related

- Code review tool
  - Use the one that works best
- Continuous Integration tool
  - Use the one that works best
- Build tool

Source Related

- Code review tool
  - Use the one that works best
- Continuous Integration tool
  - Use the one that works best
- Build tool
  - Use the one that works best

Typesafe

Source Related

- Code review tool
  - Use the one that works best
- Continuous Integration tool
  - Use the one that works best
- Build tool
  - Use the one that works best
- Source Control

Typesafe

Source Related

- Code review tool
  - Use the one that works best
- Continuous Integration tool
  - Use the one that works best
- Build tool
  - Use the one that works best
- Source Control
  - Use Git and be done with it!

Typesafe

Continuous Integration

Typesafe

Continuous Integration

- Run CI on Pull Requests before merge

Typesafe

Continuous Integration

- Run CI on Pull Requests before merge
- Run CI continuously on main branches

Typesafe

Documentation Generator

Typesafe

Documentation Generator

- All code samples should be tests that are compiled and run and automagically lifted in

Typesafe

Documentation Generator

- All code samples should be tests that are compiled and run and automagically lifted in
- Documentation generation should be a part of CI so if it fails, it fails the build

Typesafe

Documentation Generator

- All code samples should be tests that are compiled and run and automagically lifted in
- Documentation generation should be a part of CI so if it fails, it fails the build
- Generate at least HTML + PDF

Typesafe

Individual Tools

Typesafe

Editors/IDEs

- Build tool will enforce code style so use the editor you are most productive in!

Typesafe

HW Interfaces

HW Interfaces

- Don't change keyboards/layouts to write faster

Typesafe

HW Interfaces

- Don't change keyboards/layouts to write faster
  - Do it to avoid strain on your fingers and wrists

Typesafe

HW Interfaces

- Don't change keyboards/layouts to write faster
  - Do it to avoid strain on your fingers and wrists

10: Pain leads to shortcuts

Typesafe

HW Interfaces

- Don't change keyboards/layouts to write faster
  - Do it to avoid strain on your fingers and wrists

10: Pain leads to shortcuts

20: Shortcuts lead to bugs

Typesafe

HW Interfaces

- Don't change keyboards/layouts to write faster
  - Do it to avoid strain on your fingers and wrists

10: Pain leads to shortcuts

20: Shortcuts lead to bugs

30: Bugfixing means typing

Typesafe

HW Interfaces

- Don't change keyboards/layouts to write faster
  - Do it to avoid strain on your fingers and wrists
    - 10: Pain leads to shortcuts
    - 20: Shortcuts lead to bugs
    - 30: Bugfixing means typing
    - 40: **goto** 10

Typesafe

#define Protips

Typesafe

Dependencies

Dependencies

- Have a cost associated with using them

Dependencies

- Have a cost associated with using them
  - License management

Typesafe

Dependencies

- Have a cost associated with using them
  - License management
  - Repository and / or package size

Typesafe

Dependencies

- Have a cost associated with using them
  - License management
  - Repository and / or package size
  - Transitive dependencies

Dependencies

- Have a cost associated with using them
  - License management
  - Repository and / or package size
  - Transitive dependencies
    - Version conflicts

Typesafe

Dependencies

- Have a cost associated with using them
  - License management
  - Repository and / or package size
  - Transitive dependencies
    - Version conflicts
- You cannot abstract away understanding

Typesafe

Dependencies

- Have a cost associated with using them
  - License management
  - Repository and / or package size
  - Transitive dependencies
    - Version conflicts
- You cannot abstract away understanding

— only delegate responsibility

Typesafe

Planned interfaces

Typesafe

Planned interfaces

- Design and document your public API

Typesafe

Planned interfaces

- Design and document your public API
- Don't make things **API by accident**

Typesafe

Planned interfaces

- Design and document your public API
- Don't make things **API by accident**
- Public API needs to have proper documentation

Typesafe

Planned interfaces

- Design and document your public API
- Don't make things **API by accident**
- Public API needs to have proper documentation
- Good API design takes a lot of effort

Typesafe

Planned interfaces

- Design and document your public API
- Don't make things **API by accident**
- Public API needs to have proper documentation
- Good API design takes a lot of effort
  - **But is worth it!**

Typesafe

Avoid defaults in code

Typesafe

Avoid defaults in code

- Put defaults in external configuration

Typesafe

Avoid defaults in code

- Put defaults in external configuration
- Possible to change without recompiling

Typesafe

Avoid defaults in code

- Put defaults in external configuration
- Possible to change without recompiling
- Makes structure of the code cleaner

Typesafe

Avoid defaults in code

- Put defaults in external configuration
- Possible to change without recompiling
- Makes structure of the code cleaner
- Avoids scattering same "default" in multiple places (DRY)

Typesafe

Avoid shared mutable state

Typesafe

Avoid shared mutable state

- Consequences of shared mutable state:

Typesafe

Avoid shared mutable state

- Consequences of shared mutable state:
  - Interactions are hard to calculate

Typesafe

Avoid shared mutable state

- Consequences of shared mutable state:
  - Interactions are hard to calculate
  - Almost always assumes single-threaded execution

Avoid shared mutable state

- Consequences of shared mutable state:
  - Interactions are hard to calculate
  - Almost always assumes single-threaded execution
  - Hard to reason and verify impact of changes

Typesafe

All the possibilities!

Typesafe

All the possibilities!

```
public int mutable() {
  int x = 1;
  int y = 2;
  int z = 3;
  ...
  return x + y + z;
}
```

Typesafe

All the possibilities!

```
public int mutable() {
  int x = 1;
  int y = 2;
  int z = 3;
  ...
  return x + y + z;
}
```

```
public int immutable() {
  final int x = 1;
  final int y = 2;
  final int z = 3;
  ...
  return x + y + z;
}
```

Typesafe

Immutable

Immutable

- Things that do not change frees the mind from calculating the interactions

Typesafe

Immutable

- Things that do not change frees the mind from calculating the interactions
- Use **const** / **final** whenever possible

Typesafe

Immutable

- Things that do not change frees the mind from calculating the interactions
- Use **const** / **final** whenever possible
- Favor immutability in collections and data

Typesafe

Immutable

- Things that do not change frees the mind from calculating the interactions
- Use **const** / **final** whenever possible
- Favor immutability in collections and data
  - Persistent collections can be very efficient

Expressions > Statements

Typesafe

Expressions > Statements

- Statements do not "**return**" anything

Typesafe

Expressions > Statements

- Statements do not "**return**" anything
  - They can only produce results by writing to shared memory

Typesafe

Expressions > Statements

- Statements do not "**return**" anything
  - They can only produce results by writing to shared memory
  - This leads to having a lot of variables

Typesafe

Expressions > Statements

- Statements do not "**return**" anything
  - They can only produce results by writing to shared memory
  - This leads to having a lot of variables
  - Remember what we said about immutability

Expressions > Statements

Typesafe

Expressions > Statements

```
public int foo() {
  int x;
  if (smth) {
    x = 5;
  } else {
    x = 10;
  }
  ...
  return result;
}
```

Typesafe

Expressions > Statements

```
public
  int
  if
    x = 5;
  }
    x = 10;
  }
  …
  return
}
```

```
public int foo() {
  int x;
  if (smth) x = 5;
  else x = 10;
  …
  return result;
}
```

Typesafe

Expressions > Statements

```
public
  int
  if
    x = 5;
  }
    x = 10;
  }
  ...
  return
}
```

```
public
  int
  if
  else
  ...
  return
}
```

```
public int foo() {
  final int x =
    smth ? 5 : 10;
  ...
  return result;
}
```

Typesafe

Versioning

v1 . . . v5

Typesafe

Versioning

v1 . . . v5

- Very few enjoy designing versioning

Typesafe

Versioning

v1 . . . v5

- Very few enjoy designing versioning
- Very few enjoy evolving an unversioned system

Typesafe

Versioning

v1 . . . v5

- Very few enjoy designing versioning
- Very few enjoy evolving an unversioned system
- Guess which one of these usually lasts longer

Typesafe

Bad comments

Typesafe

Bad comments

- Comments also rot

Typesafe

Bad comments

- Comments also rot
- Never commit commented out code.

If it isn't worth compiling, it's not worth maintaining.

Typesafe

Don't Copy Paste

Typesafe

Don't Copy Paste

- Copy-paste violates DRY

Typesafe

Don't Copy Paste

- Copy-paste violates DRY
- Use Cut-paste

Typesafe

Don't Copy Paste

- Copy-paste violates DRY
- Use Cut-paste
- If you Cut-paste-paste-*

Typesafe

Don't Copy Paste

- Copy-paste violates DRY
- Use Cut-paste
- If you Cut-paste-paste-*
  - **Stop it**

Typesafe

Failures will occur

Failures will occur

- Fault tolerance impacts system design

Typesafe

Failures will occur

- Fault tolerance impacts system design
  - Think about failures up front

Typesafe

Failures will occur

- Fault tolerance impacts system design
- Think about failures up front
- Avoid mixing concerns in code

Failures will occur

- Fault tolerance impacts system design
  - Think about failures up front
  - Avoid mixing concerns in code
- Overload is a type of failure

Failures will occur

- Fault tolerance impacts system design
  - Think about failures up front
  - Avoid mixing concerns in code
- Overload is a type of failure
  - How should the system behave?

Typesafe

Pride in work

Typesafe

∑-ary

Typesafe

∑-ary

- Creating successful "bit rot" resistant software boils down to:

Typesafe

Σ-ary

- Creating successful "bit rot" resistant software boils down to:
  - Culture

Typesafe

Σ-ary

- Creating successful "bit rot" resistant software boils down to:
  - Culture
  - Process

Σ-ary

- Creating successful "bit rot" resistant software boils down to:
  - Culture
  - Process
  - Practices

Typesafe

Σ-ary

- Creating successful "bit rot" resistant software boils down to:
  - Culture
  - Process
  - Practices
- And last but not least:

Σ-ary

- Creating successful "bit rot" resistant software boils down to:
  - Culture
  - Process
  - Practices
- And last but not least:
  - **Continuously improving them**

“Time is the fire
in which we burn
– Delmore Schwarz
”

Typesafe

EOF