



## Preparing for the Judges: How mature is my Hackathon prototype?

Great software doesn't just happen

April 11<sup>th</sup>, 2019

GETTING SOFTWARE RIGHT

## 2018 vs 2019

Who was here last year?

→ Experiences & Remarks

What is different this year?

→ Grill sessions by JEDI

→ 'The Mature Prototype'



Software Improvement Group / PUBLIC

M

SEARCH 9 🔍  

 LATEST TOP COMMUNITY | PARTNER WITH HACKER NOON

# Writing quality code under time pressure?

These hackathon teams prove it can be done

 Joost Visser  
Apr 12, 2018 · 7 min read



# Introductions



**Luc Brandts**

CTO

@lucbrandts



**Michiel Cuijpers**

Manager Academy

Community manager BetterCodeHub

@michielcuijpers



**Software Improvement Group**

IT Management Advisory and Measurement Lab

- ISO 25010 software product inspections
- ISO 17025 lab analyses 25 million lines of code each week
- Translate technical findings into actionable recommendations

## Agenda today:

- 1 Introduction Software Improvement Group
- 2 What is Maintainable code any way? And why should we care?
  - <https://odysseyhack.github.io/code-quality-matters>
  - The 10 guidelines
- 3 How to get a mature prototype certificate
  - <https://odysseyhack.github.io>
  - The Mature Prototype Grill Masters
- 4 Questions?

# About – Software Improvement Group

## GETTING SOFTWARE RIGHT

- > Factual and actionable management advice through source code analysis and structured methods
- > Assessment of a variety of technologies by using technology-independent methods



Impartial, objective mediator between IT suppliers and clients



Active contributor to scientific research in software engineering



Providing management with insight into IT projects and systems by means of facts and measurements



Eligible for TÜViT certification of software systems  
*SIG/TÜViT Maintainability Model*

# 19 years of software quality measurement

a.k.a. The SIGRID platform

**200+**

technologies  
from mainframe to mobile

**2,500**

industry systems evaluated

**225+**

clients

**$10 \times 10^6$**

lines of code analyzed each week

**22,000+**

inspections

**$8.6 \times 10^9$**

lines of code in analysis warehouse

## Some clients of SIG



# Some services of SIG

## Assessments

Support risk-based decision making with in-depth analysis of your software



## Inspections

Profit from regular checks – security, reliability, portability, performance, usability



## Monitoring

Have a grip on quality, architecture, and productivity, at all times



## IT Due Diligence

Secure your investment by upfront identification of technology risks



## Software Delivery Assurance

Ensure a straight path from solution shaping to product delivery



## Training and Certification

Facilitate developers to deliver top achievements

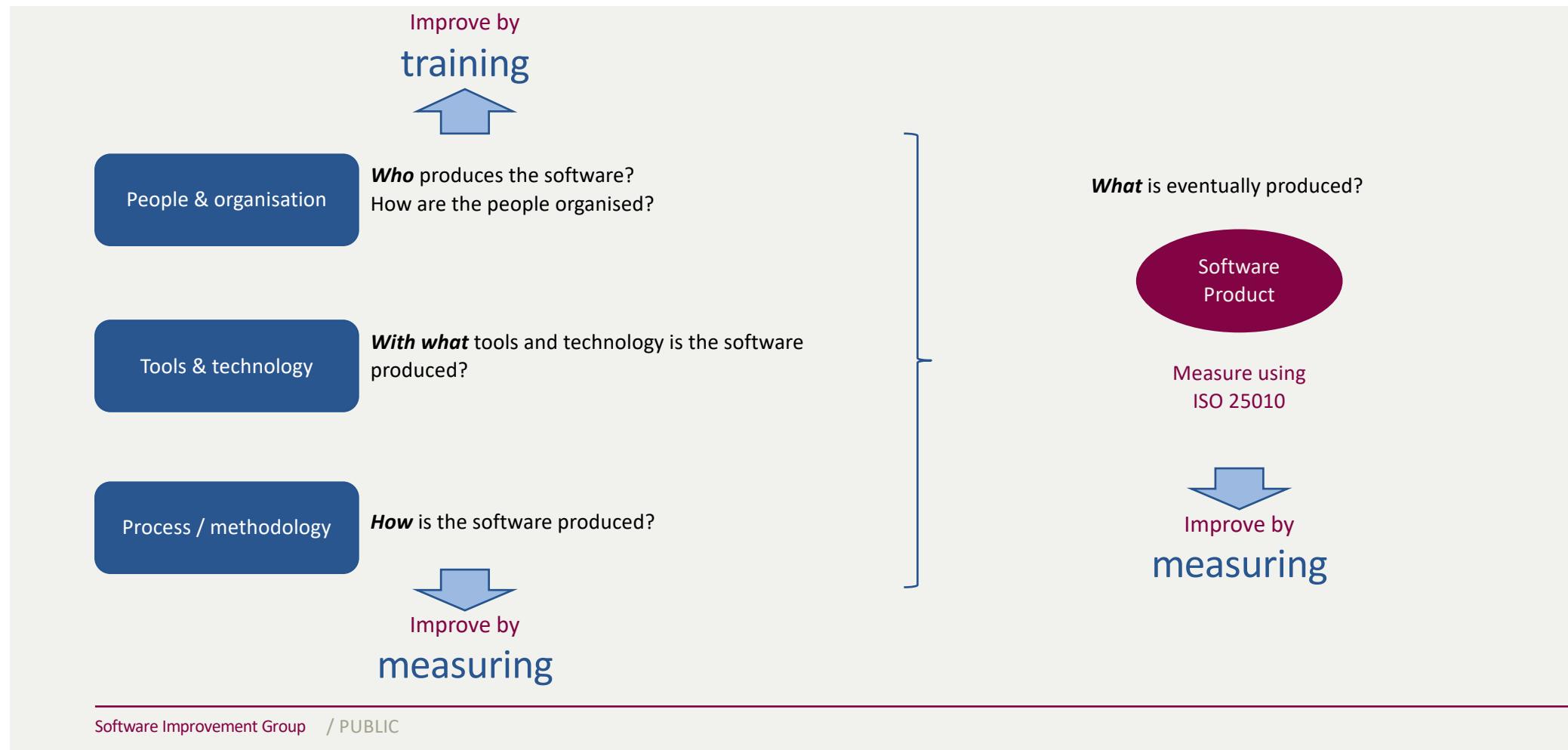


## Relation of code quality – investor value



# Great software doesn't just happen

## Train and Measure



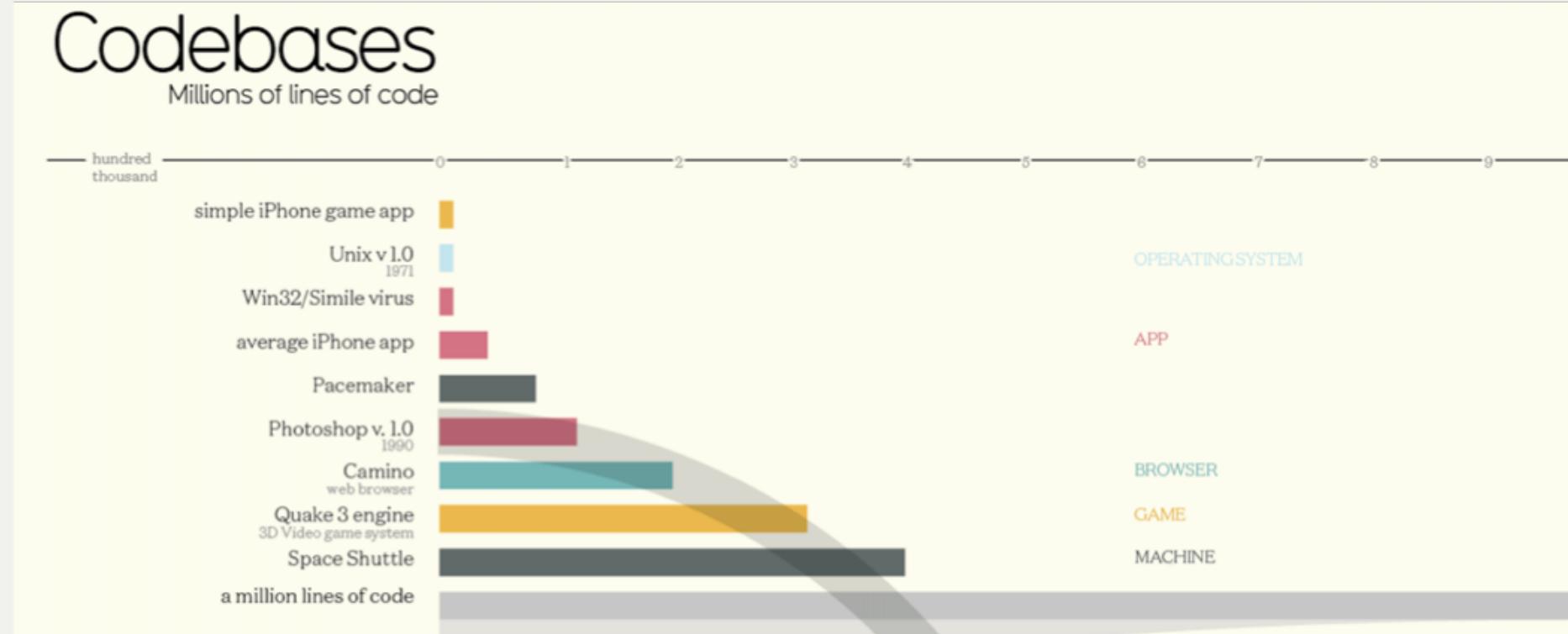
Your biggest system?

**Question:**

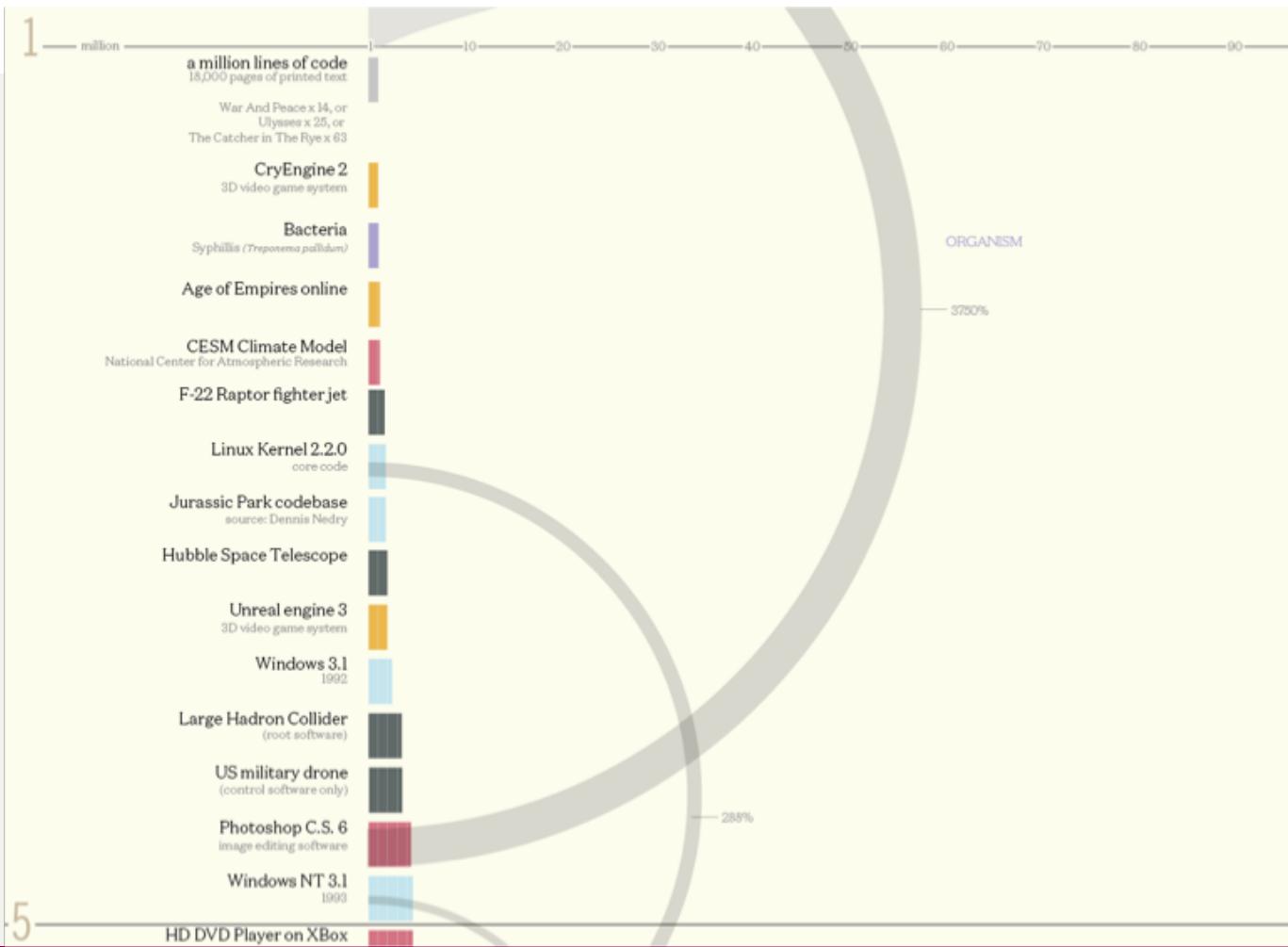
**Any idea about size?**

## About Real software

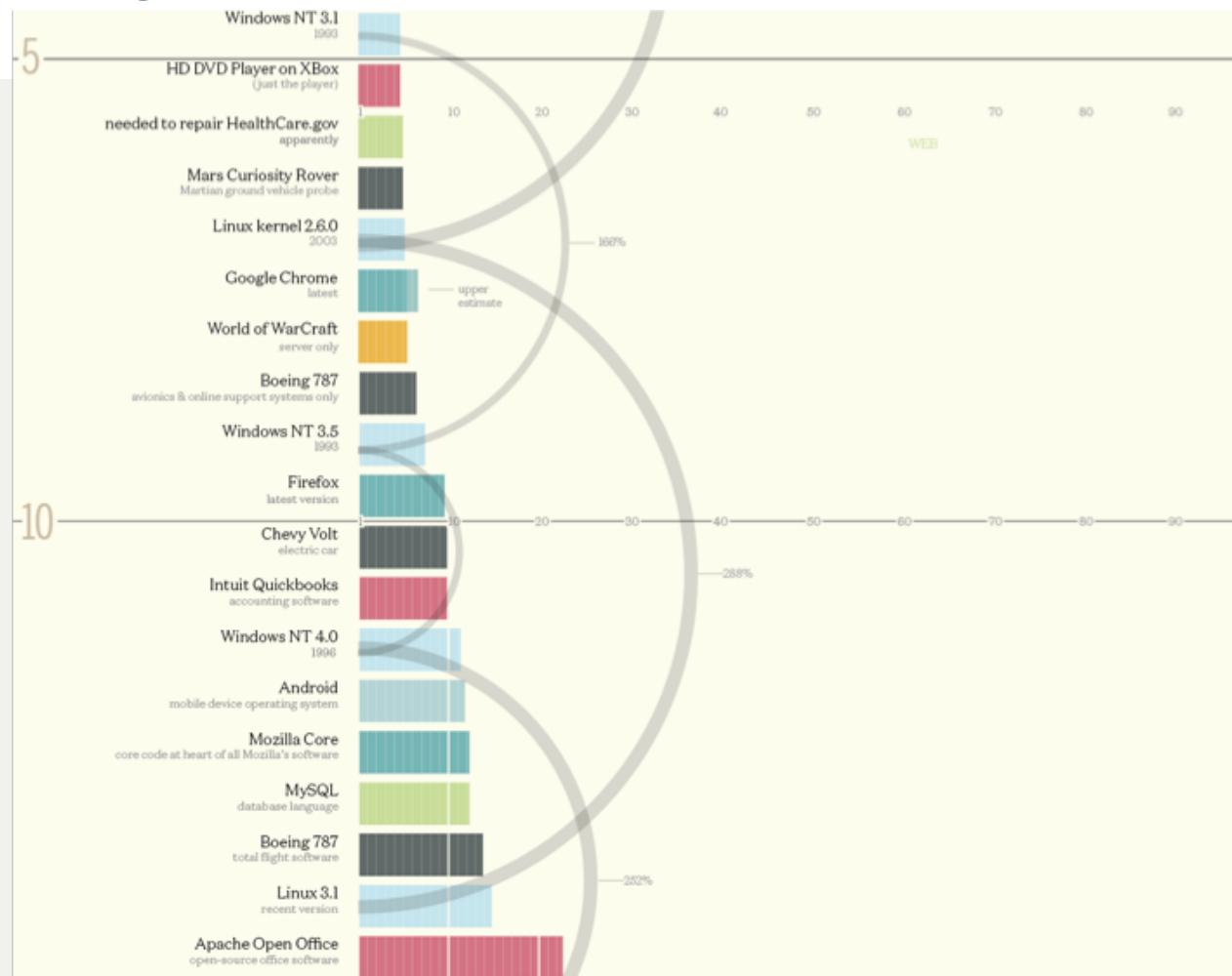
Just to get an idea how big software gets over time



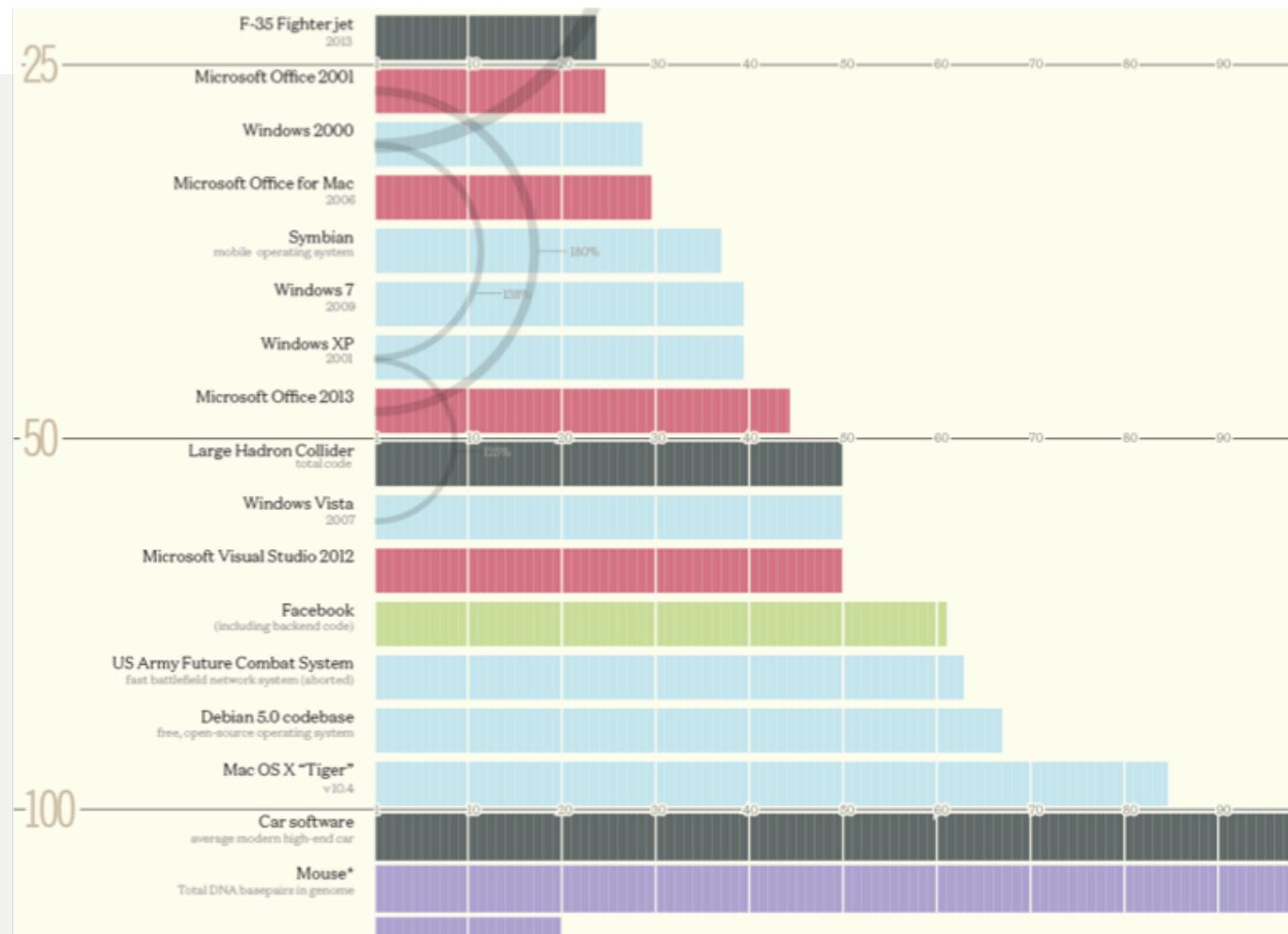
## Real software is very large...



## Real software is huge...



## Real software is absurdly huge...



Software is taking over the world...



## Some interesting statistics

### Job security ..

- > Average programmer productivity:
  - Roughly 45 lines of working Java code per working day
  - Let's say 10.000 LOC/year
- > Operational software ***yearly*** change rate:
  - Roughly 5 to 15% of the code base
- > Average ***yearly*** maintenance effort a high-end consumer car:
  - $100 \text{ MLOC} \times 5 \text{ to } 15\% = 5 \text{ to } 15 \text{ MLOC}$
  - $15 \text{ MLOC} / 10.000 \text{ LOC} / \text{year} = 500 \text{ to } 1.500 \text{ fte worth of maintenance programmers!}$

## Some more interesting statistics

- > In 2020 only 30% software projects are “new”, the rest is “maintenance.”  
(source: Software Productivity Research)
- > 0.1 - 10 bugs per 1,000 lines of code.  
(sources: SEI, Coverity)



## Relevance of code quality – speed

Rich Rogers  
@RichRogersIoT

Following

Current development speed is a function of past development quality. - @brianm



RETWEETS LIKES

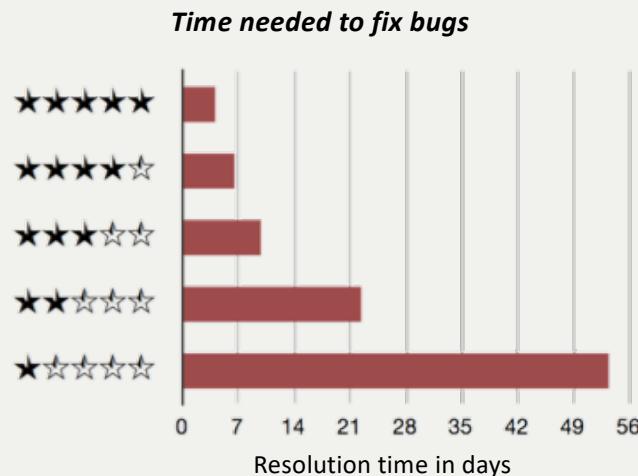
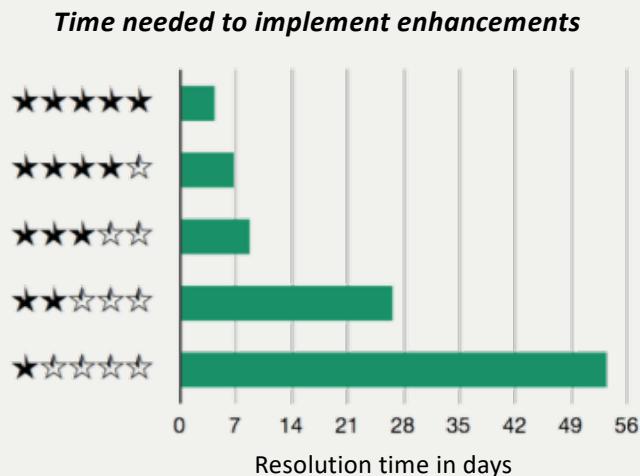
36 29

10:00 AM - 6 Mar 2017

- > Who recognizes this and dares to share?
- > Who has rejected a job because of bad code?

## Correlation between maintainability rating and issue resolution time

Issues can be resolved a factor 3.5 faster on systems of high technical quality



Source: "Dennis Bijlsma, Faster issue resolution with higher technical quality of software", Software Quality Journal, 2011

## What if we could make it easy to write **maintainable** code?



Marco di Biase  
@mardibbiase

Following

@Sharma\_\_Tushar gets it, right? @sig\_eu  
@jstvssr @MiguelBruntink #sattose17

Writing code is easy;  
maintaining it is not.

Desgnite  
www.desgnite-tools.com

RETWEETS LIKES  
7 13

2:07 AM - 8 Jun 2017

Tushar Sharma

“Improving maintainability does not require magic or rocket science. A combination of relatively simple skills and knowledge, plus the discipline and environment to apply them, leads to the largest improvement in maintainability.”  
prof Joost Visser

If you should remember just one thing...

**The quality of (your) code matters &  
should be easy to control!**

But how...?

**Software quality can be measured  
and controlled using tools.**

# Measuring software maintainability

## Foundational concepts

Simple to understand

Everyone with a software engineering background should be able to understand the measurement results.

Objective

The measurement results should not depend on the person performing the measurements.

Allow root-cause analysis

The measurements should give insight into *why* the system is maintainable.

Allow comparing systems

It should be possible to compare the measurement results for different systems, to determine which areas are more/less maintainable.

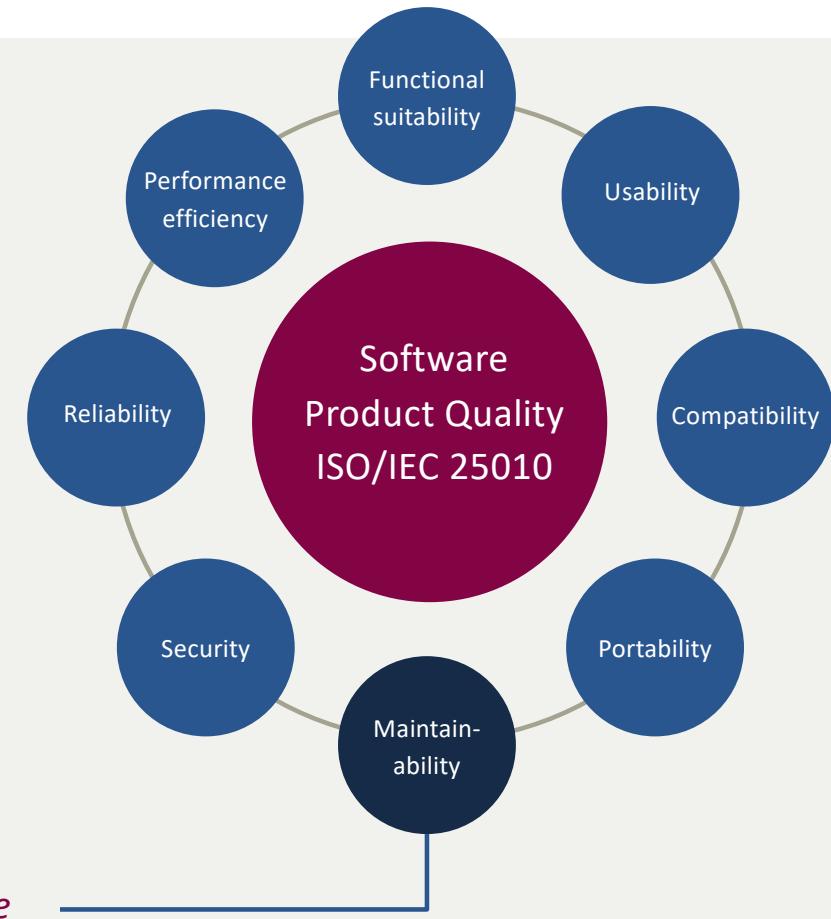
Technology-independent

The measurements should be usable regardless of what technologies were used to implement the system.

## How to define Maintainability?

Sustainable business requires  
maintainable software

Quality  
product – process – people  
measurement – management – education

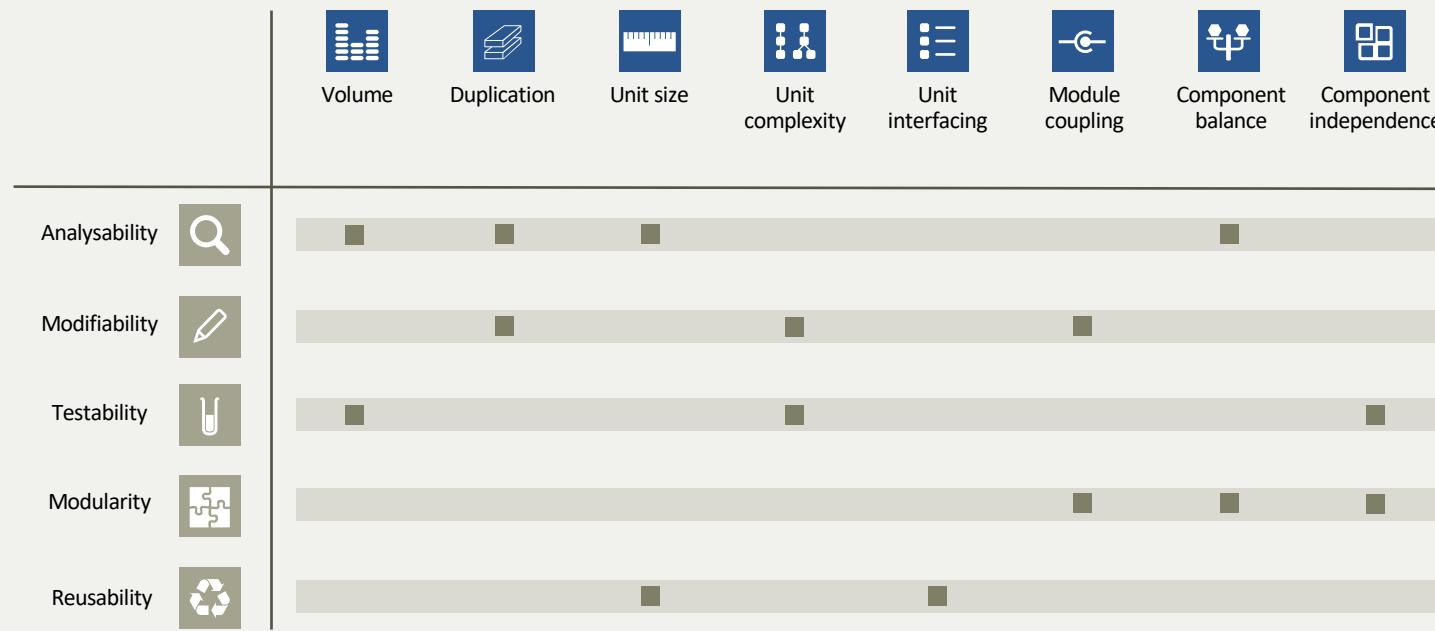


## SIG Quality Model • Maintainability

### Operationalization of ISO 25010 (was: ISO 9126)

First published as “A practical model for measuring maintainability”, Heitlager, Kuiper, Visser, QUATIC 2007.

Most Influential Paper

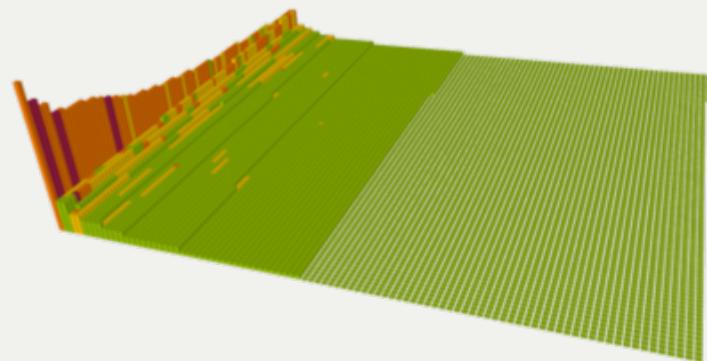


The sub-ratings are aggregated to an overall maintainability rating,  
where ★★★☆☆ is market average.

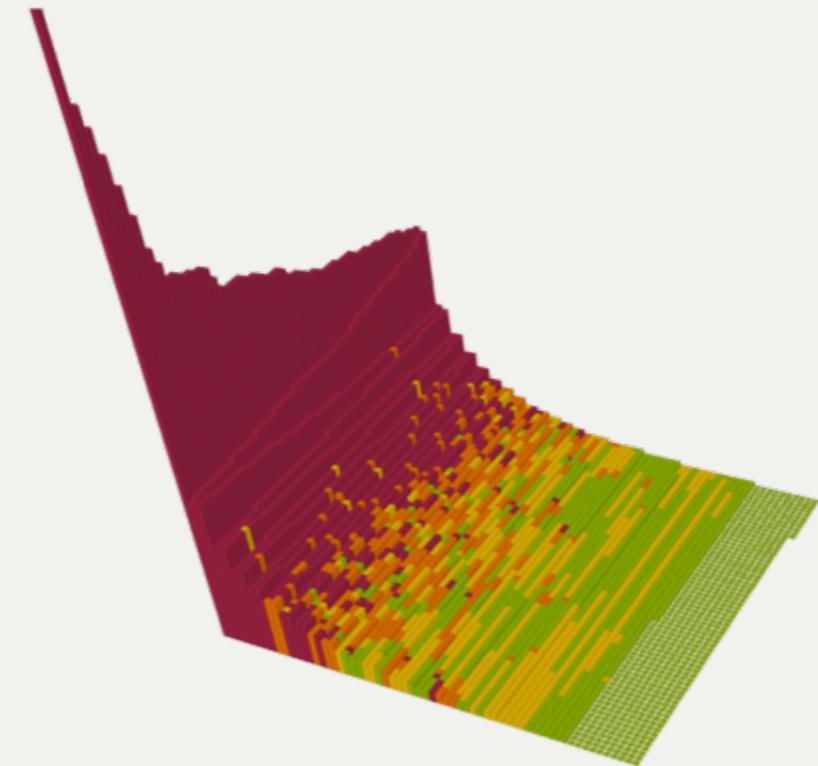
Source code measurements are aggregated in statistically sound “quality profiles” and then rated on a calibrated scale from ★★★☆☆ to ★★★★★.

No codebase is perfect!

But some code bases are more ‘perfect’ than others



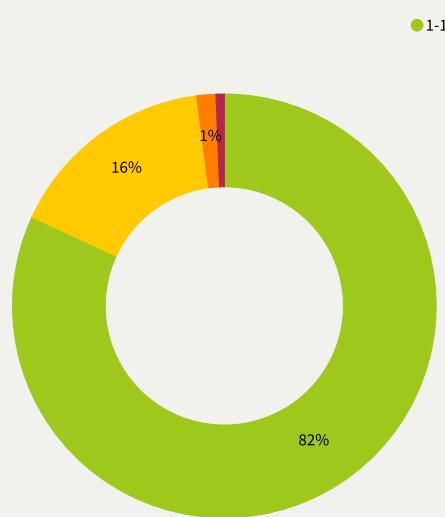
Maintainable



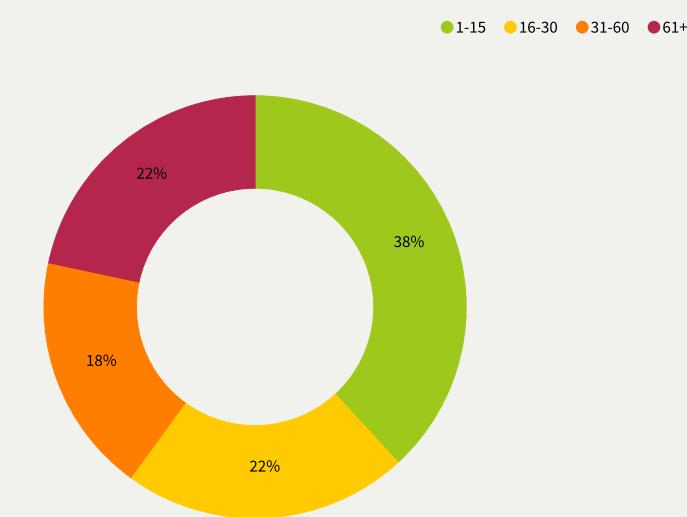
Unmaintainable

## Quality profiles: percentage of code in various risk categories

### Quality profiles for Unit Size



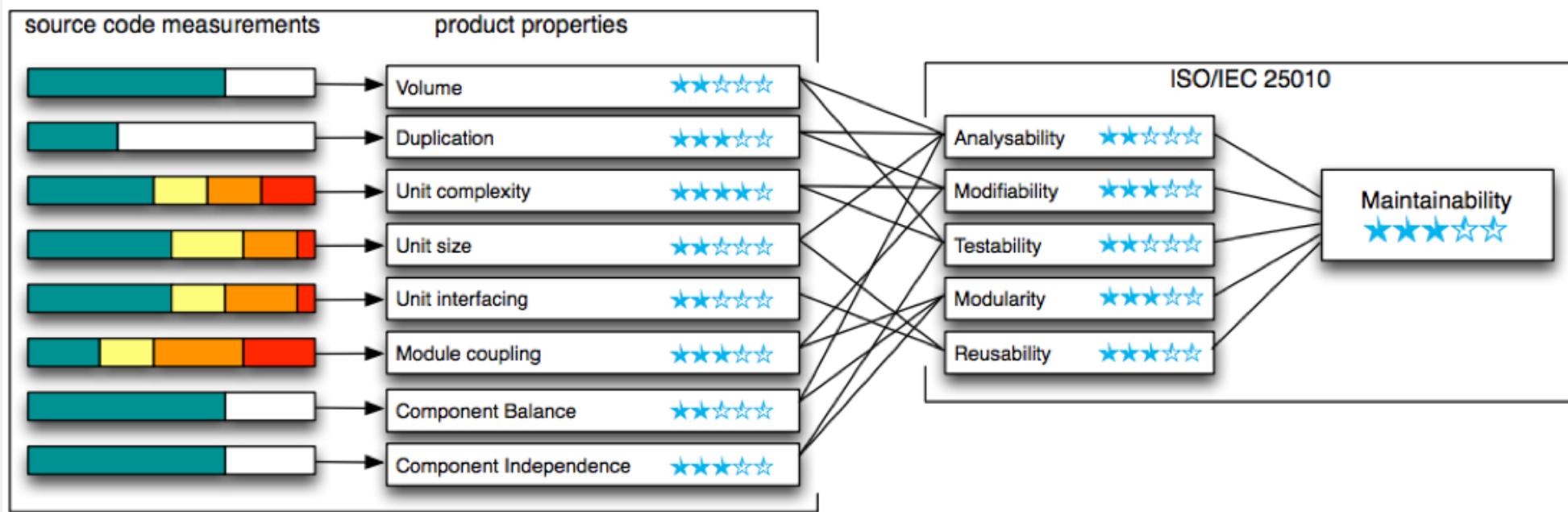
Maintainable



Unmaintainable

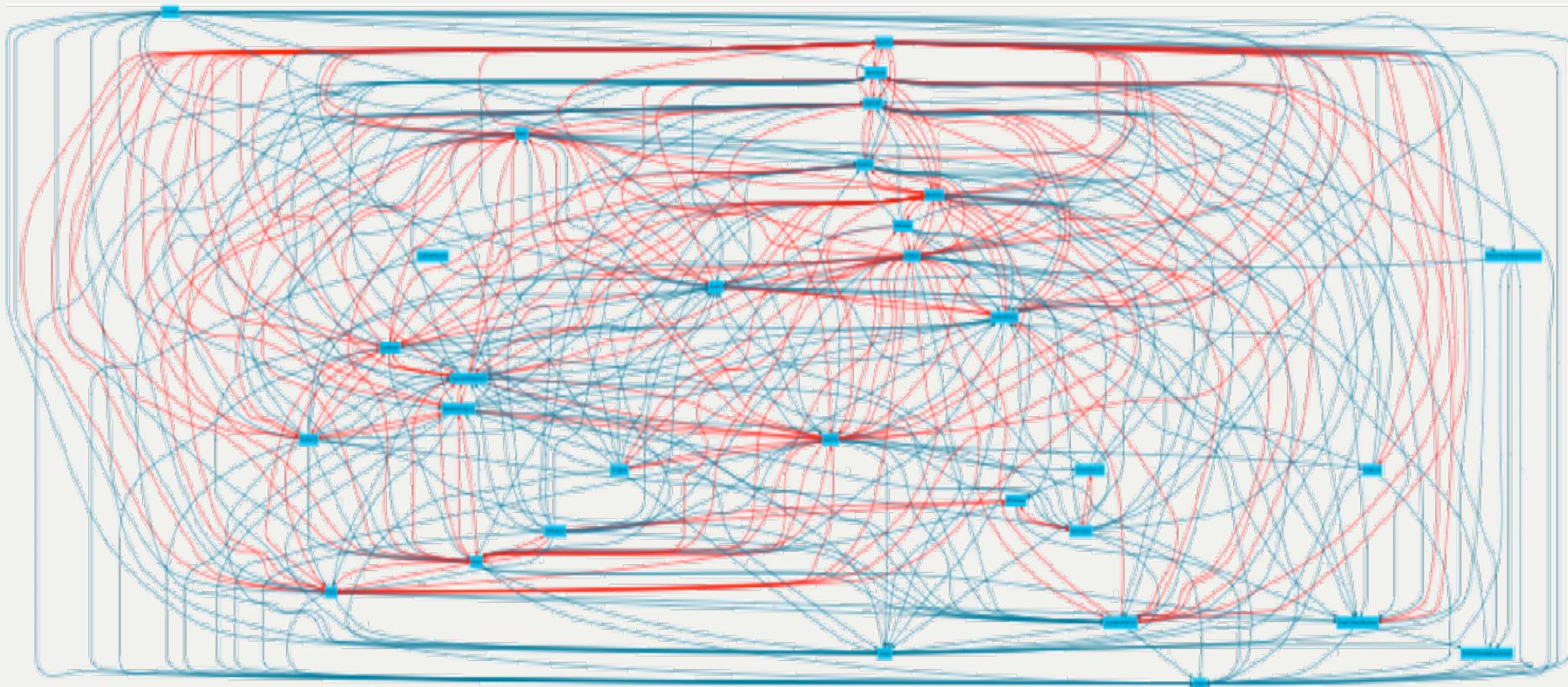
# A star rating for software system maintainability

Software Improvement Group SIGRID platform uses star rating



## What do we see & How did this happen?

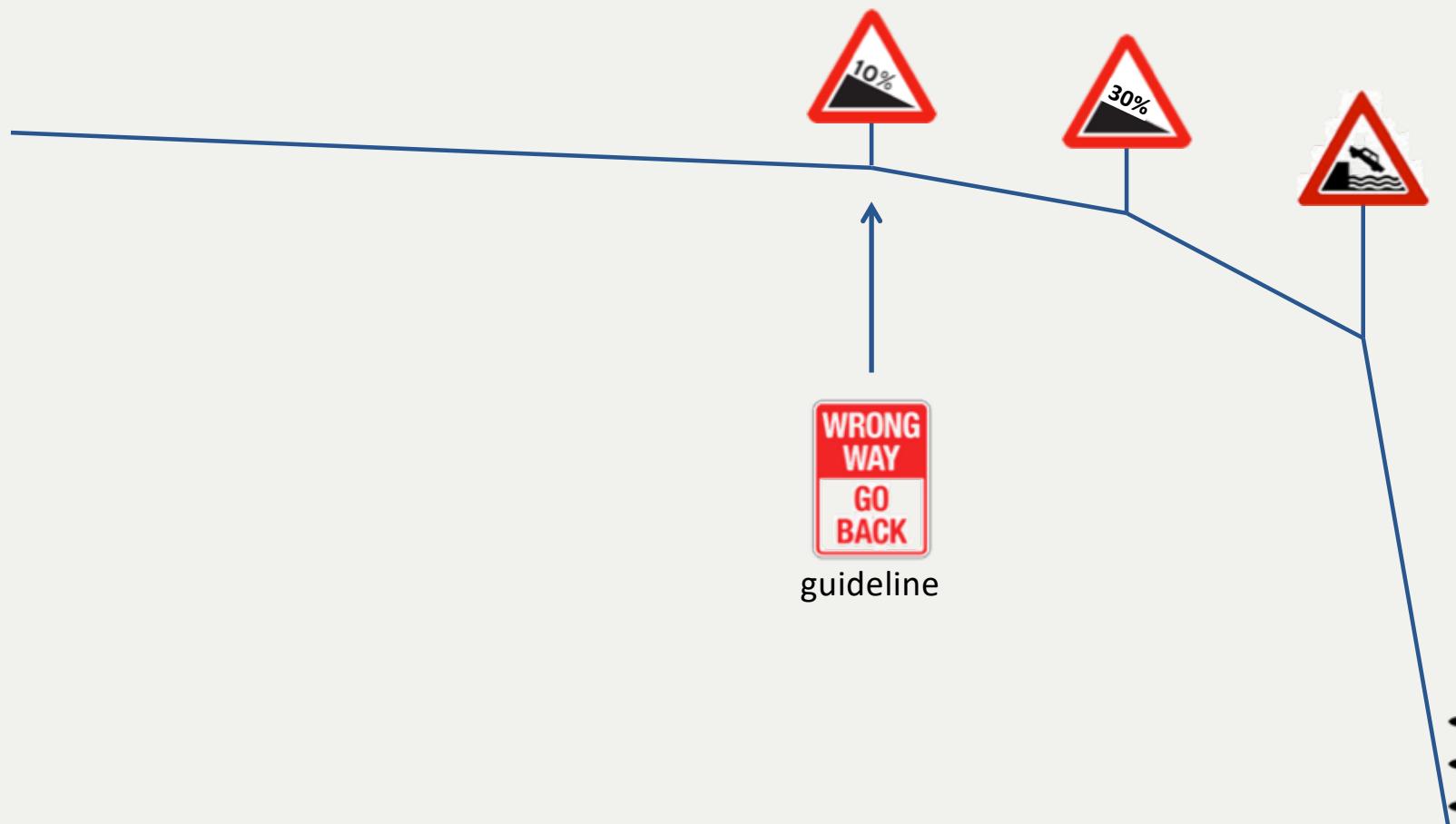
One line of code at a time



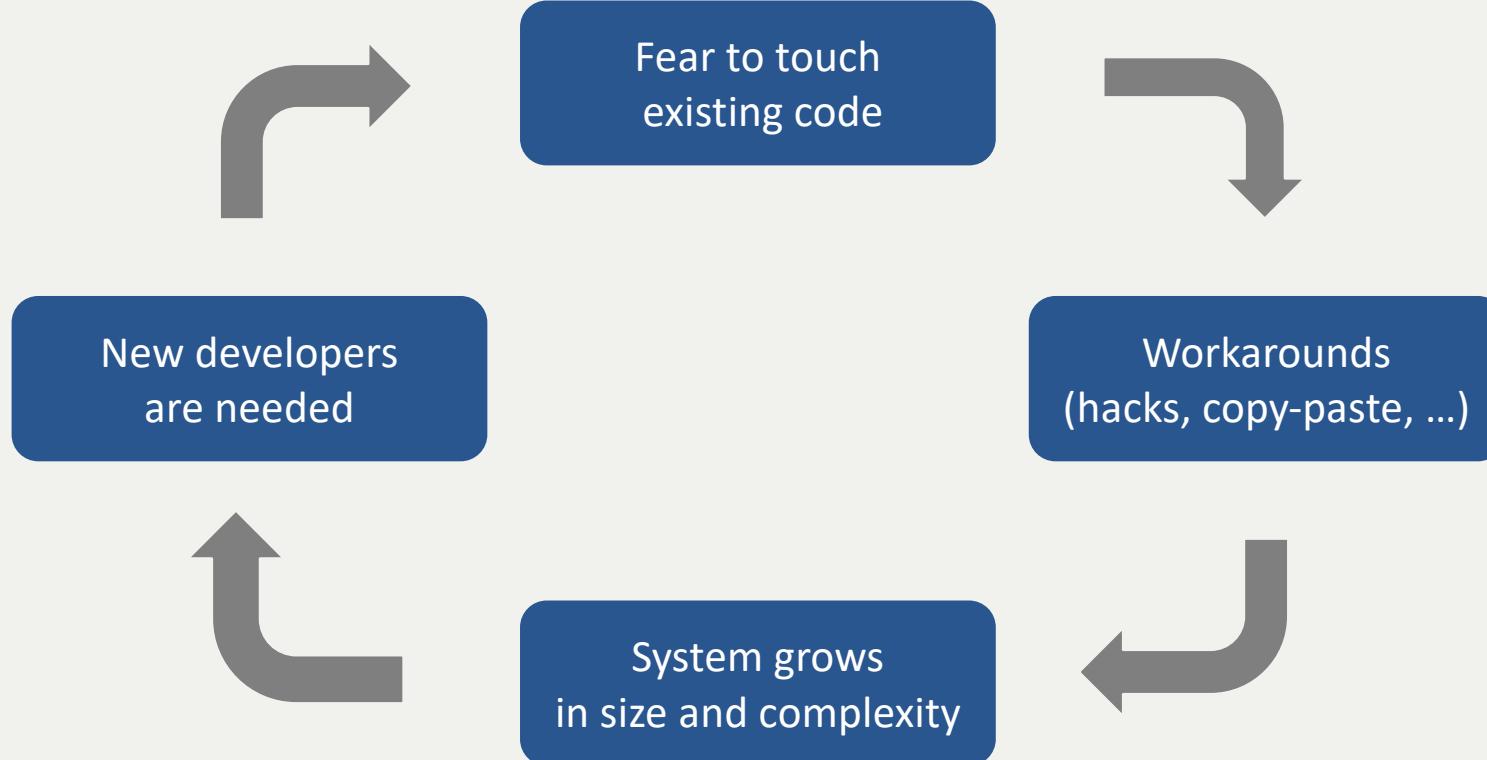
And how did *this* happen?  
Also one line of code at a time



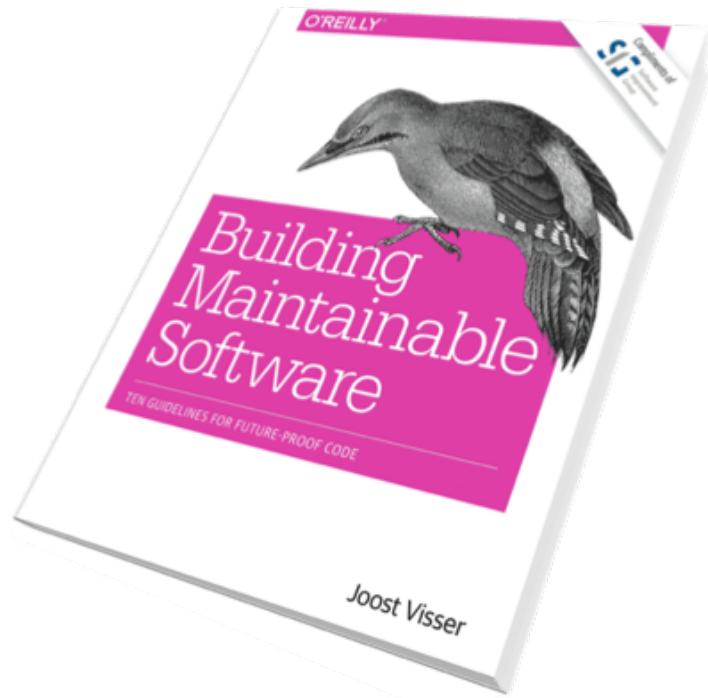
## Identifying Tipping Points



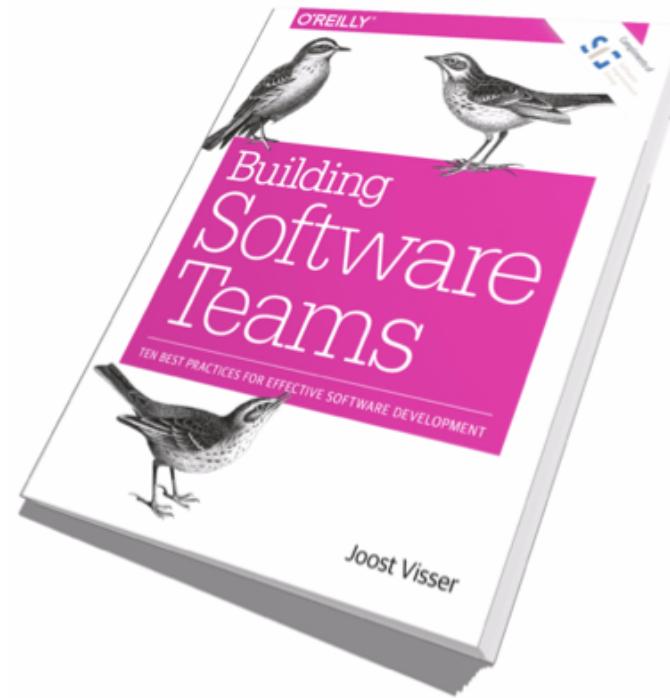
## The Vicious Cycle of Unsustainable Software Development



Recent practitioner-oriented books we published  
To break the vicious cycle



2015



2016

## The 10 Guidelines for Maintainable Software

Basic



### 1. Write Short Units of Code

Short units are easier to understand.



### 2. Write Simple Units of Code

Simple units are easier to test.



### 3. Write Code Once

Duplicated code means duplicated bugs and duplicating changes.



### 4. Keep Unit Interfaces Small

Units with small interfaces are easier to reuse.

Advanced



### 5. Separate Concerns in Modules

Modules with a single responsibility are easier to change.



### 6. Couple Architecture Components Loosely

Independent components can be maintained in isolation.



### 7. Keep Architecture Components Balanced

A balanced architecture makes it easier to find your way.



### 8. Keep Your Codebase Small

A small codebase requires less effort to maintain.

If time allows



### 9. Automate Tests

Automated tests are repeatable, and help to prevent bugs.



### 10. Write Clean Code

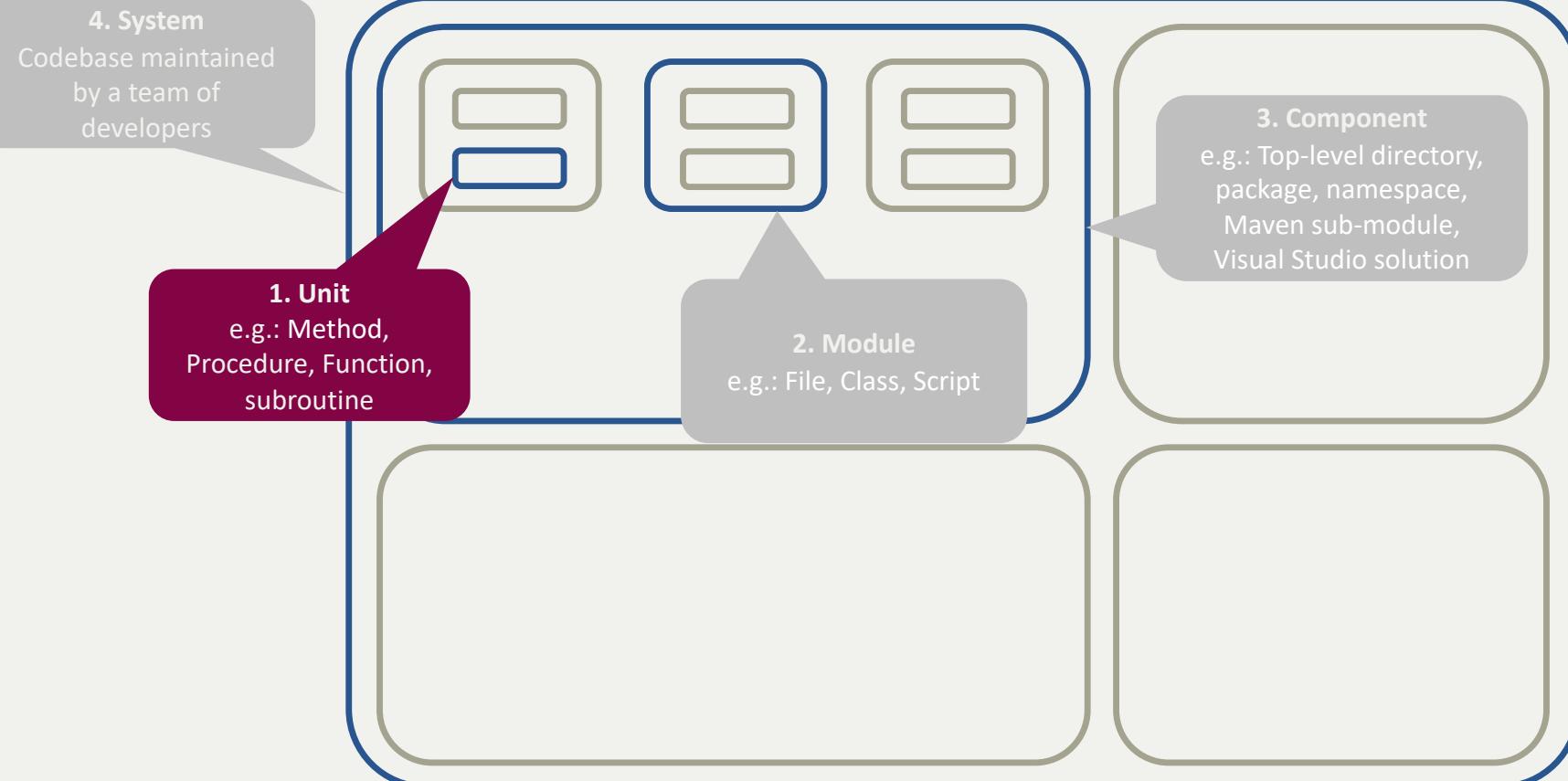
"Leave the campground cleaner than you found it."



## 1. Write Short Units of Code

Short units are easier to understand.

# What is a unit?



## Write short units of code

### The guideline

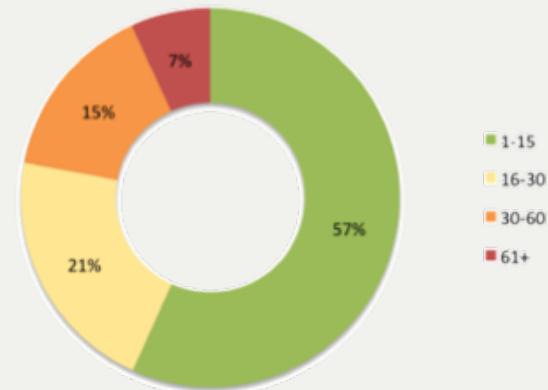
#### Do

- > Limit the length of code units to 15 lines of code

#### By

- > Not writing units that are longer than 15 lines of code
- > Splitting long units into multiple smaller units

Unit size quality profile cut-offs  
for a ★★★★☆ system



#### Because small units are

- > Easy to understand
- > Easy to test
- > Easy to reuse

#### Positive effects on other guidelines

- > Unit Complexity
- > Size of codebase

## Write short units of code

How you can measure it

### Count lines of code

- > Every line in the unit that is non-empty and does not contain only comments is a line of code.

```
1  /**
2   * Replaces all occurrences of a word in a given string with blanks.
3   */
4  public static String clearAllMatches(String input, String word) {
5      String result = "";
6      int startIndex = 0;
7      int matchIndex = input.indexOf(word);
8
9      while (matchIndex >= 0) {
10         result += input.substring(startIndex, matchIndex);
11         // Insert spaces as filler
12         for (int i = 0; i < word.length(); i++) {
13             result += " ";
14         }
15         startIndex = matchIndex + word.length();
16         matchIndex = input.indexOf(word, startIndex);
17     }
18
19     result += input.substring(startIndex);
20     return result;
21 }
```

15 lines of code

## Write short units of code

An example of non-compliant code

```
public void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    resp.setContentType("application/json");
    try {
        Connection conn = DriverManager.
            getConnection(this.conf.getProperty("handler.jdbcurl"));
        ResultSet results =
            conn.createStatement()
                .executeQuery(
                    "SELECT account, balance FROM ACCTS WHERE id="
                        + req.getParameter(conf.
                            getProperty("request.parametername")));
        float totalBalance = 0;
        resp.getWriter().print("{\"balances\":[\"");
        while (results.next()) {
            // Assuming result is 9-digit bank account number,
            // validate with 11-test:
            int sum = 0;
            for (int i = 0; i < results.getString("account")
                .length(); i++) {
                sum = sum + (9 - i)
                    * Character.getNumericValue(results.getString(
                        "account").charAt(i));
            }
        }
    }
}
```

```
if (sum % 11 == 0) {
    totalBalance += results.getFloat("balance");
    resp.getWriter().print(
        "{\"" + results.getString("account") + "\":"
            + results.getFloat("balance") + "}");
}
if (results.isLast()) {
    resp.getWriter().println("],\"");
} else {
    resp.getWriter().print(",");
}
resp.getWriter().println("\\"total\\": " + totalBalance + ")");
} catch (SQLException e) {
    System.out.println("SQL exception: " + e.getMessage());
}
}
```

This unit has 38 lines of code



## 2. Write Simple Units of Code

Simple units are easier to test.

# Write simple units of code

## The guideline

### Do

- > Limit the number of branch points to 4

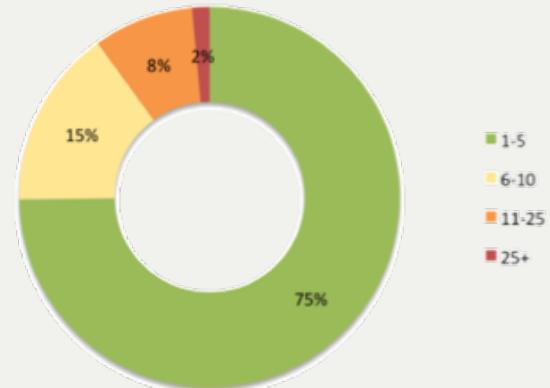
### By

- > Avoiding complex units
- > Splitting complex units into simpler ones

### Because simple units are

- > Easy to test
- > Easy to understand
- > Easy to modify

Unit complexity quality profile  
cut-offs for a ★★★★☆ system



### Positive effects on other guidelines

- > Unit Size
- > Unit Interfacing

# Write simple units of code

How you can measure it

Count the cyclomatic complexity

- > Every branch point (if, case, for, &&, ||) is counted, and we add 1 to the total

```
1  /**
2   * Replaces all occurrences of a word in a given string with blanks.
3   */
4  public static String clearAllMatches(String input, String word) {
5      String result = "";
6      int startIndex = 0;
7      int matchIndex = input.indexOf(word);
8
9      while (matchIndex >= 0) {
10          result += input.substring(startIndex, matchIndex);
11          // Insert spaces as filler
12          for (int i = 0; i < word.length(); i++) {
13              result += " ";
14          }
15          startIndex = matchIndex + word.length();
16          matchIndex = input.indexOf(word, startIndex);
17      }
18
19      result += input.substring(startIndex);
20      return result;
21 }
```

2 branch points + 1  
= cyclomatic  
complexity of 3



### **3. Write Code Once**

Duplicated code means duplicated bugs and duplicating changes.

## Write code once

### The guideline

- > Not copy code  
(no ⌘C/⌘V, no CTRL+C/CTRL+V)

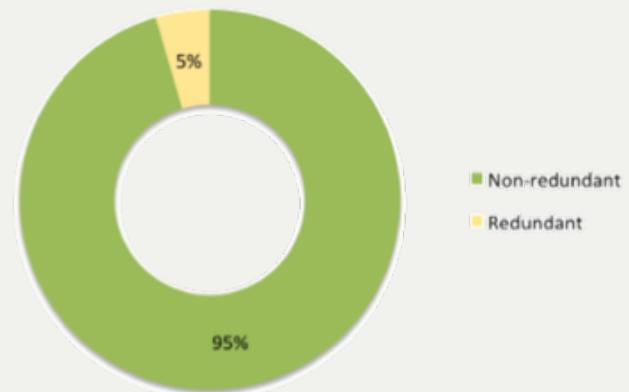
### By

- > Instead writing reusable, generic code and/or calling existing methods

### Because

- > When code is copied, bugs need to be fixed in multiple places, this is inefficient and error-prone

Code duplication quality profile  
cut-offs for a ★★★★☆ system



### Positive effects on other guidelines

- Keep your codebase small

## Write code once

How you can measure it

### Count the percentage of duplication

- > Every piece of code of 6+ lines of code that occurs more than once in the codebase is a duplicate

0: abc	34: xxxxx
1: def	35: def
2: ghi	36: ghi
3: jkl	37: jkl
4: mno	38: mno
5: pqr	39: pqr
6: stu	40: stu
7: vwx	41: vwx
8: yz	42: xxxxxx

14 duplicated lines  
7 redundant lines

## Write code once

### Impact of duplicated code on future changes

#### Low duplication

- One defect means one fix



#### High duplication

- One defect means several fixes ...
- ... and even more might be missed





#### **4. Keep Unit Interfaces Small**

Units with small interfaces are easier to reuse.

# Keep unit interfaces small

## The guideline

### Do

- > Limit the number of parameters per unit to 4

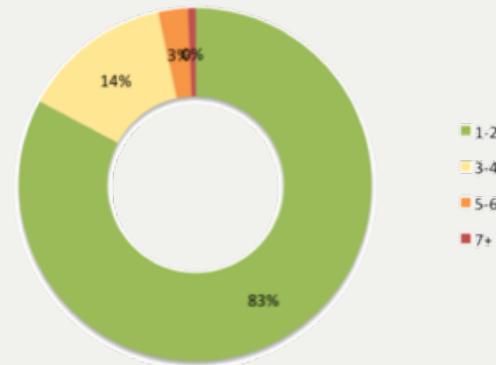
### By

- > Extracting parameters into objects

### Because units with few parameters are

- > Easy to understand
- > Easy to test
- > Easy to reuse

Unit interfacing quality profile  
cut-offs for a ★★★★☆ system



### Positive effects on other guidelines

- Simple Units
- Short Units

# Keep unit interfaces small

How you can measure it

## Count the number of parameters

- > Every parameter in the unit header

```
1  /**
2   * Replaces all occurrences of a word in a given string with blanks.
3   */
4  public static String clearAllMatches(String input, String word) {
5      String result = "";
6      int startIndex = 0;
7      int matchIndex = input.indexOf(word);
8
9      while (matchIndex >= 0) {
10          result += input.substring(startIndex, matchIndex);
11          // Insert spaces as filler
12          for (int i = 0; i < word.length(); i++) {
13              result += " ";
14          }
15          startIndex = matchIndex + word.length();
16          matchIndex = input.indexOf(word, startIndex);
17      }
18
19      result += input.substring(startIndex);
20      return result;
21 }
```

2 parameters

# Keep unit interfaces small

## The guideline

### Do

- > Limit the number of parameters per unit to 4

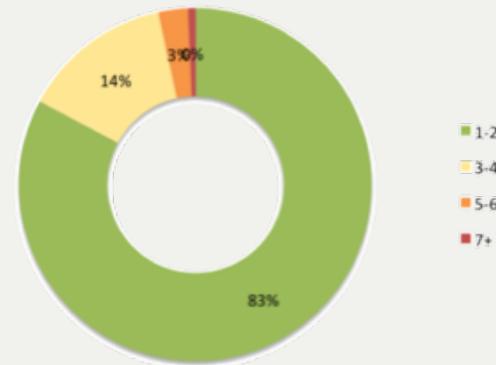
### By

- > Extracting parameters into objects

### Because units with few parameters are

- > Easy to understand
- > Easy to test
- > Easy to reuse

Unit interfacing quality profile  
cut-offs for a ★★★★☆ system



### Positive effects on other guidelines

- Simple Units
- Short Units

# Keep unit interfaces small

How you can measure it

## Count the number of parameters

- > Every parameter in the unit header

```
1  /**
2   * Replaces all occurrences of a word in a given string with blanks.
3   */
4  public static String clearAllMatches(String input, String word) {
5      String result = "";
6      int startIndex = 0;
7      int matchIndex = input.indexOf(word);
8
9      while (matchIndex >= 0) {
10          result += input.substring(startIndex, matchIndex);
11          // Insert spaces as filler
12          for (int i = 0; i < word.length(); i++) {
13              result += " ";
14          }
15          startIndex = matchIndex + word.length();
16          matchIndex = input.indexOf(word, startIndex);
17      }
18
19      result += input.substring(startIndex);
20      return result;
21 }
```

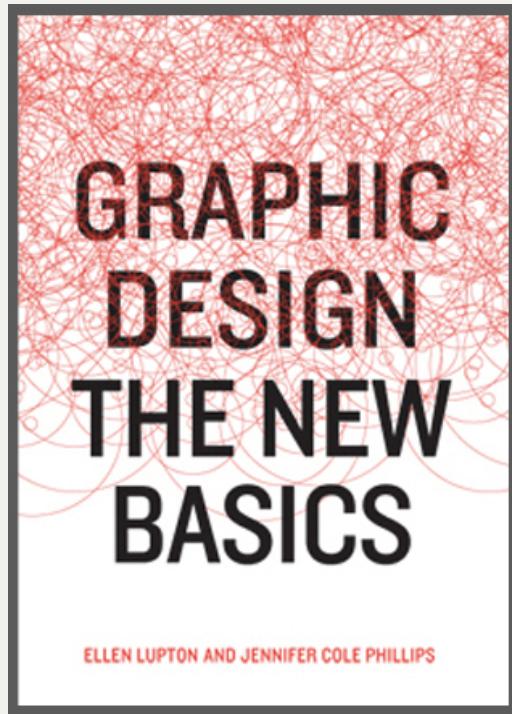
2 parameters

**Software design & architecture**

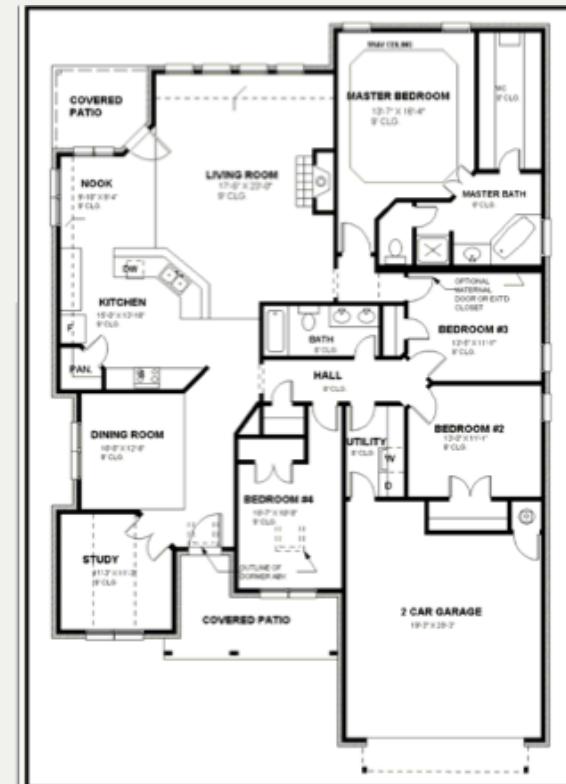
# What do we mean by software design?

No, not that kind of design

(though it's also important)

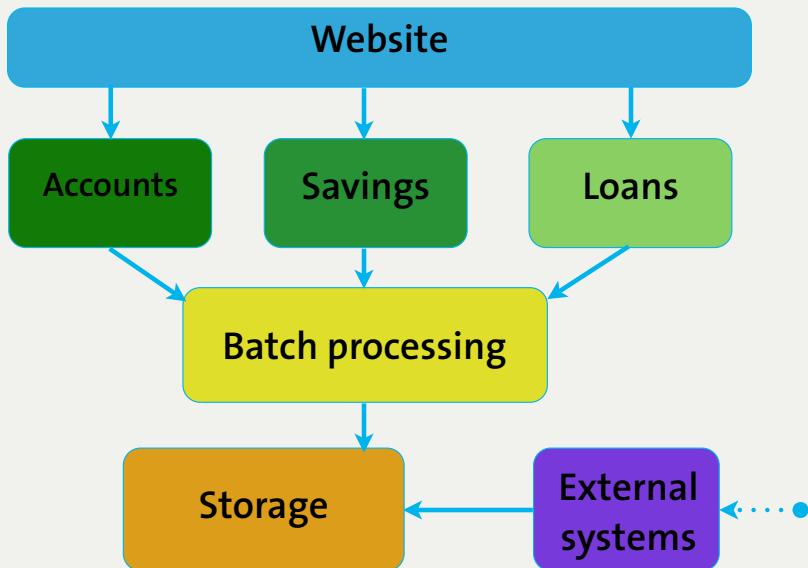


We mean “design” as in a high-level plan

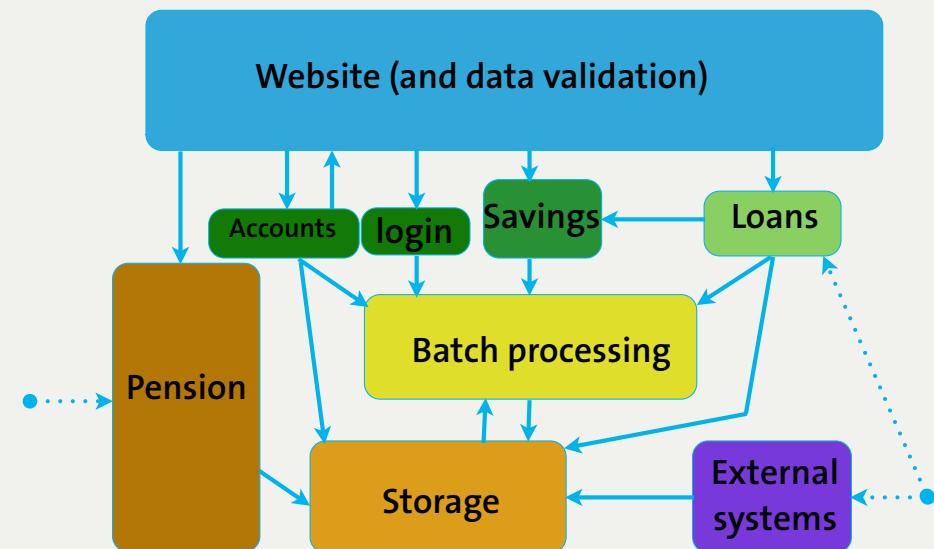


## Intended design vs. implemented design

Intended design

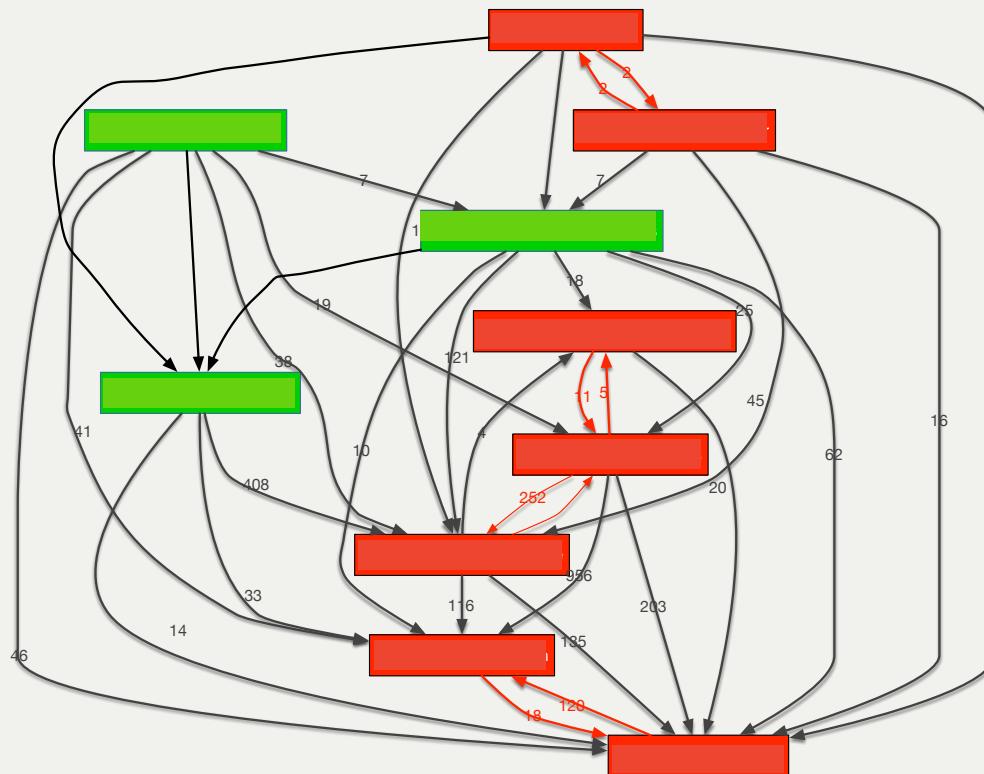


Implemented design



How did this happen?

## This is what happens when architecture is not governed



- Over time, unintended dependencies are introduced
- Responsibilities between components are fading over time
- Circular dependencies are introduced



## 9. Automate Tests

Automated tests are repeatable, and help to prevent bugs.

## Automate Tests

### The guideline

Do:

- > Automate tests for your codebase

By:

- > Writing automated tests using a test framework
- > Executing these tests as often as feasible

Because automated tests:

- > Are repeatable
- > Are more efficient
- > Make your code predictable



## 10. Write Clean Code

"Leave the campground cleaner than you found it."

# Write Clean Code

## The guideline

Do:

- > Write clean code

By:

- > Avoiding code smells
- > Not leaving smelly code behind

Because clean code:

- > Is more maintainable
- > Is less confusing than unhygienic code

## Write Clean Code

Seven ‘Boy Scout rules’

1. Leave no **unit level code smells** behind
2. Leave no **bad comments** behind
3. Leave no **code in comments** behind
4. Leave no **dead code** behind
5. Leave no **long identifiers** behind
6. Leave no **magic constants** behind
7. Leave no **badly handled exceptions** behind

## Better Code Hub

Improve what matters • online • integration with GitHub

**Software Improvement Group  
supports Odyssey.org in getting code  
quality right!**



Backlog &  
Documentation



Team  
Communication



Version  
Control



Product  
Quality



Continuous  
Integration



Cloud  
Hosting



Analytics

# Odyssey GitHub Org



1. Teams
2. Private repo per team
  - ‘Weakest link’
3. Invites have been sent to team leads
4. Send me github handles
5. **Your IP is protected**

The screenshot shows the GitHub organization page for 'ODYSSEY HACKATHON 2019'. The top navigation bar includes links for 'Repositories 99', 'People 90', 'Teams 98', 'Projects 0', and 'Settings'. Below the navigation is a search bar with placeholder text 'Find a repository...', and filters for 'Type: All' and 'Language: All'. A green button labeled 'New' is visible. The main content area displays four repository cards:

- purplepension** [Private] - Updated an hour ago
- odysseyhack.github.io** - Updated 10 hours ago
- circularise** [Private] - Updated 14 hours ago
- planet-society** [Private] - Updated 18 hours ago

On the right side, there is a 'Top languages' section showing 'HTML' with a red dot, and a 'People' section showing 90 users with their profile pictures. A button 'Invite someone' is located at the bottom of the people section.

# Better Code Hub a tool by Software Improvement Group

<https://bettercodehub.com>

1. 17 technologies
2. Private and public repos
3. Use Github account
4. Accept scopes
5. View results
6. Select refactor candidates
7. Plan sprint
8. Run again
  1. Repeat step 3
9. Know when done
10. Show score with a badge

The screenshot shows the homepage of Better Code Hub. At the top, there's a dark blue header with the logo 'Better Code Hub BY SOFTWARE IMPROVEMENT GROUP' and navigation links for 'Repositories', 'About', 'Pricing', 'Docs', and a user profile icon. Below the header, a large central section features the text 'Write Better Code. With a Definition of Done.' in bold blue font. Underneath, a subtext reads: 'Better Code Hub helps development teams spend less time fixing bugs and more time shipping new features.' Two buttons are present: 'Try it for free' (blue) and 'Watch how it works' (purple). To the right of the text is a cartoon illustration of a person sitting at a desk, working on a laptop, with a coffee cup nearby. At the bottom of the page is a dark blue footer bar containing icons for various programming languages: TS, C++, Python, C#, PHP, JS, Swift, and React Native, along with left and right arrows.

# Odysseyhack now

The screenshot shows the Better Code Hub interface with the following details:

**Header:** Better Code Hub, Your repositories, Search, Analyzed only, Hide forks, github.com/MichielCuijpers, Michiel Cuijpers

**Repository Analysis Results:**

- odysseyhack/model-family**: Last analyzed: 18 hours ago, Score: 9
- odysseyhack/quack**: Last analyzed: 2 days ago, Score: 8
- odysseyhack/odysseyhack.github.io**: Last analyzed: 9 days ago, Score: 5
- odysseyhack/a-solid-web**: Repository not analyzed
- odysseyhack/apg-rescue-rangers**: Repository not analyzed
- odysseyhack/Arceus**: Repository not analyzed
- odysseyhack/atos-blockchain-factory**: Repository not analyzed
- odysseyhack/bcec-team-b**: Repository not analyzed
- odysseyhack/bencom**: Repository not analyzed

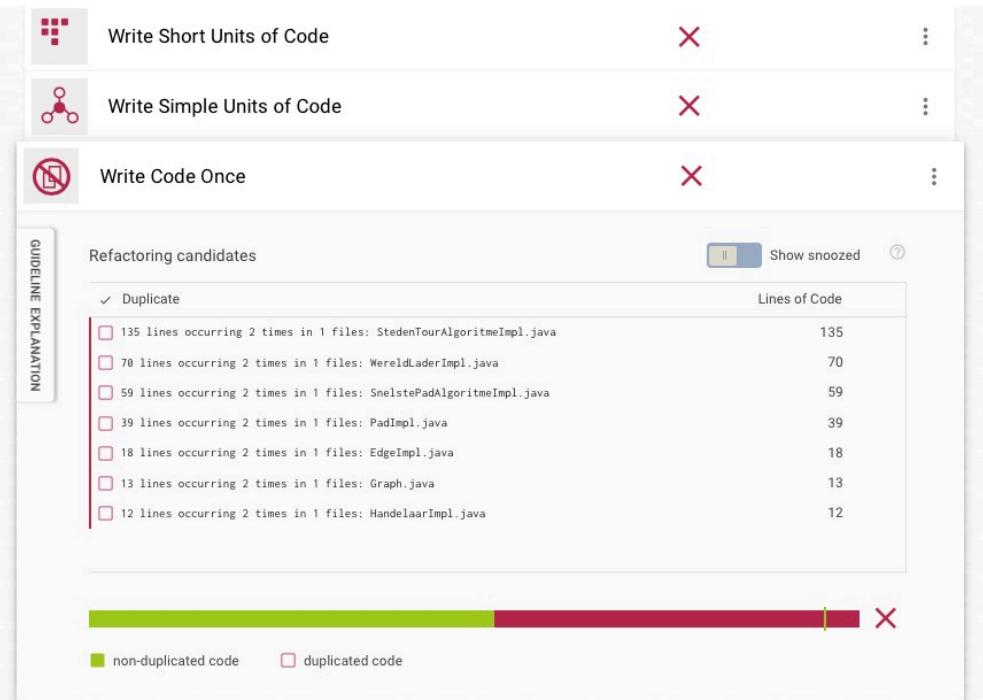
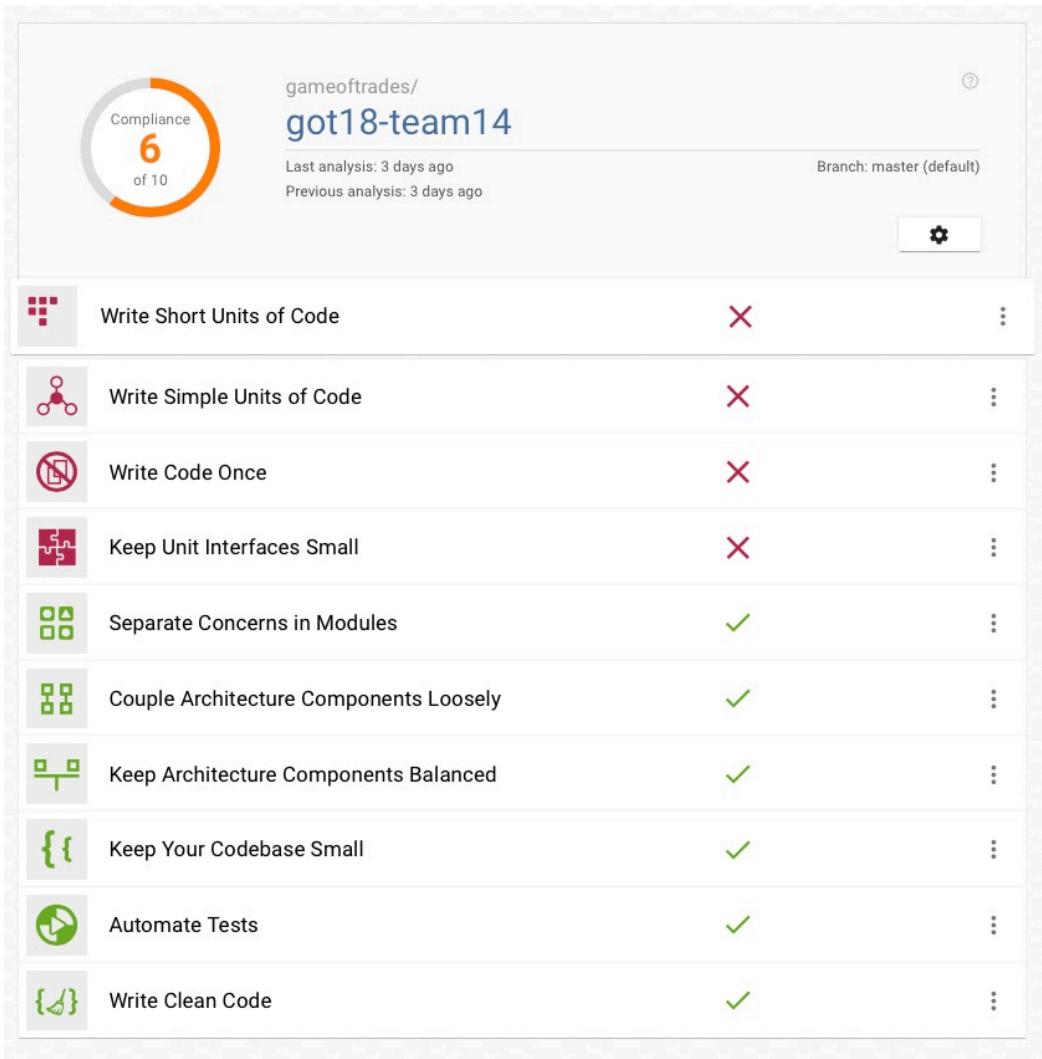
**Footer:** Software Improvement Group / PUBLIC

# Example overview

## Gameoftrades for Haagse Hogeschool

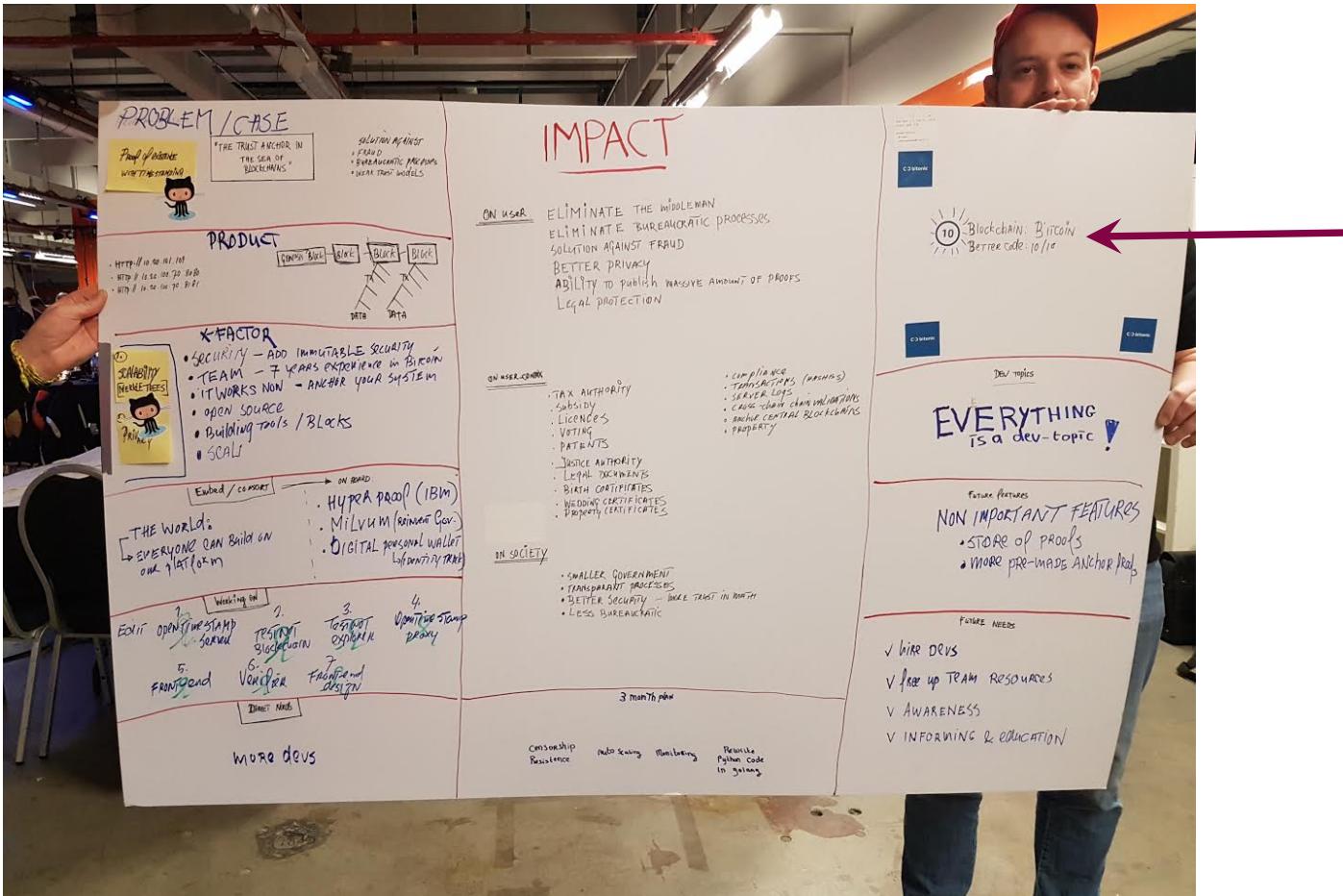
The screenshot shows a dashboard from Better Code Hub displaying nine repository cards arranged in a 3x3 grid. Each card represents a different team's repository, all under the 'gameoftrades' organization. The cards include icons for code analysis, pull requests, and issues, along with a large circular progress indicator showing the current analysis status.

Repository	Last analyzed	Progress (Circle)
gameoftrades/got18-team14	3 days ago	6
gameoftrades/got18-team35	3 days ago	5
gameoftrades/got18-team44	3 days ago	8
gameoftrades/got18-team38	3 days ago	8
gameoftrades/got18-team18	3 days ago	7
gameoftrades/got18-team26	3 days ago	8
gameoftrades/got18-team43	4 days ago	7
gameoftrades/got18-team12	4 days ago	8
gameoftrades/got18-team36	4 days ago	9



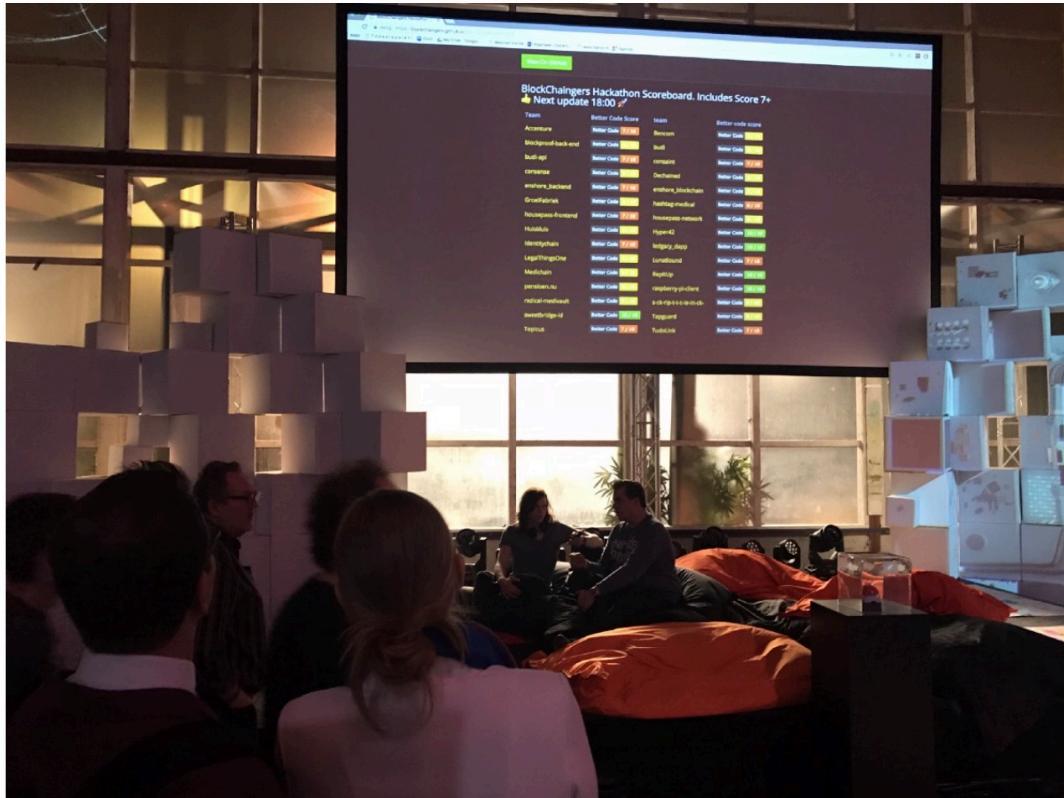
# Better Code Hub at Blockchain Hackathons

It started at the Big Building, Groningen



# Better Code Hub at Dutch Blockchain Hackathon

## Scoreboard



Team	Better Code Hub Score
AC-BC	Better Code 8 / 10
Accenture	Better Code 6 / 10
atma	Better Code 10 / 10
BadgeX	Better Code 8 / 10
Bencom	Better Code 10 / 10
blockproof-back-end	Better Code 9 / 10
blockproof-business-network	Better Code 8 / 10
blockproof-front-end	Better Code 10 / 10
budi	Better Code 9 / 10
budi-api	Better Code 10 / 10
cgi	Better Code 7 / 10
consaint	Better Code 9 / 10
consense	Better Code 10 / 10

The current code quality scores for all teams were displayed as GitHub badges on a central scoreboard for all to see.

## Mature prototype

### How to get a 'Mature Prototype' Certificate?

- If your main repo scores a Better Code Hub 10/10 you qualify to schedule a grilling session.
- Schedule the 10 minute slot for your session at the Jedi Hangout (limited slots available, so be quick!)
- Get Grilled in this interview (slots available Saturday 20.00-22.00 and Sunday 9.00-12.00)

## Deliverable 1 of 4 - Working prototype in your Github account

Teams commit code for

- Better Code Hub
- (Continuous) feedback and ranking
- Looking under the hood  
(does it work?)
- Unlock “mature prototype status”

Audit by Software Improvement Group

Your IP is protected.

Read the General Participation Conditions

<https://www.odyssey.org/general-conditions-for-participants-of-the-odyssey-hackathon-2019/>



# Appreciative Score Model

Used by the jury: maximum of 20 points per team

Quality of Prototype  
0-5 points awarded for this quadrant

Impact  
0-5 points awarded for this quadrant

21st Century Team & Solution  
0-5 points awarded for this quadrant

Incubation plan  
0-5 points awarded for this quadrant

## Appreciative Model in detail - Quality of Prototype

Solution viability	How does the prototype demonstrate it can solve the described problem?
Technical difficulty	To what level is the technical approach innovative?
Blockchain and/or AI implementation in prototype	How are the benefits of blockchain and/or AI technology leveraged for this solution?
Compatibility	To what level are internet and industry standards used where relevant?
Maturity of Prototype	Has the team earned a star indicating they have a mature prototype?
Clean Code?	What is the Better Code Hub Score?



## How to earn this star for Mature Prototype?

Qualify by scoring 10 out of 10 for your code quality?

Schedule your 10 minute Grilling Session at the Jedi Hangout

Earn your 'Mature Code' Star to proof to the jury you have a mature prototype

Don't snooze - only limited spots available

Saturday 20.00-22.00

Sunday 9.00-12.00



## What do you check during the Grilling Session?

<b>Odyssey EASSI model</b>	How mature is your prototype?
Existential	Why Blockchain? Why is Blockchain a good fit for your application, as opposed to a more traditional approach?
Architecture	How good is your system architecture? SIG Software engineering system characteristics.
Scalability	Will it scale, what are the bottlenecks? This is what every investor wants to know!
Security	Encryption (private) key storage. How do you encourage/enable your users to safely score their private key?
Inertia	Number of Nodes running the BC. How to make it interesting to join the chain.



## Code Coaches JEDI



**Jan Laan** • 1st

Software Security Consultant at Software Improvement Group (SIG)



**Reinier Vis** • 1st

Head of Software Consultants at Software Improvement Group (SIG)



**Bugra M. Yildiz** • 1st

Software Consultant at Software Improvement Group (SIG),  
Software Engineer



**Hugo Schoonewille** • 1st

Software Consultant at Software Improvement Group (SIG)

# Professional certification for software maintainability

The screenshot shows the PEOPLECERT website's homepage for Test Takers. The main headline reads "Better Code is <cheaper/>!". Below it, a sub-headline says "Get it right with Software Maintainability Qualifications". A sidebar on the left contains a section about "Quality Software Development Qualifications – New for 2016" and another about "Software Maintainability". A large orange call-to-action button at the bottom right encourages users to "Fill in your details to request training or accreditation information".

PEOPLECERT  
Certifying Professionals

Language: English  
Search: Search

Candidates' Login | Partners' Area | PASSPORT Login | Contact

Test Takers   Test Owners   Training Providers   Corporations   About Us

Home ▶ Test Takers ▶ Quality Software Development

## Better Code is <cheaper/>!

Get it right with  
**Software  
Maintainability  
Qualifications**

Quality Software Development Qualifications – New for 2016

**Software Maintainability**

Demand for good software is growing very fast, however building quality software remains a major challenge. To tackle this, PEOPLECERT in collaboration with the Software Improvement Group (SIG), has developed a qualification in Quality Software Development to certify programmers who have the necessary skillset to produce high quality code.

Fill in your details to request training or accreditation information

A few details about you:

\* Mandatory field

Please select as appropriate\*

Select

Name \*

# Roles in academic education

## Software Engineering



The screenshot shows the homepage of the 'MAKE IT WORK' program at Hogeschool van Amsterdam. The header features the Hogeschool van Amsterdam logo and the text 'Hogeschool van Amsterdam' above 'MAKE IT WORK'. The main banner has a blue background with white text asking 'BEN JE HOOGOPGELEID (HBO/WO) EN OP ZOEK NAAR (ANDER) WERK?'. Below the banner, there's a photo of a person working on a laptop. Navigation links include HOME, OVER MAKE IT WORK, ACTUEEL, WERKGEBER, and ICT'ER IN. Below the banner, there's a section titled 'MAKE IT WORK NIEUWS' with news items like 'Start zevende groep Make IT Work' (7 februari 2017) and 'Robotproject zesde cohort Make IT Work' (24 januari 2017). To the right, there's a box for 'SOFTWARE ENGINEER IN DE MEDIA' with text about a collaboration with Hilversum Media Campus and a 'MEER INFORMATIE' button.

### Guest lecturing & courses:

- Open Code Clinic @ TU Delft
- Make IT Work program @ HvA
- Master Software Engineering @ UvA
- Courses at Radboud University Nijmegen
- Guest lectures at Hanze Hogeschool
- ...



## Contact

 +31 20 314 09 50

 m.cuijpers@sig.eu

 @sig\_eu

