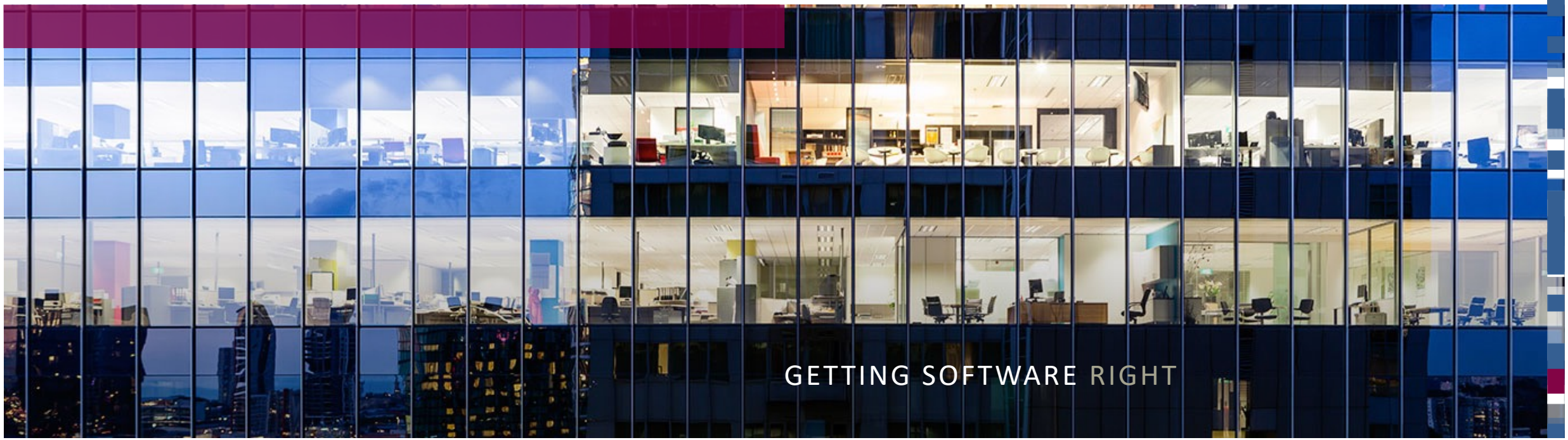
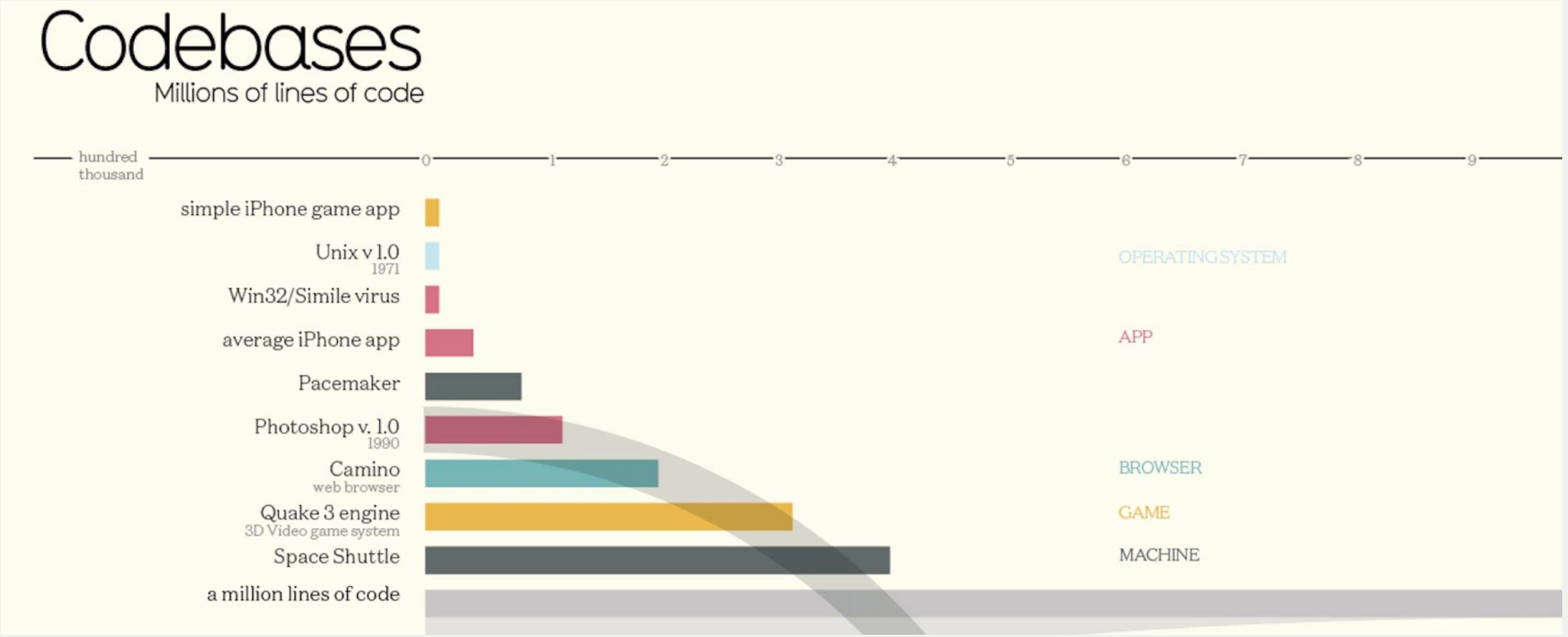
Software Improvement Group

Software metrics in practice, 2022

Magiel Bruntink, Head of Research

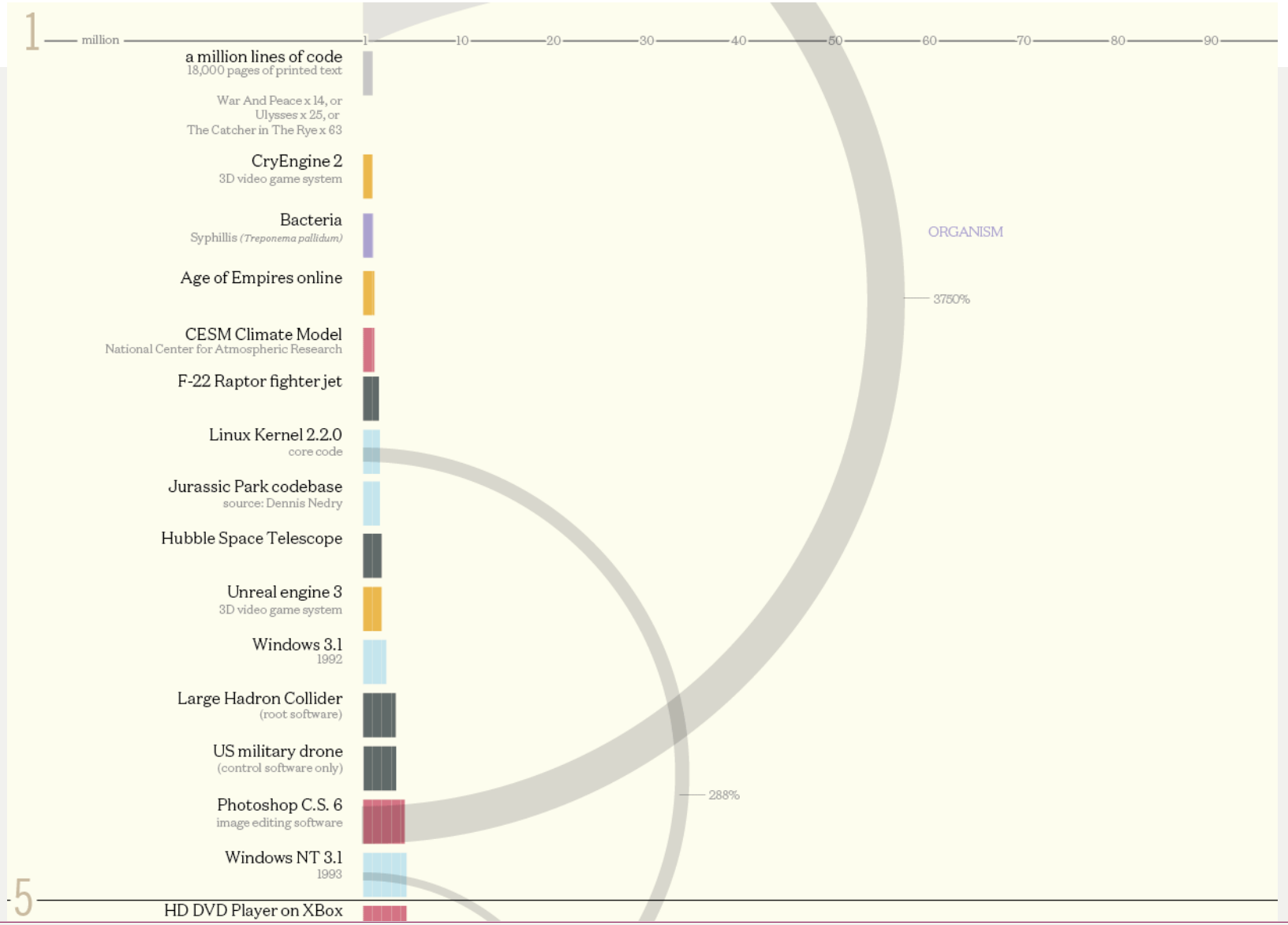


Real software is large…

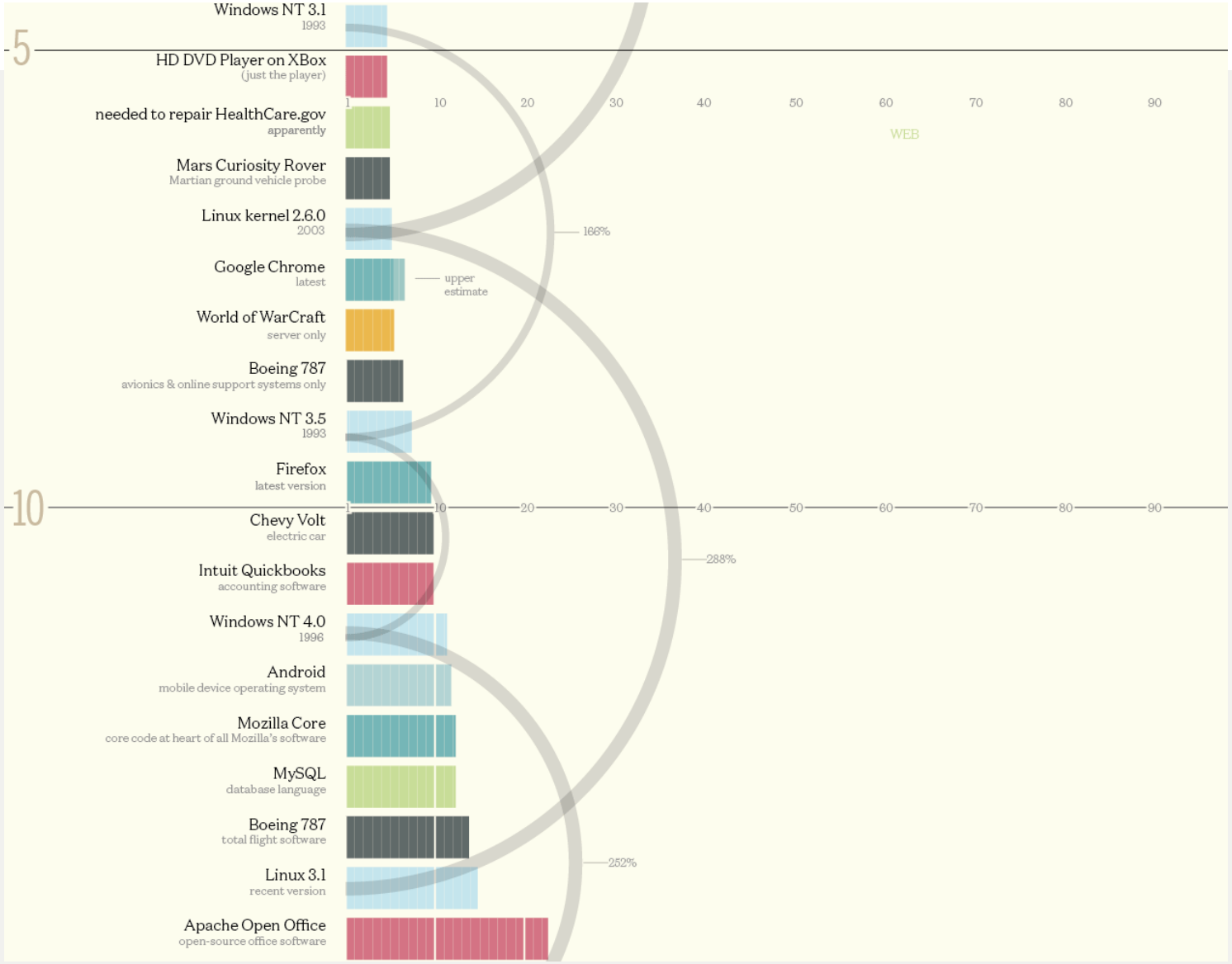


https://informationisbeautiful.net/visualizations/million-lines-of-code

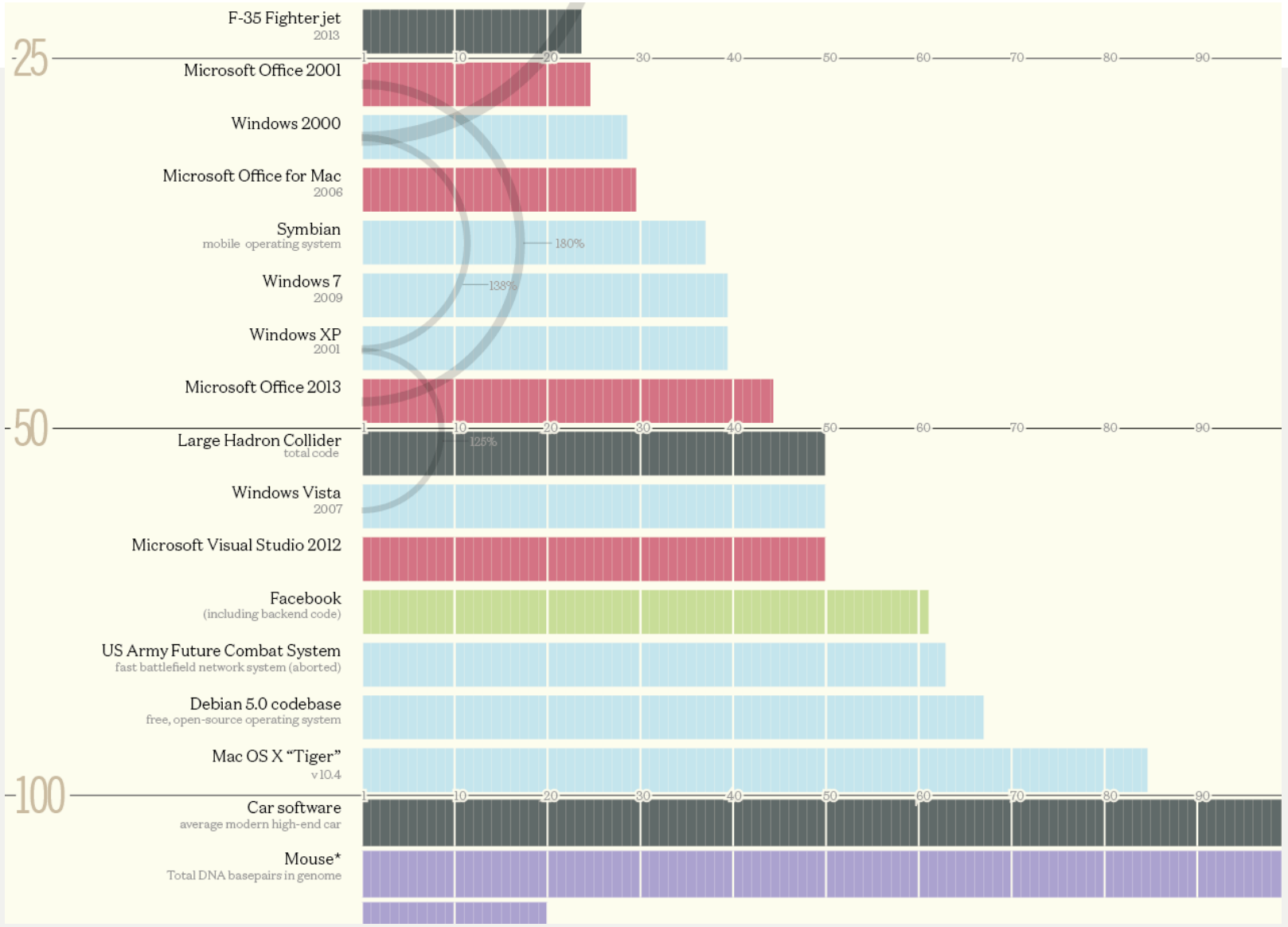
Real software is very large…



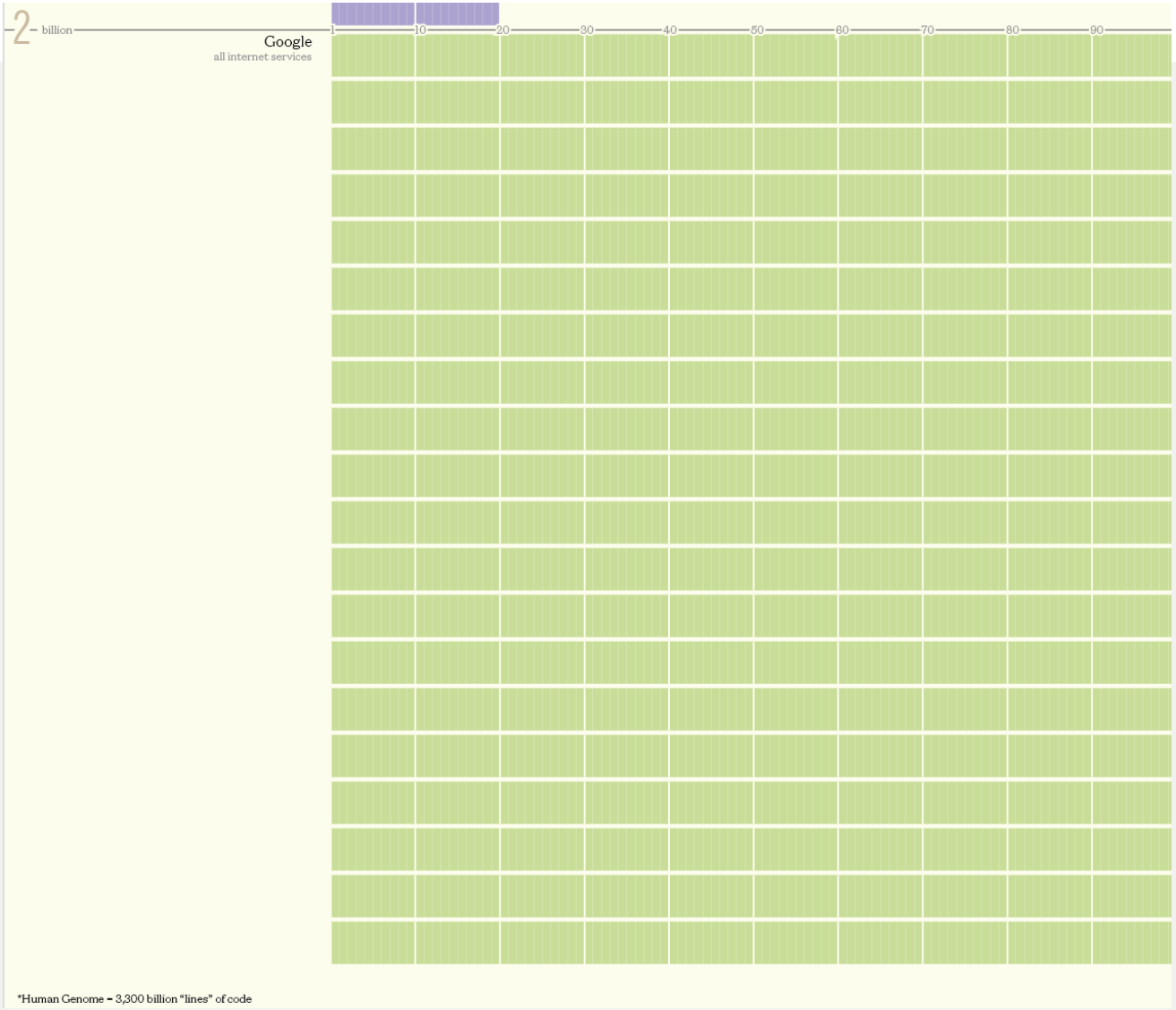
Real software is huge…



Real software is absurdly huge…



Software is taking over the world…



Some interesting statistics

> Average programmer productivity:
  - Roughly 45 lines of working Java code per working day
  - Let’s say 10.000 LOC/year

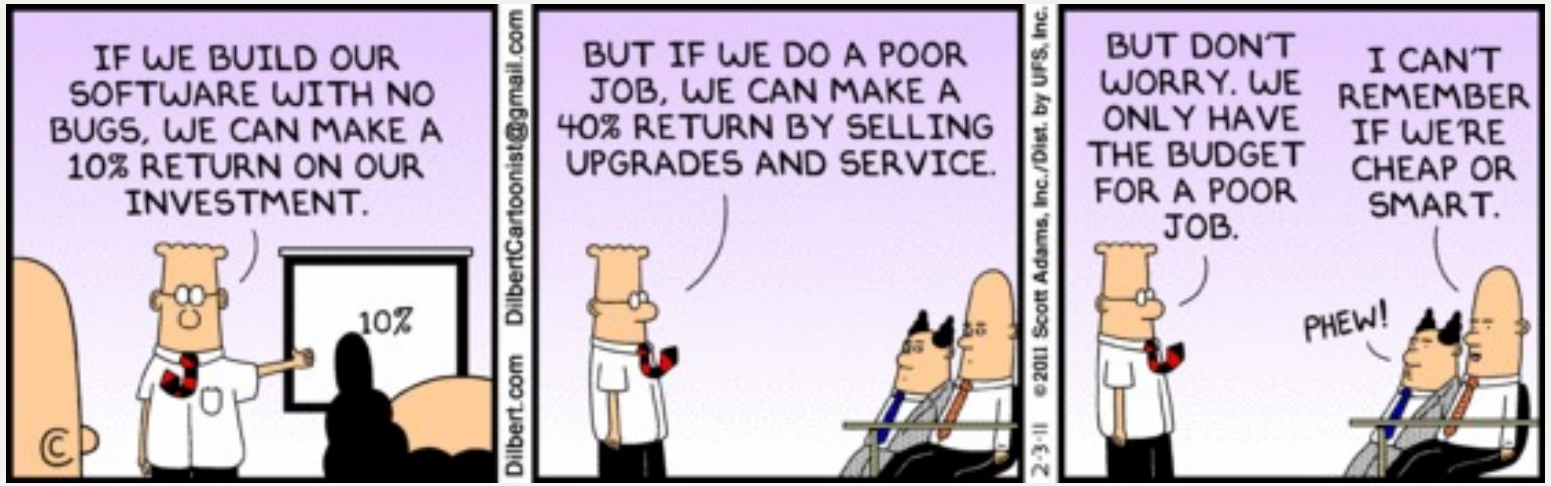
> Operational software ***yearly*** change rate:
  - Roughly 5 to 15% of the code base

> Average ***yearly*** maintenance effort a high-end consumer car:
  - 100 MLOC x 5 to 15% = 5 to 15 MLOC
  - 15 MLOC / 10.000 LOC / year = 500 to 1.500 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)



A famous example of a code smell: Apple's SSL/TLS bug

```
618      hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
619      hashOut.length = SSL_SHA1_DIGEST_LEN;
620      if ((err = SSLFreeBuffer(&hashCtx)) != 0)
621          goto fail;
622
623      if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
624          goto fail;
625      if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
626          goto fail;
627      if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
628          goto fail;
629      if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
630          goto fail;
631          goto fail;
632      if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
633          goto fail;
634
635      err = sslRawVerify(ctx,
636                         ctx->peerPubKey,
637                         dataToSign,              /* plaintext */
638                         dataToSignLen,           /* plaintext length */
639                         signature,
640                         signatureLen);
641      if(err) {
642          sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
643                      "returned %d\n", (int)err);
644          goto fail;
645      }
646
647  fail:
```

Background details

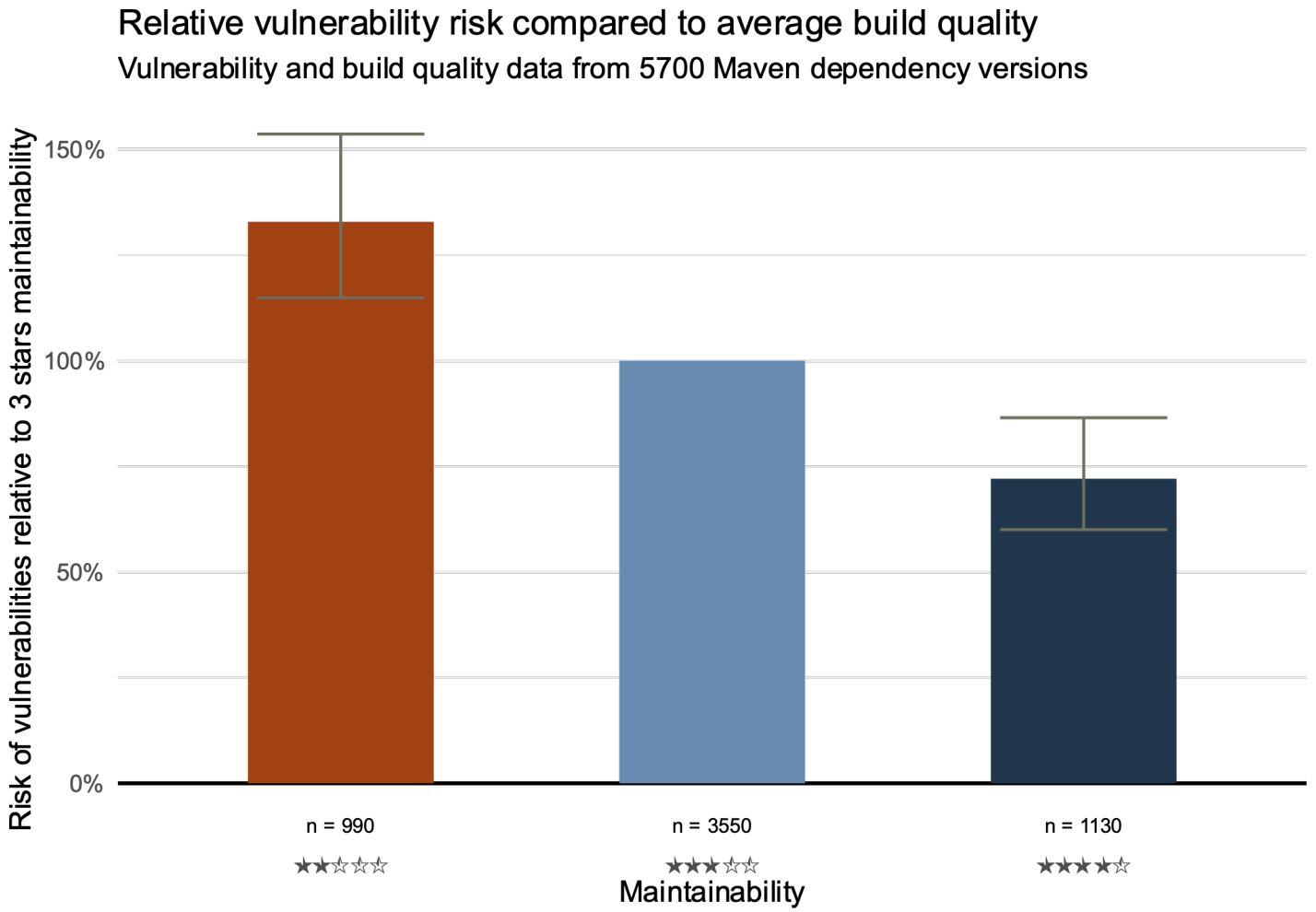
- CVE ID: CVE-2014-1266
- Apple iOS 6.x (< 6.1.6)
- Apple iOS 7.x (< 7.0.6)
- Apple TV 6.x (< 6.0.2)
- Apple OS X 10.9.x (< 10.9.2)
- Impact: millions of devices?

Unit:

SSLVerifySignedServerKeyExchange(…)
LOC: 64
McCabe: 14
Nr. of parameters: 5

*Read more: https://www.imperialviolet.org/2014/02/22/applebug.html

More risk of vulnerabilities with lower quality code



Key findings:

- The risk of security vulnerabilities is related to the **build quality of open-source libraries.**
- A 2-star library has close to **2X more risk** than a 4-star library.
- Consider library build quality to inform the choice of open-source libraries to reduce risk of future exposure to vulnerabilties.

If you should remember just one thing...

The quality of (your) code matters!

SIG offers services based on software measurement

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

20 years of software quality measurement at SIG

300
technologies
from mainframe to mobile

9,000
industry systems evaluated

345 x 10^6
lines of code analyzed each week

75 x 10^9
lines of code in analysis warehouse

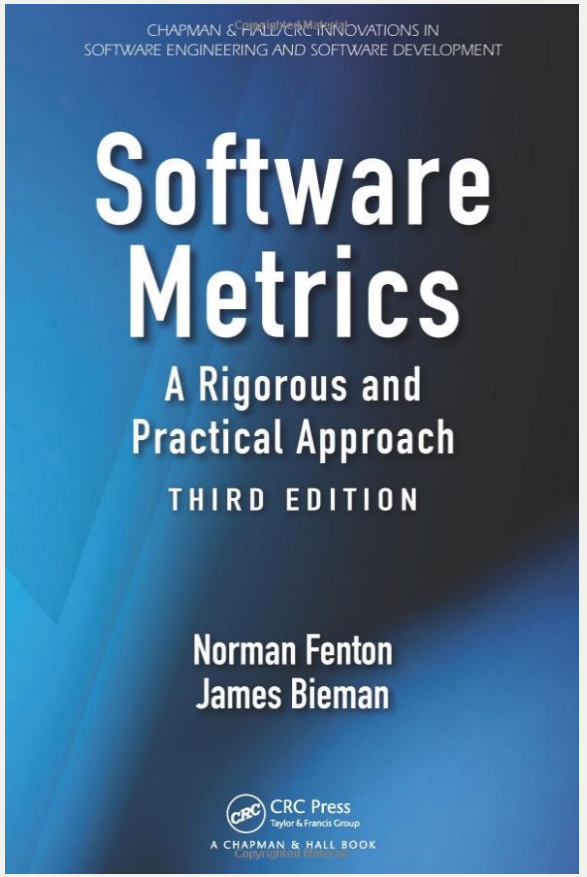
Software quality can be measured using tools.

Measurement concepts

"Formally, we define measurement as a mapping from the empirical world to the formal, relational world."

Source: Fenton et al., Software Metrics: a rigorous and practical approach. ISBN 0534954251

Entity	The object that is being measured.
Attribute	An aspect/property of the entity that is being measured.
Measure	How to express the value of the attribute.
Mapping	Ensures repeated measurements are reliable and comparable.



Meaningless Metrics, Treacherous Targets

> Often heard: “by Drucker and/or Demming”:
“You can’t manage what you don’t measure”

> Turns out they either didn’t say it, or indeed said quite the opposite!

> Read this blog by Florian Haas
https://xahteiwi.eu/blog/2021/11/14/meaningless-metrics-treacherous-targets/

> What to do?

1. Make measurement cost no effort
2. Generate measurements all the time
3. Know you are not measuring many important things
4. Share measurement data points
5. Apply real statistics and teach people
6. Apply science before changing policy

My own:

- Know your metrics are never done or perfect
- Incorporate domain expert feedback
- Always consider the actionability

The SIG Maintainability model[1]

[1] Ilja Heitlager, Tobias Kuipers, and Joost Visser. 2007. A Practical Model for Measuring Maintainability. In *Proceedings of the 6th International Conference on Quality of Information and Communications Technology* (QUATIC '07). IEEE Computer Society, Washington, DC, USA, 30-39. DOI: https://doi.org/10.1109/QUATIC.2007.7

Measuring software maintainability

Goals

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 maintainble.
Technology-independent	The measurements should be usuable regardless of what technologies were used to implement the system.

Maintainability is an important attribute to control

ISO/IEC 25010:2011

Maintainability is a special aspect of software quality

It can enable (or hinder) improvements on other quality characteristics

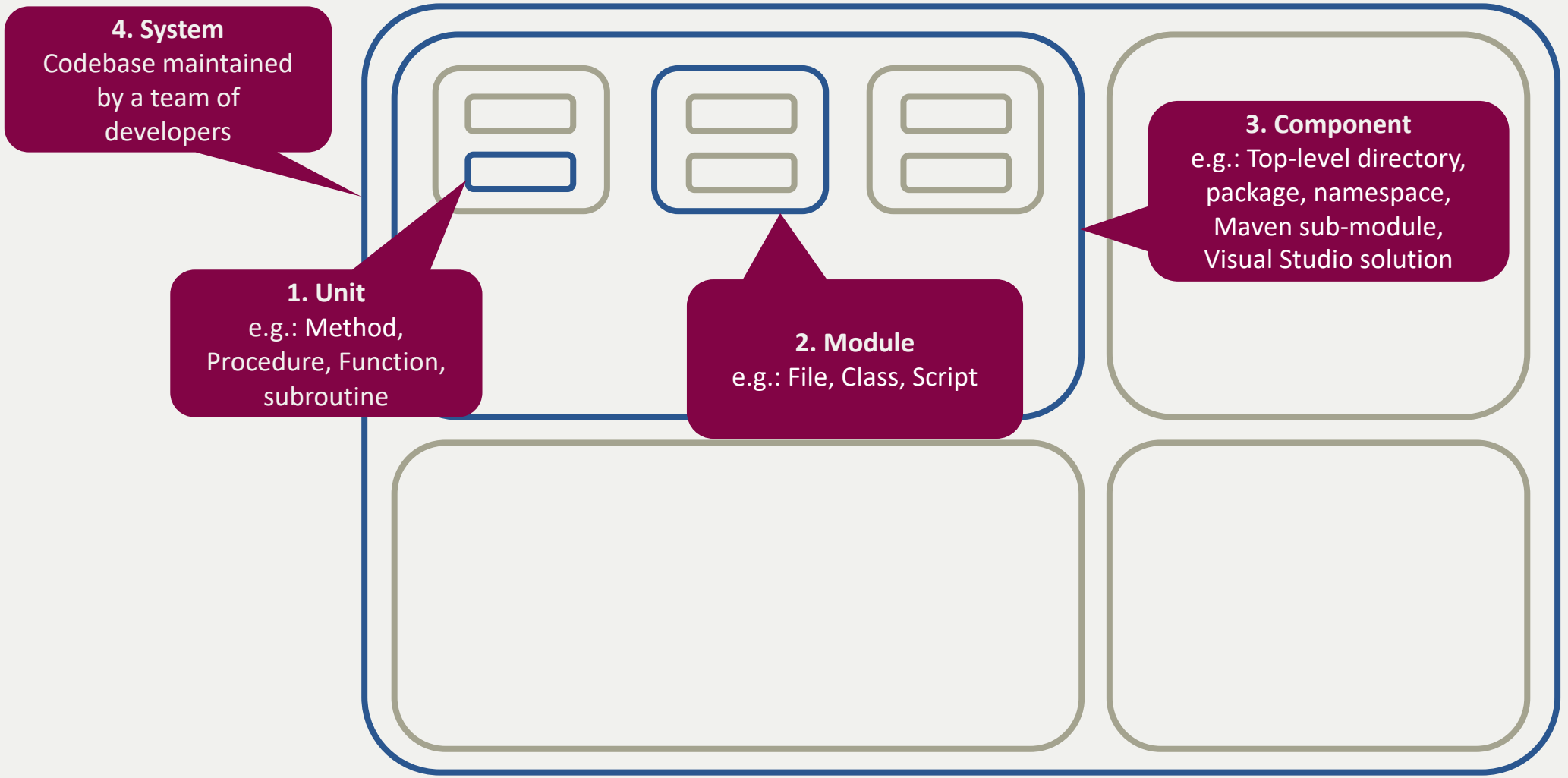
Maintainability according to ISO 25010

Sub-characteristics explained

"Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers."

Sub-characteristic	Definition
Analyzability	Degree to which it is possible to assess the impact of an intended change.
Modifiability	Degree to which the product can be modified without introducing defects or degrading product quality.
Testability	Can test criteria be established for a product and can tests be performed to determine whether those criteria have been met?
Modularity	Is the product composed of components such that a change to one component has minimal impact on other components?
Reusability	Degree to which an asset can be used in more than one system.

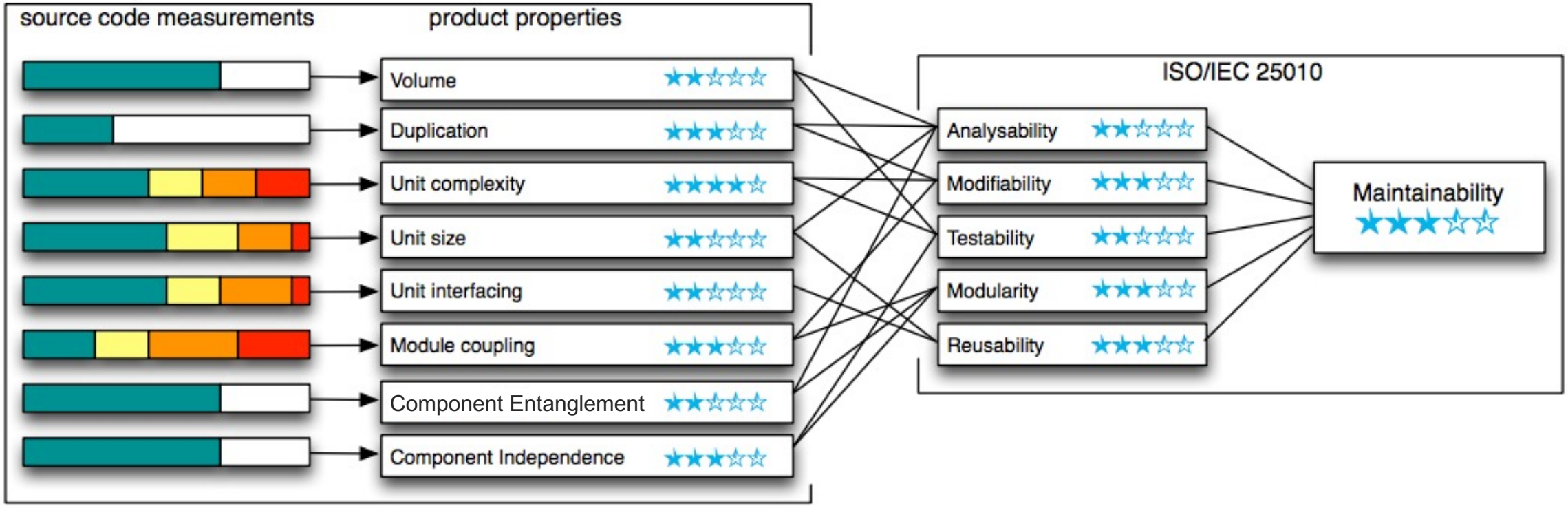
Levels of measurement



ISO 25010 describes Maintainability in terms of 5 sub-characteristics

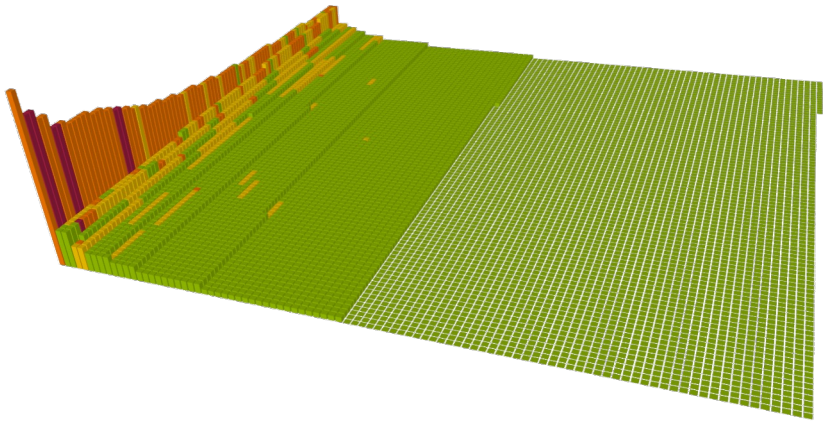
System property ratings are determined by systems in the SIG benchmark

A star rating for software system maintainability

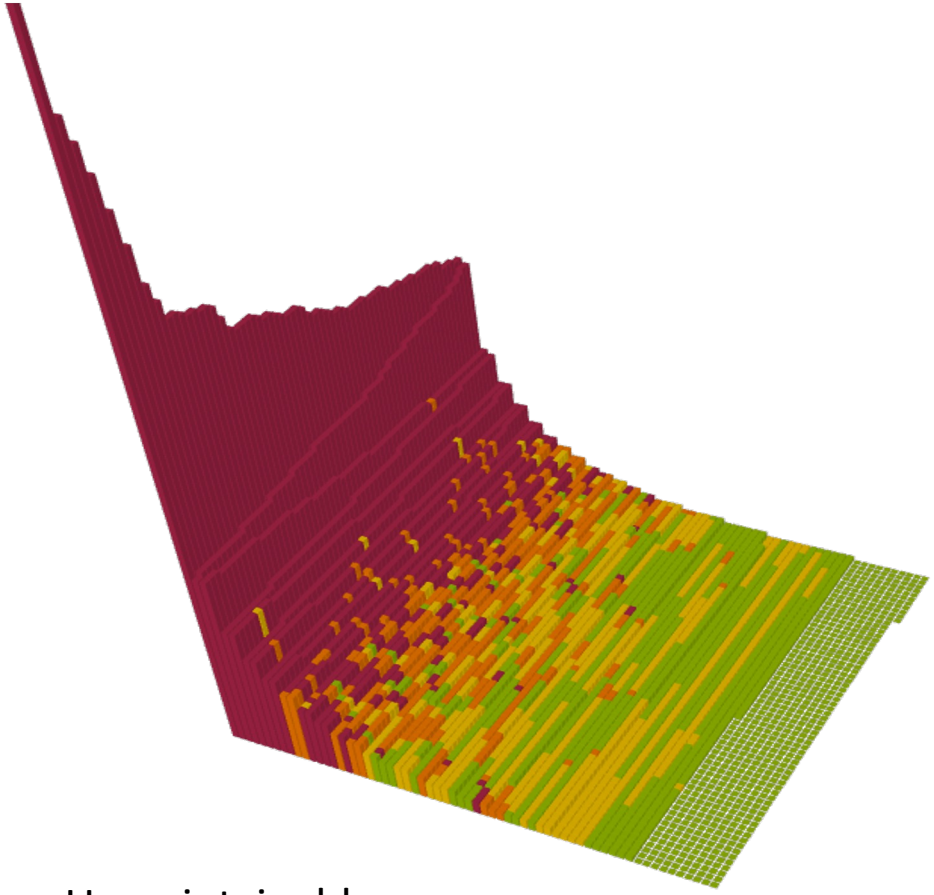


No codebase is perfect!

But some code bases are more ‘perfect’ than others



Maintainable



Unmaintainable

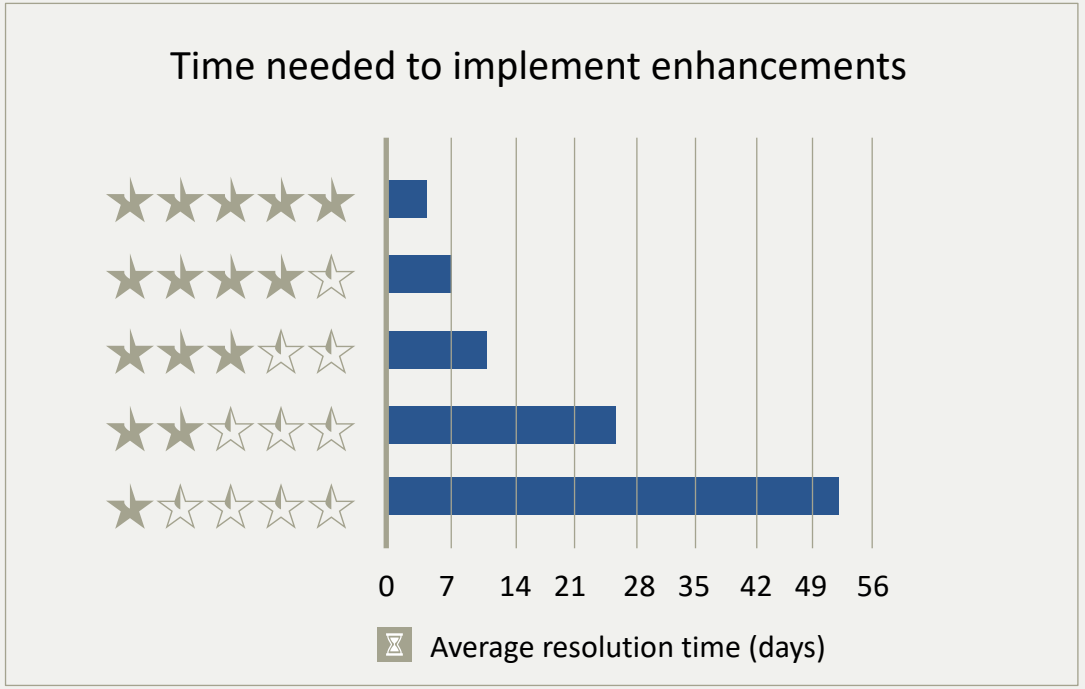
Quantifying the advantages of high quality

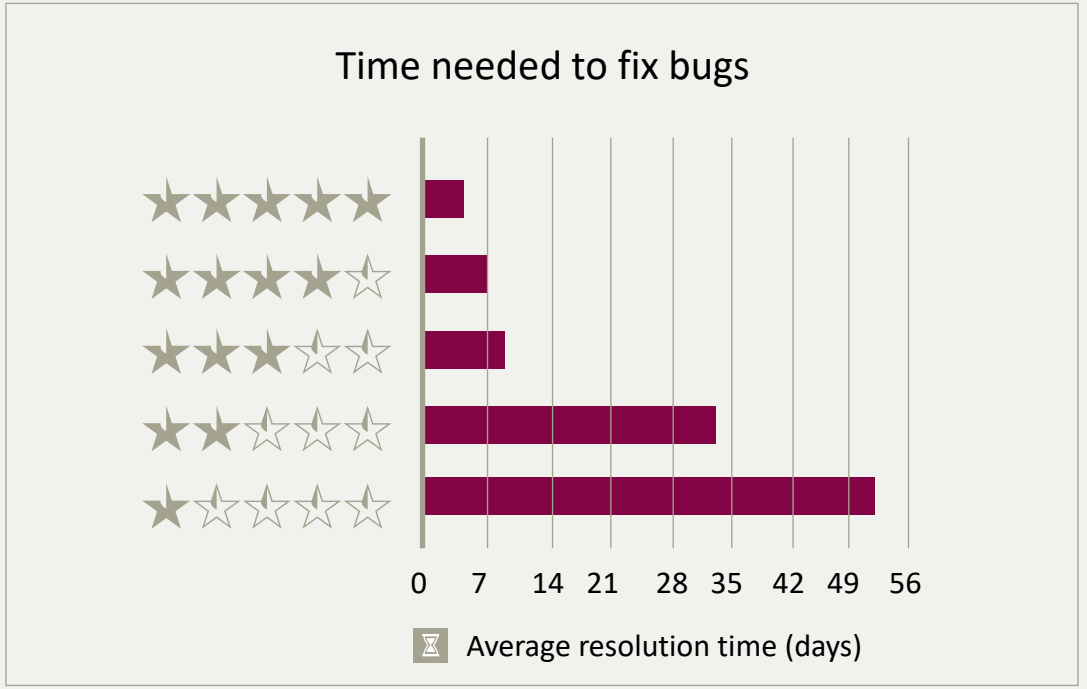
New functionality can be implemented faster in more maintainable systems

A higher software product quality leads to:

> The faster implementation of enhancements and the solution of defects

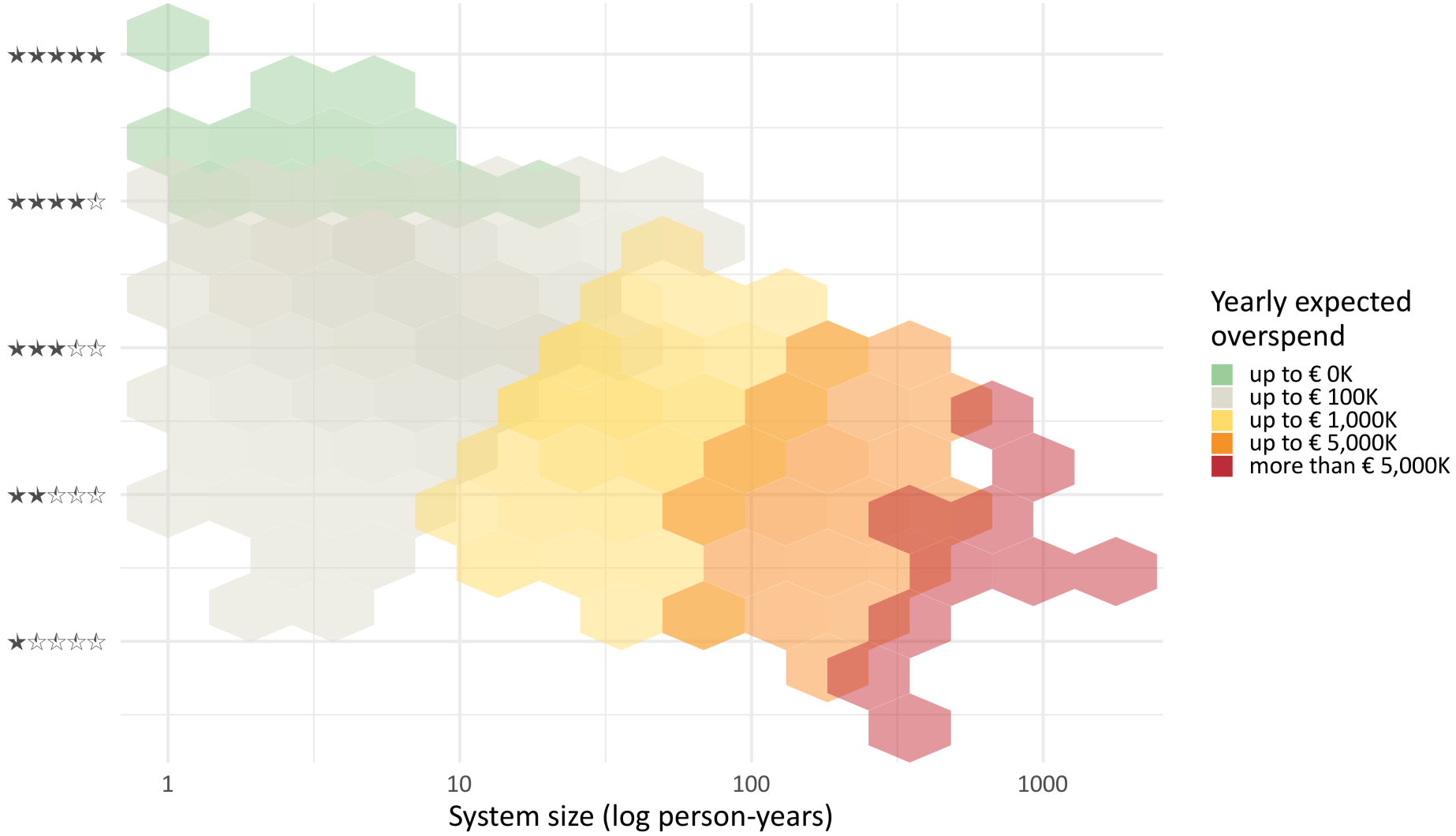
> The throughput rate improves by factor 3.5 to 4.0 between 2 and 4 stars



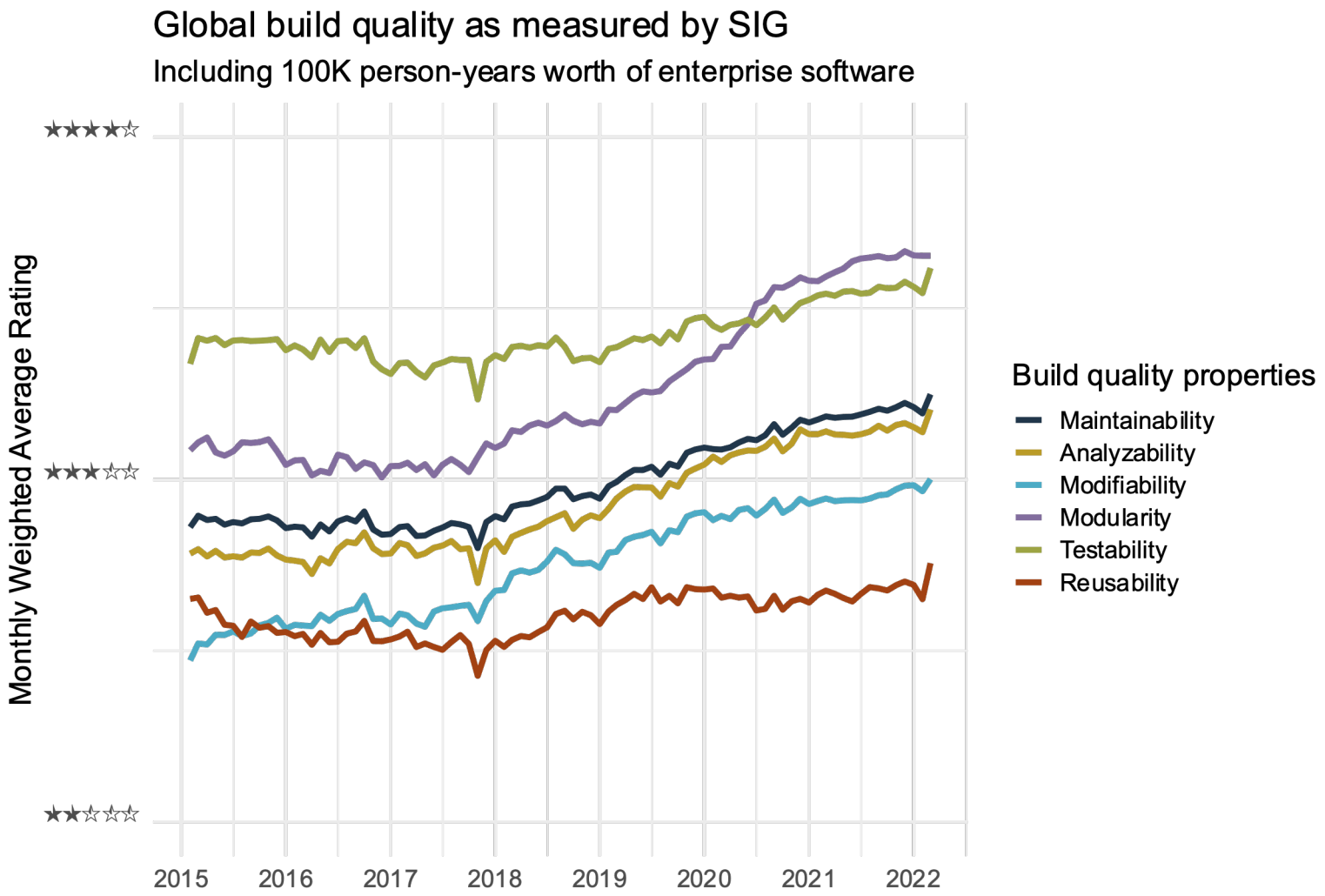


Source: "Faster issue resolution with higher technical quality of software", Software Quality Journal, 2011

Impact of code quality



Global maintainability trend: up or down?

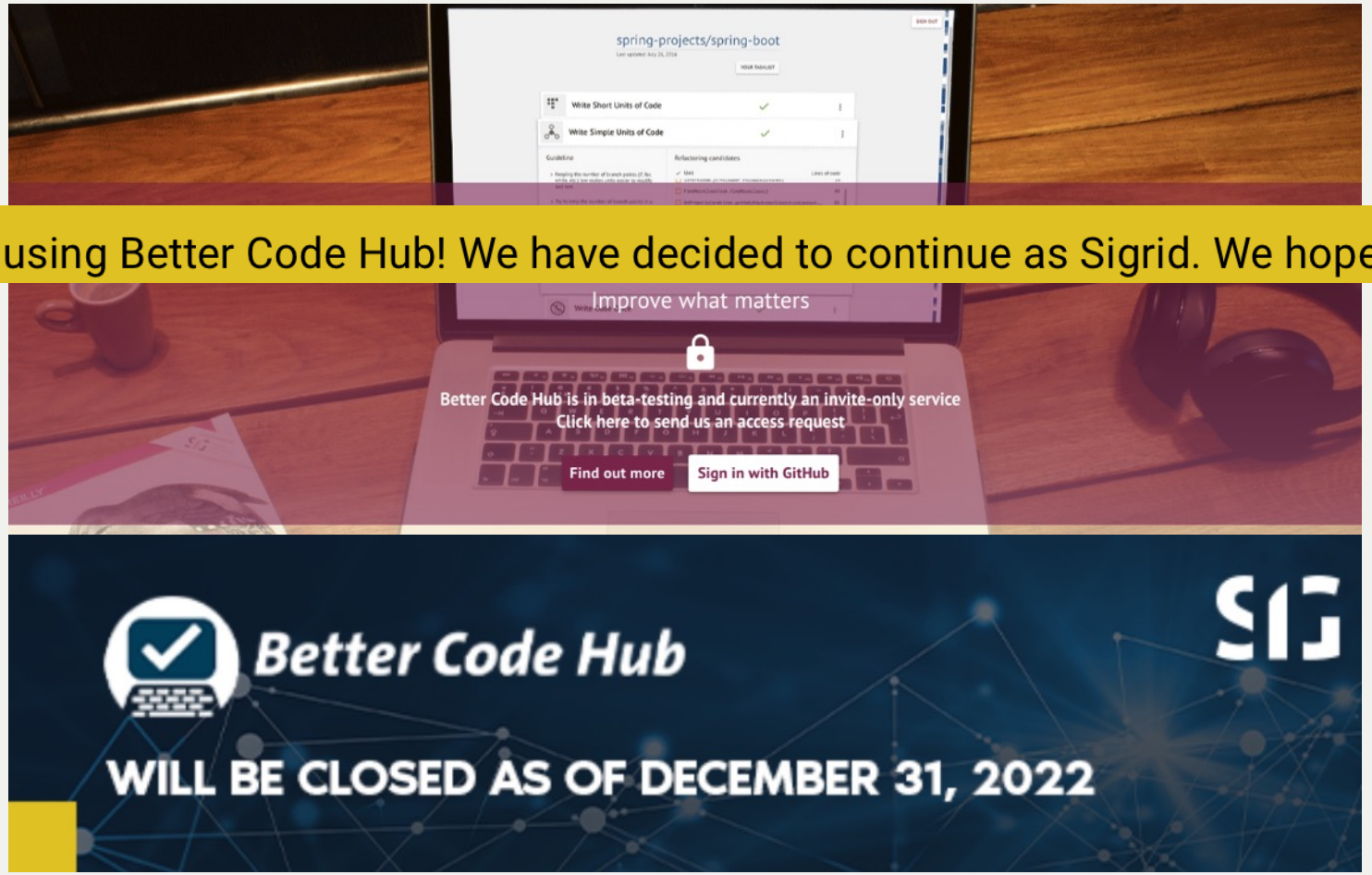


Key findings:

- Global build quality keeps getter better: keep improving to keep up!
- The largest build quality improvement is seen for Modularity, an architecture property.
- Software applications developed recently keep getting smaller and better structured than in the past.

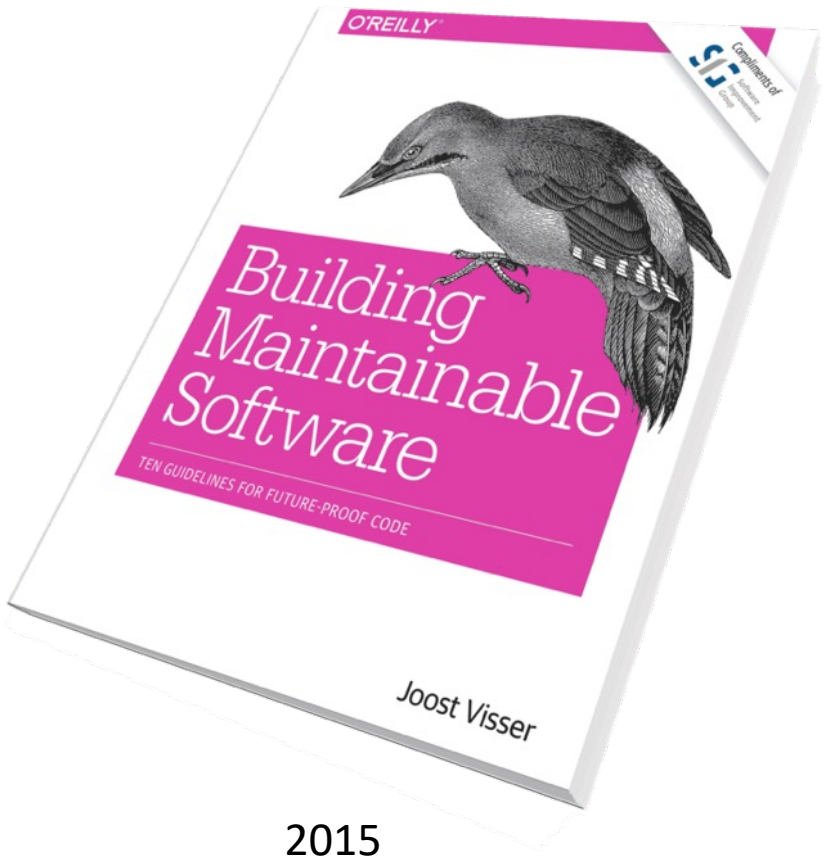
Software quality measurement should be actionable in day-to-day development.

Sigrid does code quality metrics right for everyone

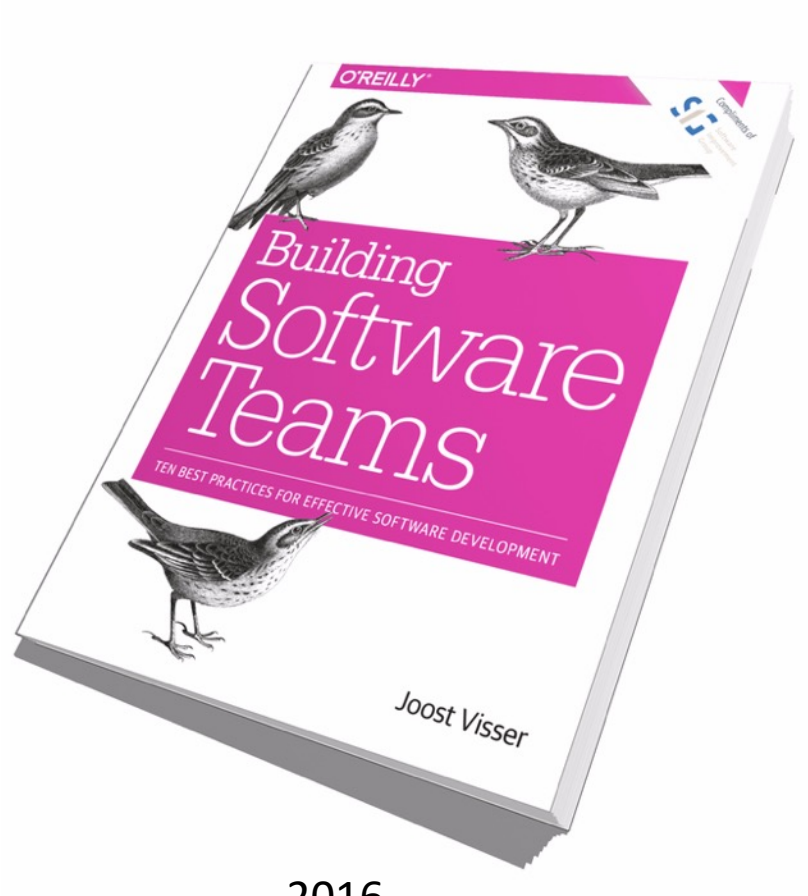


Thank you for using Better Code Hub! We have decided to continue as Sigrid. We hope to meet again.

Practitioner-oriented books we published



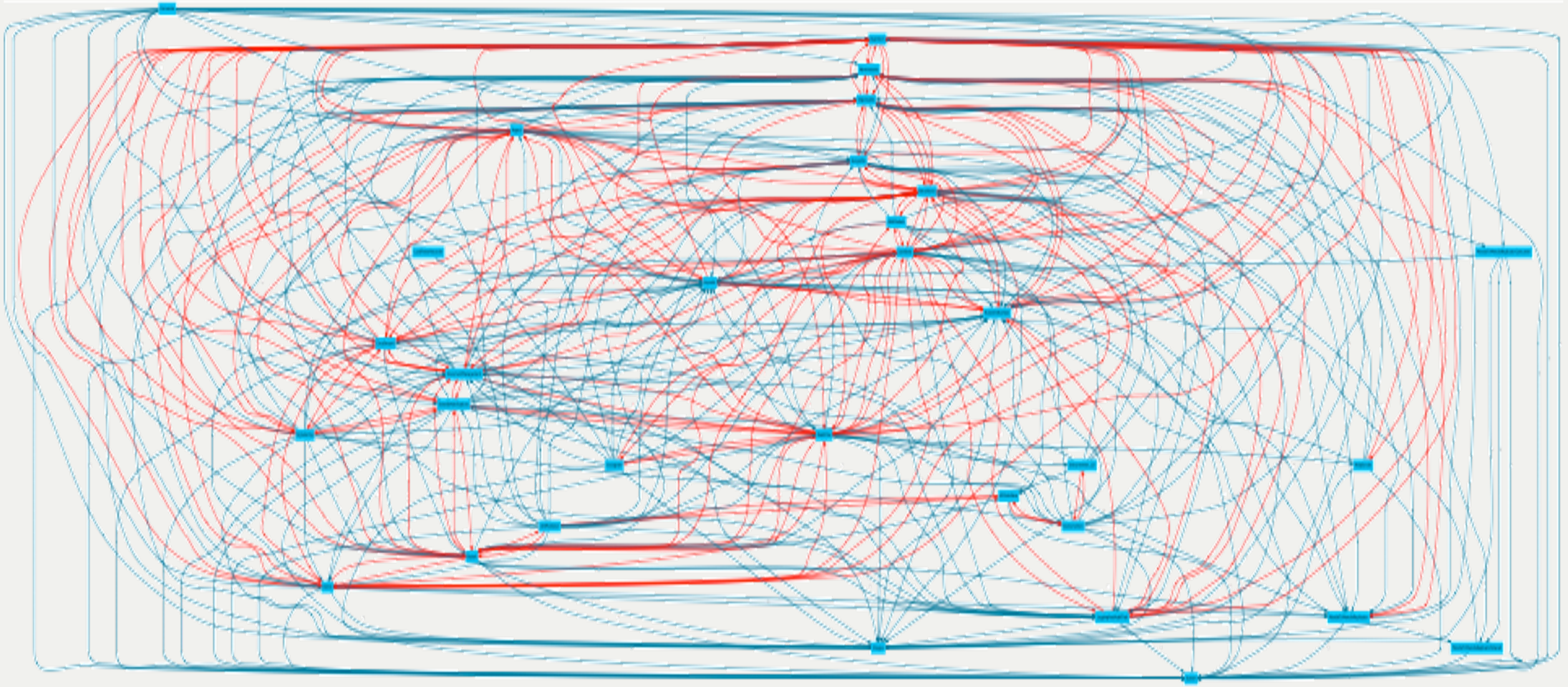
2015



2016

How did this happen?

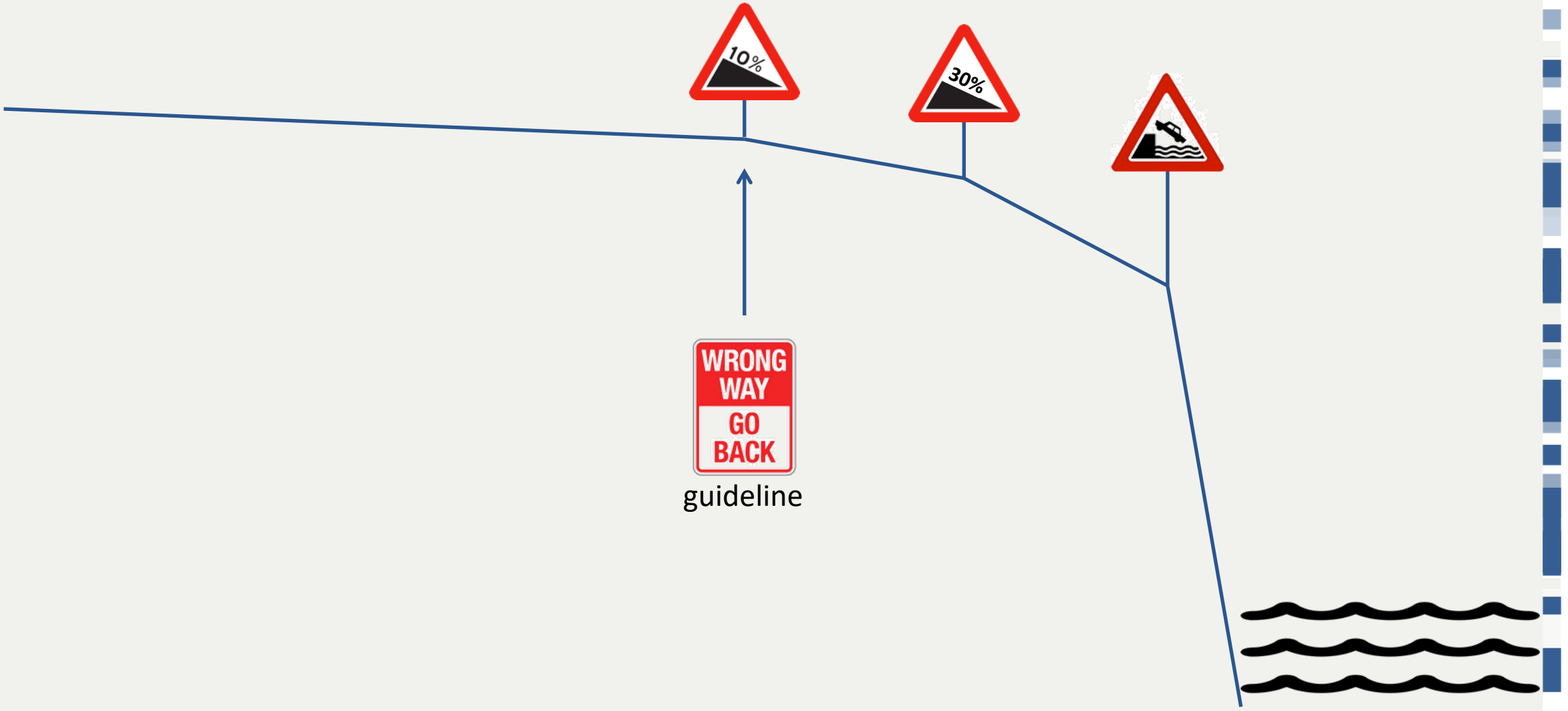
One line of code at a time



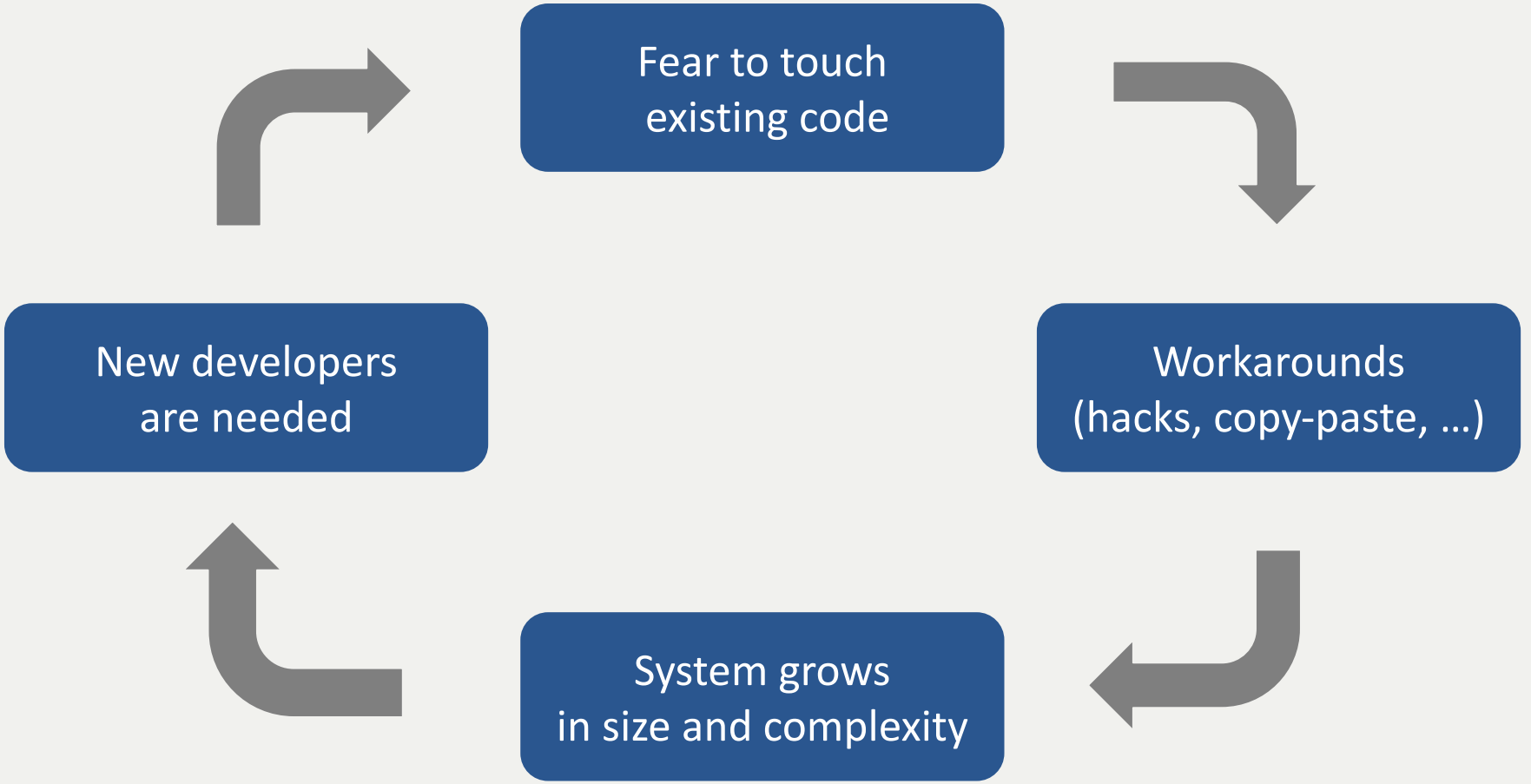
And how did *this* happen?

Also one line of code at a time

Identifying Tipping Points



The Vicious Cycle of Unsustainable Software Development



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.

```java
/**
 * Replaces all occurrences of a word in a given string with blanks.
 */
public static String clearAllMatches(String input, String word) {
   String result = "";
   int startIndex = 0;
   int matchIndex = input.indexOf(word);

   while (matchIndex >= 0) {
      result += input.substring(startIndex, matchIndex);
      // Insert spaces as filler
      for (int i = 0; i < word.length(); i++) {
         result += " ";
      }
      startIndex = matchIndex + word.length();
      matchIndex = input.indexOf(word, startIndex);
   }

   result += input.substring(startIndex);
   return result;
}
```

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

Write simple units of code

How you can measure it

Count the cyclomatic complexity

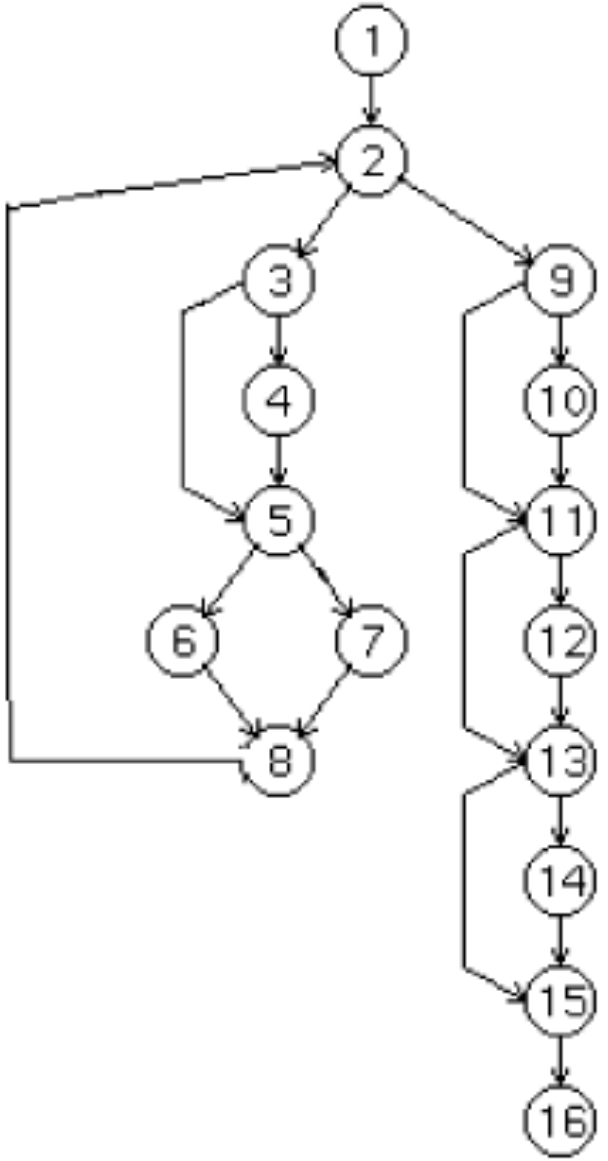
> Every branch point (if, case, for, &&, ||) is counted, and we add 1 to the total

```java
/**
 * Replaces all occurrences of a word in a given string with blanks.
 */
public static String clearAllMatches(String input, String word) {
   String result = "";
   int startIndex = 0;
   int matchIndex = input.indexOf(word);

   while (matchIndex >= 0) {
      result += input.substring(startIndex, matchIndex);
      // Insert spaces as filler
      for (int i = 0; i < word.length(); i++) {
         result += " ";
      }
      startIndex = matchIndex + word.length();
      matchIndex = input.indexOf(word, startIndex);
   }

   result += input.substring(startIndex);
   return result;
}
```

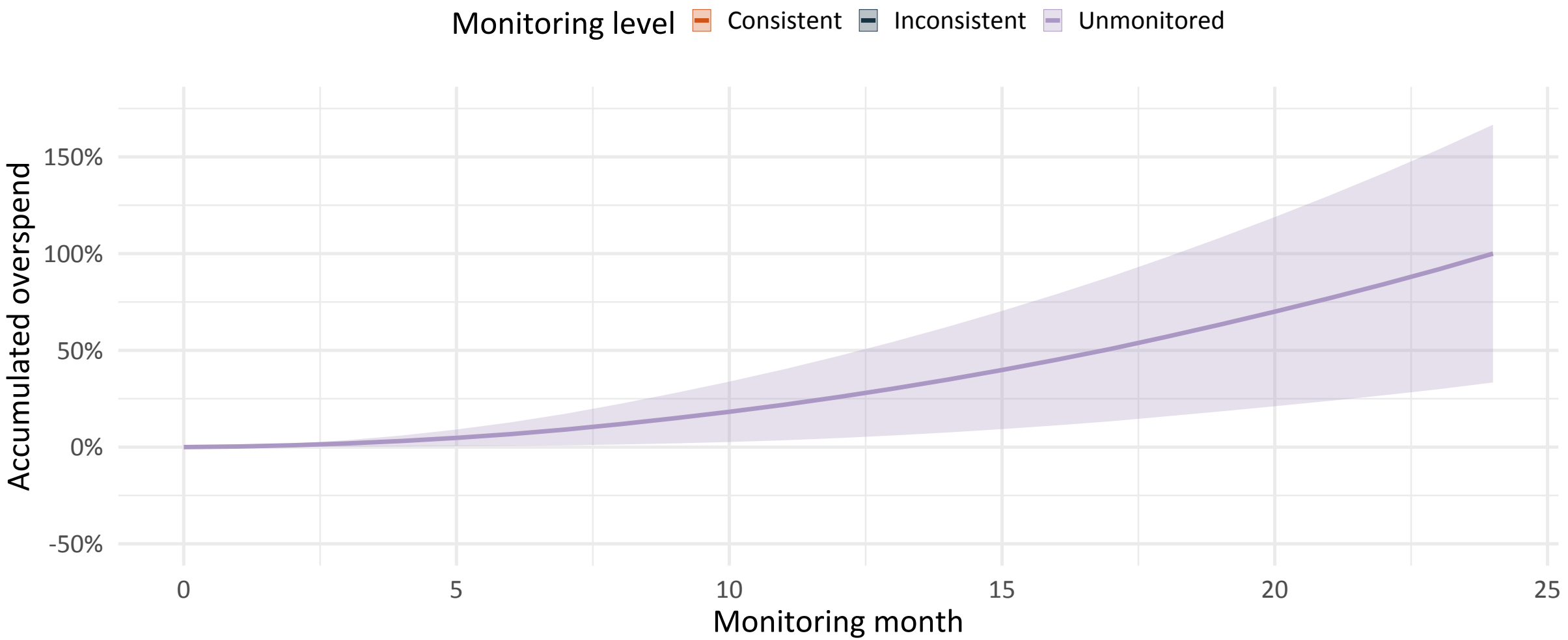
2 branch points + 1 = cyclomatic complexity of 3

Background: paths in control flow graph

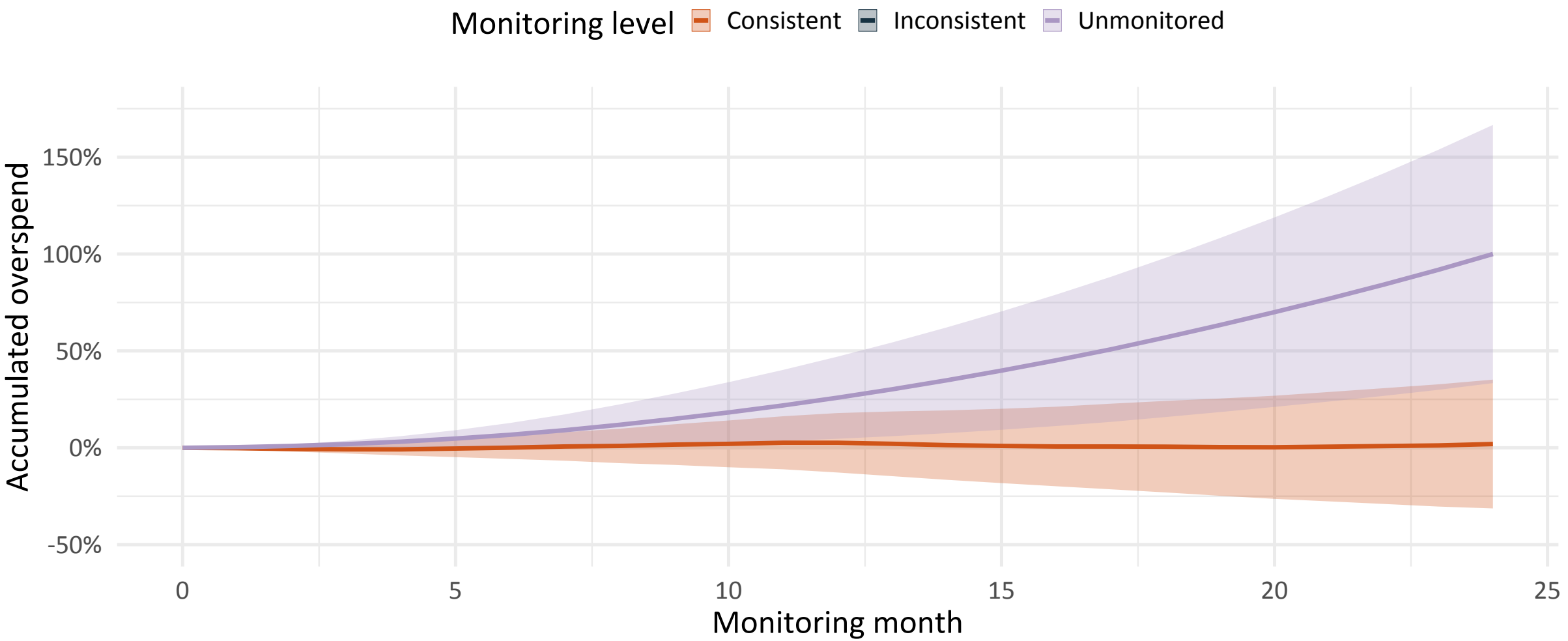
```
stcode getlist(char *lin, int *i, stcode *status)
/*  1 */ {
/*  1 */  int num, done;
/*  1 */  line2 = 0;
/*  1 */  nlines = 0;
/*  1 */  done = (getone(lin,i,&num,status)!=OK);
/*  2 */  while (!done)
/*  3 */  {
/*  3 */    line1 = line2;
/*  3 */    line2 = num;
/*  3 */    nlines++;
/*  3 */    if (lin[*i]==SEMICOL)
/*  4 */      curln = num;
/*  5 */    if ((lin[*i]==COMMA)||(lin[*i]==SEMICOL))
/*  6 */    {
/*  6 */      *i = *i + 1;
/*  6 */      done = (getone(lin,i,&num,status)!=OK);
/*  6 */    }
/*  7 */     else
/*  7 */       done = 1;
/*  8 */  }
/*  9 */  nlines = min(nlines,2);
/*  9 */  if (nlines == 0)
/* 10 */    line2 = curln;
/* 11 */  if (nlines <= 1)
/* 12 */    line1 = line2;
/* 13 */  if (*status != ERR)
/* 14 */    *status = OK;
/* 15 */  return(*status);
/* 16 */ }
```



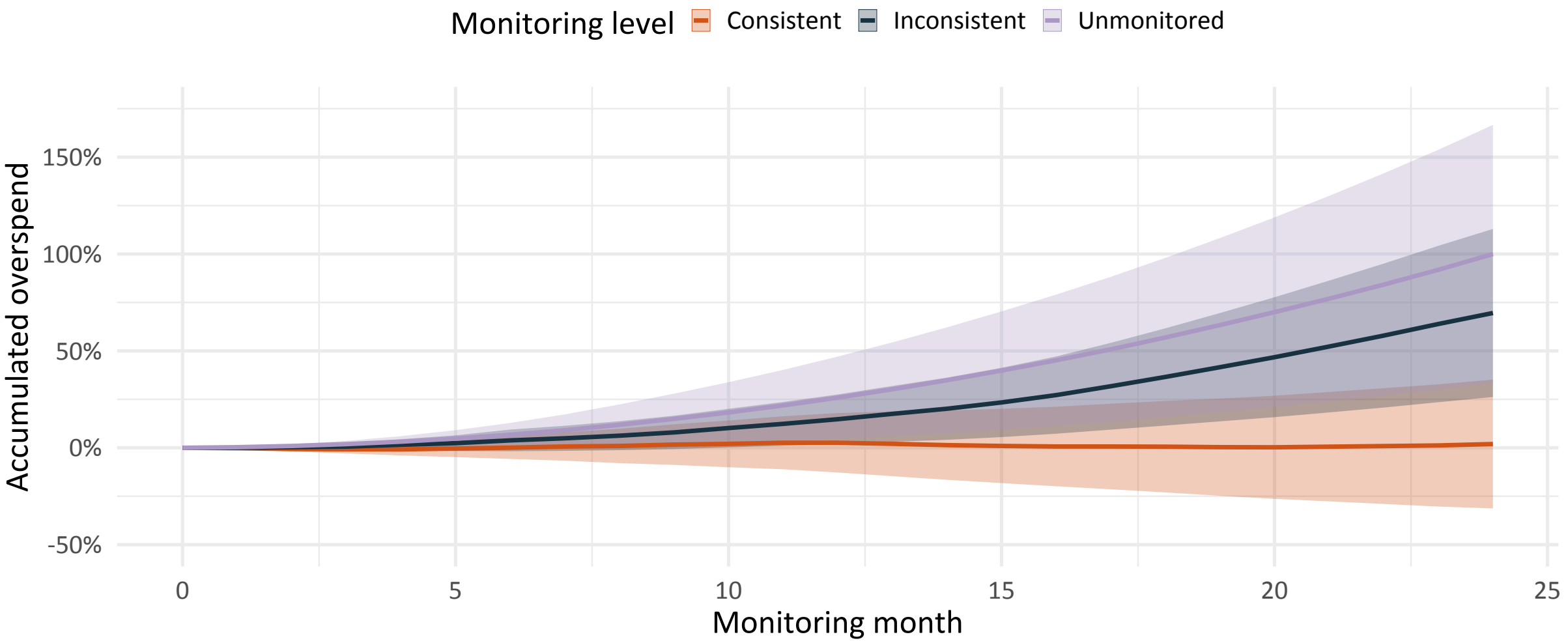
Consistent monitoring of each system leads to lower cost of ownership?



Consistent monitoring of each system leads to lower cost of ownership?



Consistent monitoring of each system leads to lower cost of ownership?



Projects

- Commit Classification from Source Code Changes
- Does this metric matter? Automated support for interpreting software metric values.
- From anecdotes to theory: what factors cause SCRUM teams to be effective?
- Finding Alternatives for Dependencies using Digraph Communities
- Can we proof that maintainable software leads to higher development efficiency?
- Intelligent refactoring advice
- The impact of team stability on architecture
- Putting Software into Context with Knowledge Graphs
- Transformer-based call graph pruning
- Can we measure the output of developers?
- Towards A Practical Model for Software Sustainability

Software Improvement Group

Contact

+31 20 314 09 50

m.bruntink@sig.eu

@sig_eu