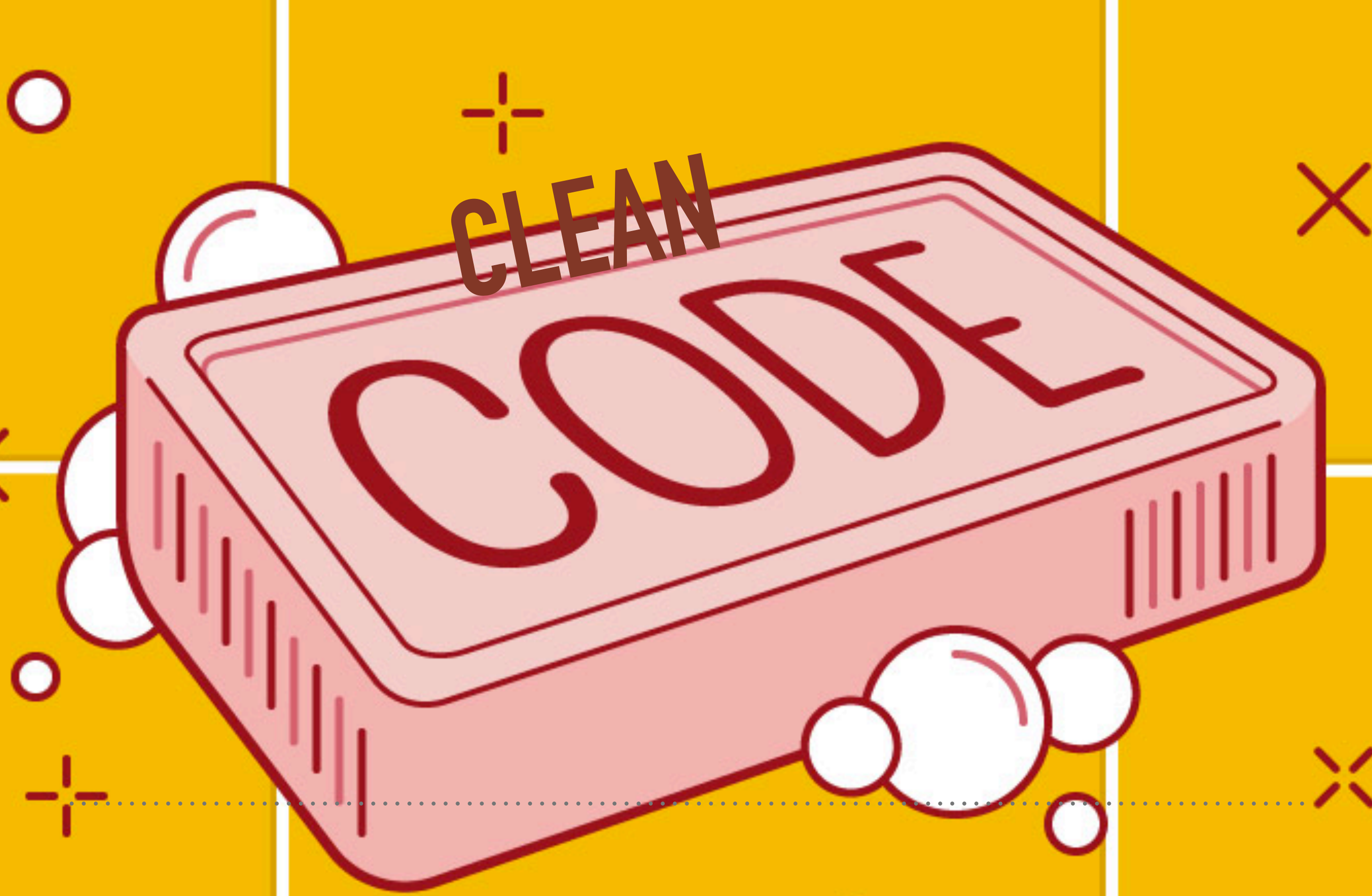


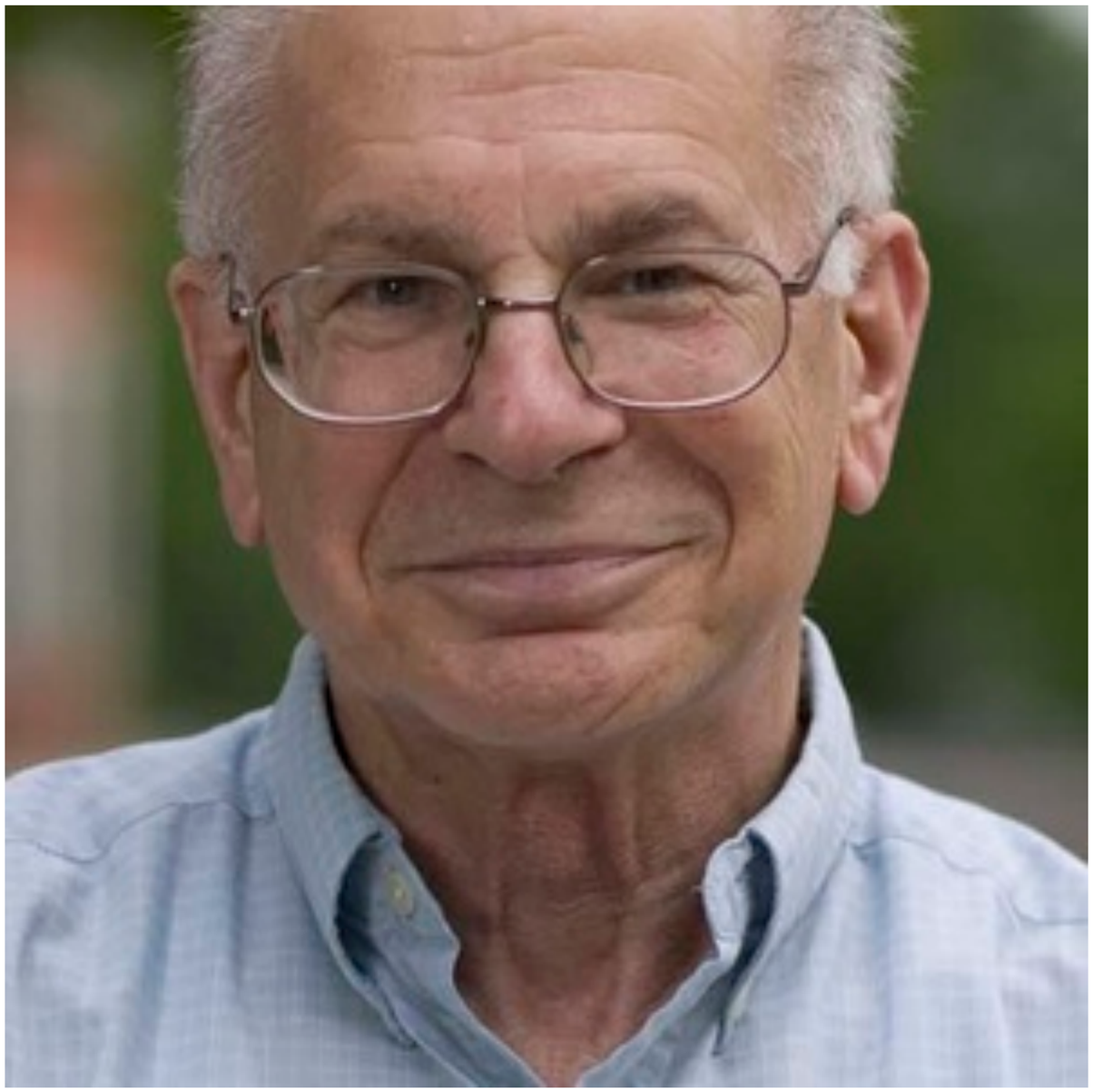
Ganesh Samarthyam
ganesh@codeops.tech

GETTING STARTED

WHAT IS COST OF BALL?

- Jack went to a shop with his son Jackson
- Jack bought a bat
- Jackson bought a ball
- Totally they paid 110 rupees
- Jack’s bat cost 100 rupees more than Jacksons ball
- How much does the ball cost?

INTUITION VS. REASONING



- Daniel Kahneman, in his Nobel prize receiving lecture in 2002, gave this example while talking about differences between intuition and reasoning!
- If we proceed logically and say cost of ball is x and cost of bat is x + 100 and total is 110 cents, we'll arrive at x = 5 rupees, which is the correct answer!

92

9/9

0800 Arctan started
1000 " stopped - arctan ✓ { 1.2700 9.037 847 025
9.037 846 795 correct
13"sc (032) MP - MC ~~1.9821440000~~ ~~2.13047641 5 (033)~~ 4.61592505 9(-2)
(033) PRO 2 2.130476415
correct 2.130676415
Relays 6-2 in 033 failed special speed test
In relay " 11.00 test.
Relays changed
Relay 2145
Relay 3370
1100 Started Cosine Tape (Sine check)
1525 Started Multy Adder Test.
1545 Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.
~~1655~~ 1630 Arctangent started.
1700 closed down.

ORIGIN OF TERMS BUGS AND PATCH

- Mark II - one of the earliest, electromechanical, computer - was used in US Navy. During 1947, when it was being tested, calculations were giving wrong results
  - To find out what was going on, the computer was opened!
    - There was a “moth" (an insect aka a bug) found that caused the malfunction.
    - The bug was removed and pinned on the log report
- The word “patch" - meaning applying fix for a bug in a program - comes from olden days when programmers used to fix a program stored on paper tape by using glue and paper!

NOT ALL BUGS BECOME DEFECTS

```
/* C1: Will this condition get executed? */
void writeStrToFile(char* str) {

  /* C2: Will fopen fail? */
  FILE* fp = fopen(''file.txt'', ''w'');

  /* C3: Will str be non-null? */
  if(str)
    fprintf(fp, str);
  /* more code ... */
}
```

KNOWN BUGS ARE JUST TIP OF THE ICEBERG!

LARGE LEAKS OR SMALL LEAKS – WHICH IS IMPORTANT?

- One can argue that large leaks are more important than the small leaks because they obviously hog more memory space and hence important to fix.
  - But: It is easy to find and fix large leaks and if we don't fix them, they might result crashing the system with out-of-memory error.
  - However, small leaks are harder to find and fix. If we don’t find and fix them, then they might grow slowly and then bring down the system eventually.
  - Fixing the small leaks will take more time and effort than the large ones.
- Hence, it is important to understand the impact of “small bugs” or “seemingly insignificant problems”.

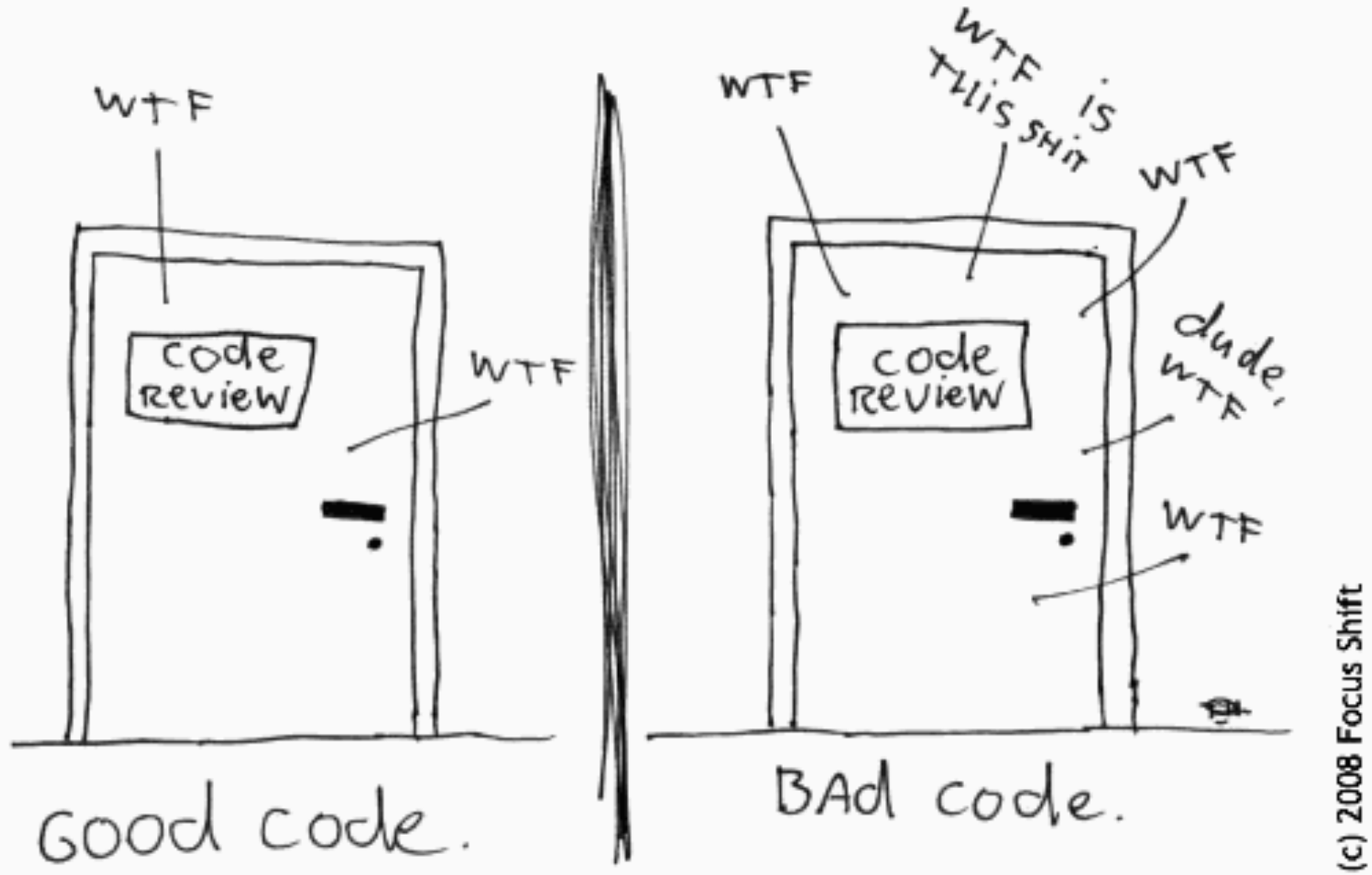
BUG – KINDS

Bug type	Description (programming errors perspective)	Example
Omission	Not doing something that should have been done	Missing null-check before dereferencing a variable
Incorrect fact	Mis-representation of a concept or a mismatch from general expectation or wrong implementation than what is expected	Incorrect method implementation of equals method in Java (Equals in C#)
Inconsistency	Part of the implementation is inconsistent with some other part	Only one of getter or setter method synchronized
Ambiguity	Confusing implementation which can result in mis-interpretation or mis-understanding	When a base class member is hidden in a derived class, an unqualified access to that member is not clear (ambiguous) to the reader of the code
Extraneous Information	Unnecessary parts in the implementation that should not be part of the implementation or that can be safely removed	Explicit cast for converting from derived to base type reference

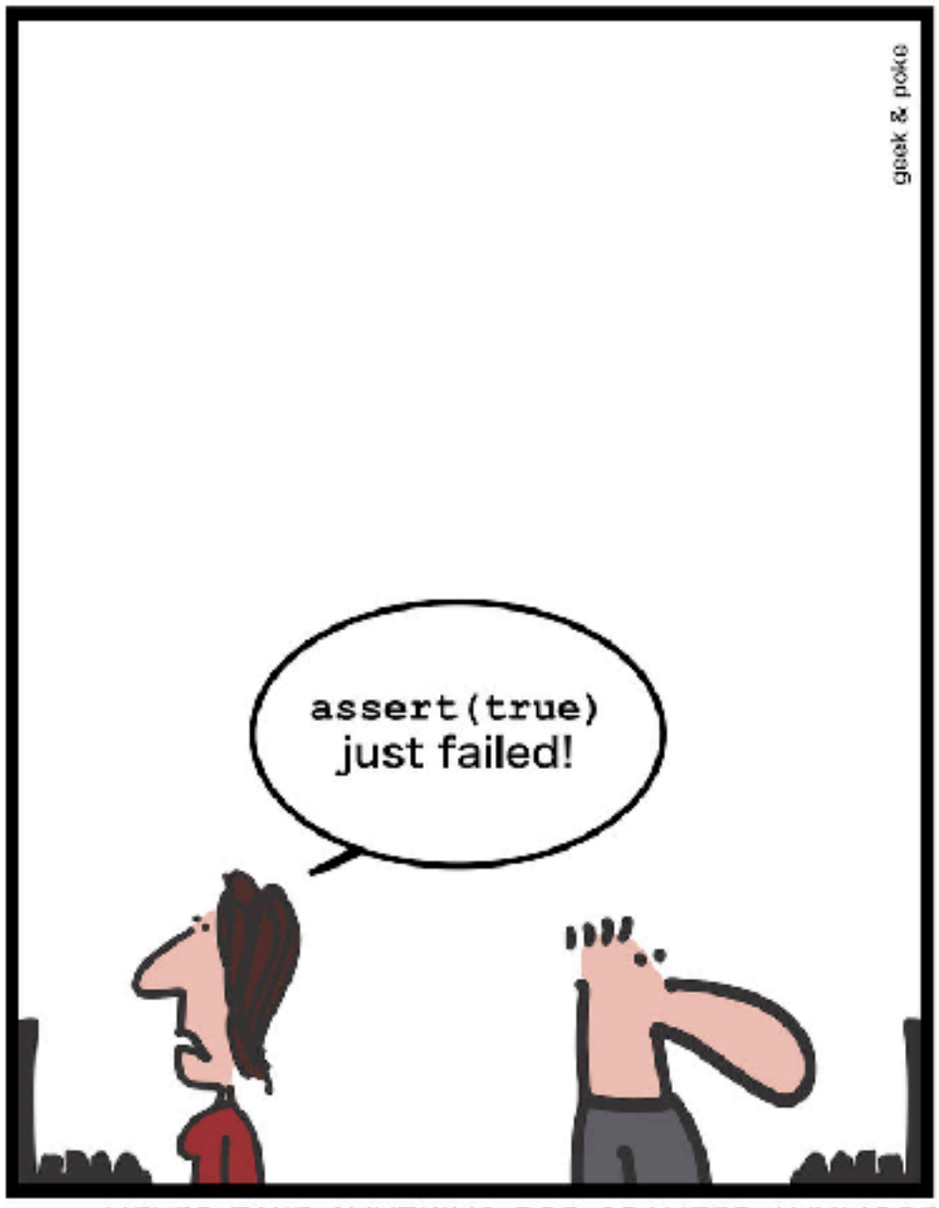
PURCHASE AND WINDER'S BUG TAXONOMY

Bug Type (what kind of fault)	History (where it could occur)	Mindset (how human error can cause it)
Perceptual	Initial requirements definition	Inadequate problem analysis; communication failure
Specification	Design	Inadequate problem analysis; specification error
Abstraction	Design of top-level objects and protocols	Inadequate knowledge of design techniques; premature hierarchic factoring or levelling
Algorithmic	Design of top-level objects and protocols	Misunderstood specification; incorrect implementation
Reuse	Design	Inadequate knowledge; Incorrect or misunderstood component, hierarchy or protocol
Logical	Implementation	Misuse of reuse component; ad hoc extensions of inheritance
Semantic	Implementation	Inadequate programming knowledge; misuse of target environment services
Syntactic	Implementation	Inadequate programming knowledge; typographic error
Domain adherence	Runtime	Inadequate knowledge of application or target services

The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE



WHAT 2016 TAUGHT US



NEVER TAKE ANYTHING FOR GRANTED ANYMORE

DOES THIS CODE OPEN IE OR FIREFOX?

```
#include <stdio.h>

int main() {
    http://www.google.com
    printf("hello");
}
```

YES, CODING IS FUN, BUT WATCH FOR ICE!

OPEN SOURCE CODE SAMPLES

- ACE is an open-source framework that provides many components and patterns for developing high-performance, distributed real-time and embedded systems. ACE provides powerful yet efficient abstractions for sockets, demultiplexing loops, threads, and synchronization primitives.
- DemoGL is an OpenGL-based, C++, Win32 execution platform for audiovisual effects. It allows the development of stand-alone executables or screensavers under the Microsoft Windows platform.
- OpenCL is (or rather, is trying to be) a portable, easy-to-use, and efficient C++ class library for cryptography.
- QtChat is a Yahoo! Chat client based on the Qt library. It includes such features as a plethora of autoignore options, highlighting, private messaging capability, and others.

Source: Diomidis Spinellis. *Code Quality: The Open Source Perspective*. Addison Wesley, 2006

OPEN SOURCE CODE SAMPLES

- Purenum is a C++ bignum library for programmers. Its unlimited-width Integer type works with all of the C++ math operators, but unlike the int type there can never be overflow errors. Division by zero and running out of memory are catchable exceptions. Optimized inline code is used, and the software Integer operations for single-width values run almost as quickly as hardware int operations do. An Array type provides for resizeable arrays of bignums.
- The socket++ library defines a family of C++ classes that can be used more effectively for using sockets than directly calling the underlying low-level system functions. One distinct advantage of Socket++ is that it has the same interface as that of the iostream so that the users can perform type-safe input/output.
- The Visual Component Framework is a C++ framework designed as a completely cross-platform GUI framework. It was inspired by the ease of use of environments like NEXTStep's Interface Builder, Java IDEs like JBuilder, Visual J++, and Borland's Delphi and the C++ Builder.

Source: Diomidis Spinellis. *Code Quality: The Open Source Perspective*. Addison Wesley, 2006

WRITING READABLE CODE

IS READABILITY IMPORTANT?

```
533   long
534   xxx(boolean st)
535   {
536           static long f, s;
537           long r;
538
539           if (st) {
540                   f = 37;
541                   s = 7;
542                   return(0L);
543           }
544           r = ((f * s) + 9337) % 8887;
545           f = s;
546           s = r;
547           return(r);
548   }
```

https://github.com/ctdk/bsdgames-osx/blob/master/rogue/score.c

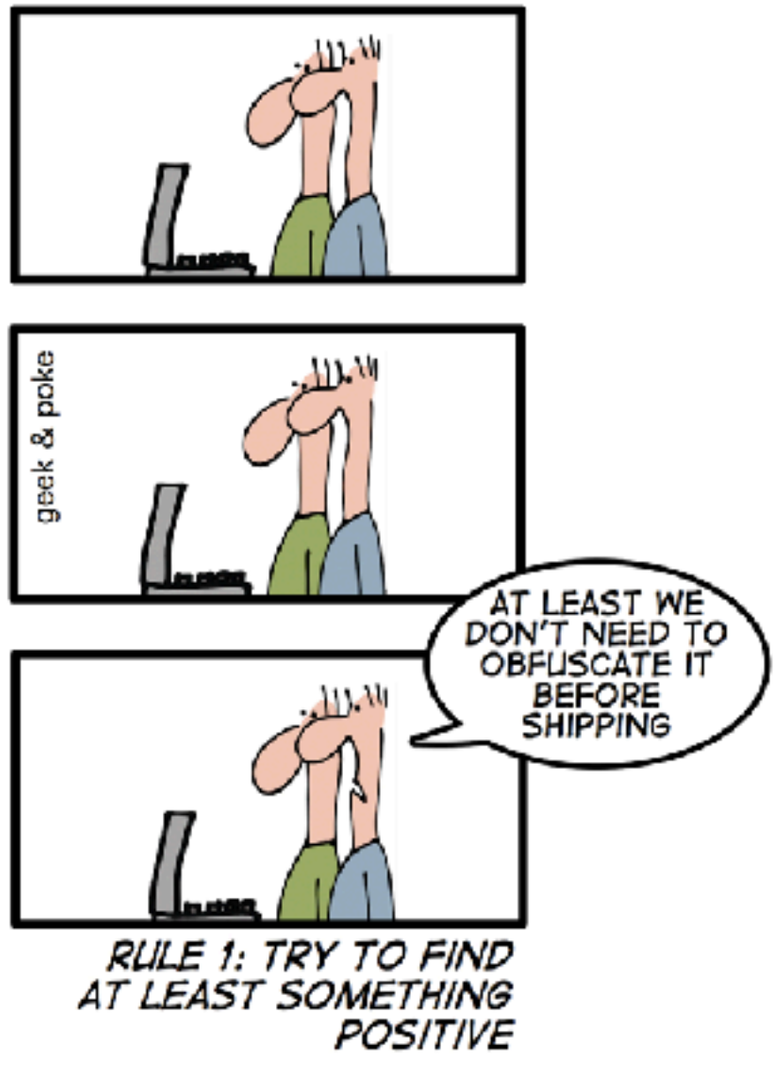
IS THIS CODE OKAY?

```
void mystrcpy(char *t, char *s) {
    while(*t++ = *s++);
}
```

HOW ABOUT THIS ONE?

```
#include <stdio.h>

int main() {
     int c = 0, r = 0;
     do {
          if(c >= r)
               printf(" %c", (~c & r) ? '.' : '+');

          c++;
          if (c >= 32) {
               c = 0;
               r++;
               printf("\n");
          }
     } while (r != 32);
}
```

HOW TO MAKE A GOOD CODE REVIEW



RULE 1: TRY TO FIND AT LEAST SOMETHING POSITIVE

C TO C++

FIND THE BUG

```
#include <iostream>
using namespace std;

int main() {
     const char *likes[] = { "cooking"
                             "cats",
                              "singing"
                           };

     for(auto &like : likes) {
          cout << like <<endl;
     }
}
```

ALL IS WELL?

```
if (c1 < 64 && legal_ansi_char_array[c1] & 8);
  return err_val;
if (c1 < c2)
  return -1;
```

FIND THE BUG

```
#include <iostream>
#include <array>
#include <string>
using namespace std;

int main() {
     std::array<string, 3> likes =
          { "cooking", "cats", "singing" };
     int i = 0;
     while(i < likes.size());
         cout << likes[i++] <<endl;
}
```

WILL THIS PROGRAM EVER WORK?

```
#include <stdio.h>

int main() {
  int i = "hi";
  printf(i);
}
```

```
sh-4.2$ gcc --version
gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-4)
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

sh-4.2$ gcc -w -o main *.c; ./main
hi
sh-4.2$
```

WILL THIS PROGRAM EVER WORK?

C++ enforces type-safety; however, C is “weakly typed”!

```
$ g++ williteverwork.cpp
williteverwork.cpp:4:6: error: cannot initialize a variable of type 'int' with
an lvalue of type 'const char [3]'
        int i = "hi";
            ^   ~~~~
1 error generated.
```

USE C++ FEATURES INSTEAD OF C FEATURES!

C feature	Preferred C++ alternative
arrays	std::vector
char*s and wchar*s	std::strings and std::wstrings
C-style casts	C++ style casts
C-style comments	C++ style comments
memcpy, memcmp etc for structs	operator =, == etc. for classes
unions	inheritance
bitfields	std::bitset
global static variables	unnamed namespaces
malloc and free	new and delete
setjmp and longjmp	exception handling constructs
gotos	other control flow constructs
hand-coded algorithms	STL algorithms
custom data structures	STL containers

- Prefer const and inline to #define
- Prefer <iostream> to <stdio.h>
- Prefer new and delete to malloc and free
- Prefer C++ style comments

(Item numbers 1 to 4 in Effective C++)

- Scott Meyers

When in Rome, do as the Romans do

LanguageTies·com/proverbs

ESSENTIALS

LOOPING ERRORS

- On Dec 31 2008, all Zune media players froze!

- Cause traced back to a non-terminating loop error in code

```
year = ORIGINYEAR; /* = 1980 */

while (days > 365)
{
    if (IsLeapYear(year))
    {
        if (days > 366)
        {
            days -= 366;
            year += 1;
        }
    }
    else
    {
        days -= 365;
        year += 1;
    }
}
```

IS THIS CODE OKAY?

```
int LCM(int a, int b) {
        int result;
        for(int i = 1; i <= b; i++) {
                if( (a * i) % b == 0) {
                        result = a * i;
                        break;
                }
        }
        return result;
}
```

FIND THE BUG

```
#include <iostream>
using namespace std;

bool validateWaterTemperature(int temperature) {
        return (0 <= temperature <= 100);
}

int main() {
        cout << std::boolalpha << validateWaterTemperature(250);
}
```

UNDERSTANDING SHORT-CIRCUIT EVALUATION

```
#include <iostream>
using namespace std;

bool foo() { cout << "foo" << endl; return true; }
bool bar() { cout << "bar" << endl; return false; }

int main() {
  bool b1 = foo() ||  bar();
  bool b2 = foo() | bar();
  cout << std::boolalpha <<  b1;
  cout << std::boolalpha << b2;
}
```

UNDERSTANDING OVERFLOW

```
// Otherwise, do a binary search
int low = 0;
int high = list.size()-1;

while (low <= high) {
    int mid =(low + high)/2;
    Object midVal = list.get(mid);
    int cmp = ((Comparable)midVal).compareTo(key);

    if (cmp < 0)
    low = mid + 1;
    else if (cmp > 0)
    high = mid - 1;
    else
    return mid; // key found
}
return -(low + 1);  // key not found
```

Nearly all binary searches and mergesorts are broken! - Josh Bloch
googleresearch.blogspot.com/2006/06/extra-extra-read-all-about-it-nearly.html

FIND THE BUG

```
#include <stdio.h>

struct bits {
    int b : 1;
} bits;

int main() {
    bits.b = 1;
    printf("%d", bits.b);
}
```

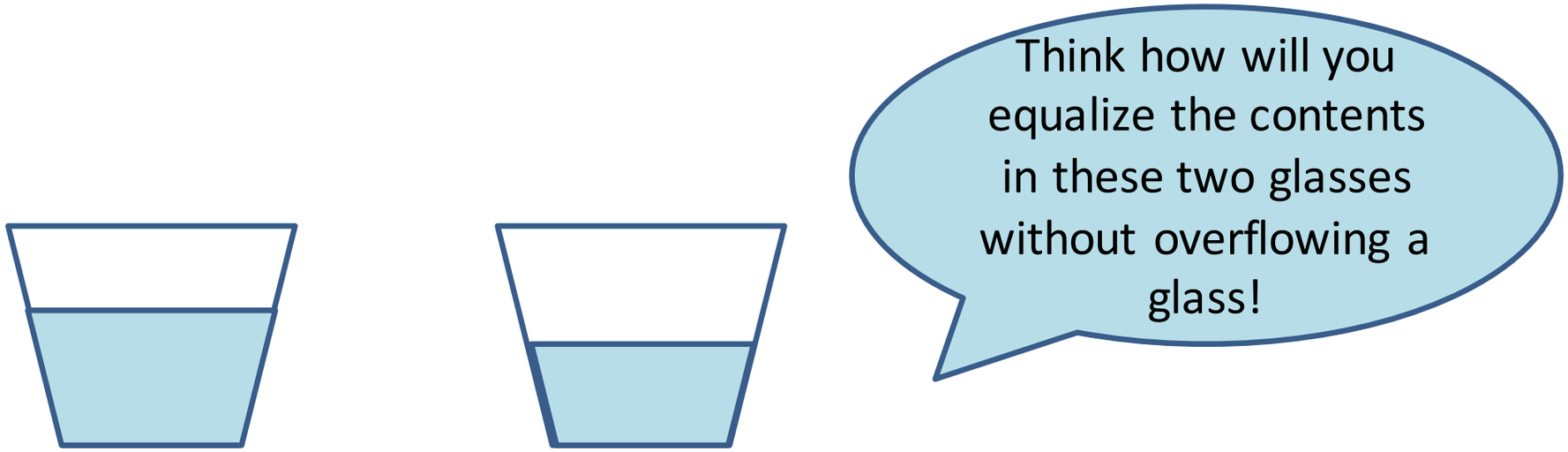
FIND THE BUG

```
#include <iostream>
using namespace std;

int main() {
    // millisecs * secs * mins * hours * days
    long hundred_days = 1000 * 60 * 60 * 24 * 100;
    cout << hundred_days << endl;
}
```

YEAR 2038 BUG

- Most Unix/Linux systems use 32-bit signed integer for representing time (typedef'ed as time t). For each second, the integer is incremented.
- As we know, the range of values a signed integer can represent is (-2 power 31) to (2 power 31) - 1, which is -2147483648 to 2147483647. This represents the time starting from 00:00:00 January 1, 1970 (the time when Unix was developed) to 3:14:07 January 19, 2038. So, once the time counter value reaches 2147483647, it will overflow to -2147483648.
- This problem is known as The Year 2038 Bug. For example, it can cause bugs and problems in the applications that uses system time for calculations. Say, calculation of home loan interest for 30 years can give wrong results.

UNDERSTANDING OVERFLOW



mid = high + (high - low) / 2;

FIND THE BUG

```
#include <iostream>
#include <cmath>
#include <cassert>

int main() {
    double two = 2.0;
    double sqrt_of_two = sqrt(two);
    assert((sqrt_of_two * sqrt_of_two) == two);
}
```

```
Assertion failed: ((sqrt_of_two * sqrt_of_two) == two), function
main, file sqrt.cpp, line 8.
Abort trap: 6
```

FIND THE BUG

```
#include <iostream>
using namespace std;

int main() {
    cout << "start" << endl;
    for (float f = 0.0; f < 20000000.0; f++)
        ; // just loop over, do nothing
    cout << "done" << endl;
}
```

FIND THE BUG

```
#include <iostream>
using namespace std;

int factorial(int x) {
     if(x <= 1)
         return x;
     else
         return factorial(--x) * x;
}

int main() {
     cout << "factorial(4): " << factorial(4) << endl;
}
```

FIND THE BUG

```
$ cat one.cpp
extern int i;

int j = i + 10;

$ cat two.cpp
#include <iostream>
using namespace std;

extern int j;

int i = j + 20;

int main() {
  cout << "i = " << i << " j = " << j << endl;
}
```

ORDER OF INIT PROBLEM

```
$ cat one.cpp
extern int i;

int j = i + 10;

$ cat two.cpp
#include <iostream>
using namespace std;

extern int j;

int i = j + 20;

int main() {
  cout << "i = " << i << " j = " << j << endl;
}

$ g++ one.cpp two.cpp ; ./a.out
i = 30 j = 10

$ g++ two.cpp one.cpp ; ./a.out
i = 20 j = 30
```

BOOLEAN BUGS

"Boolean bugs occur because the mathematical precision of Boolean algebra has virtually nothing to do with the equivalent English words. When we say 'and', we really mean the Boolean 'or' and vice versa".

Source: Matthew Telles and Yuan Hsieh. The Science of Debugging.Coriolis Group Books, 2001

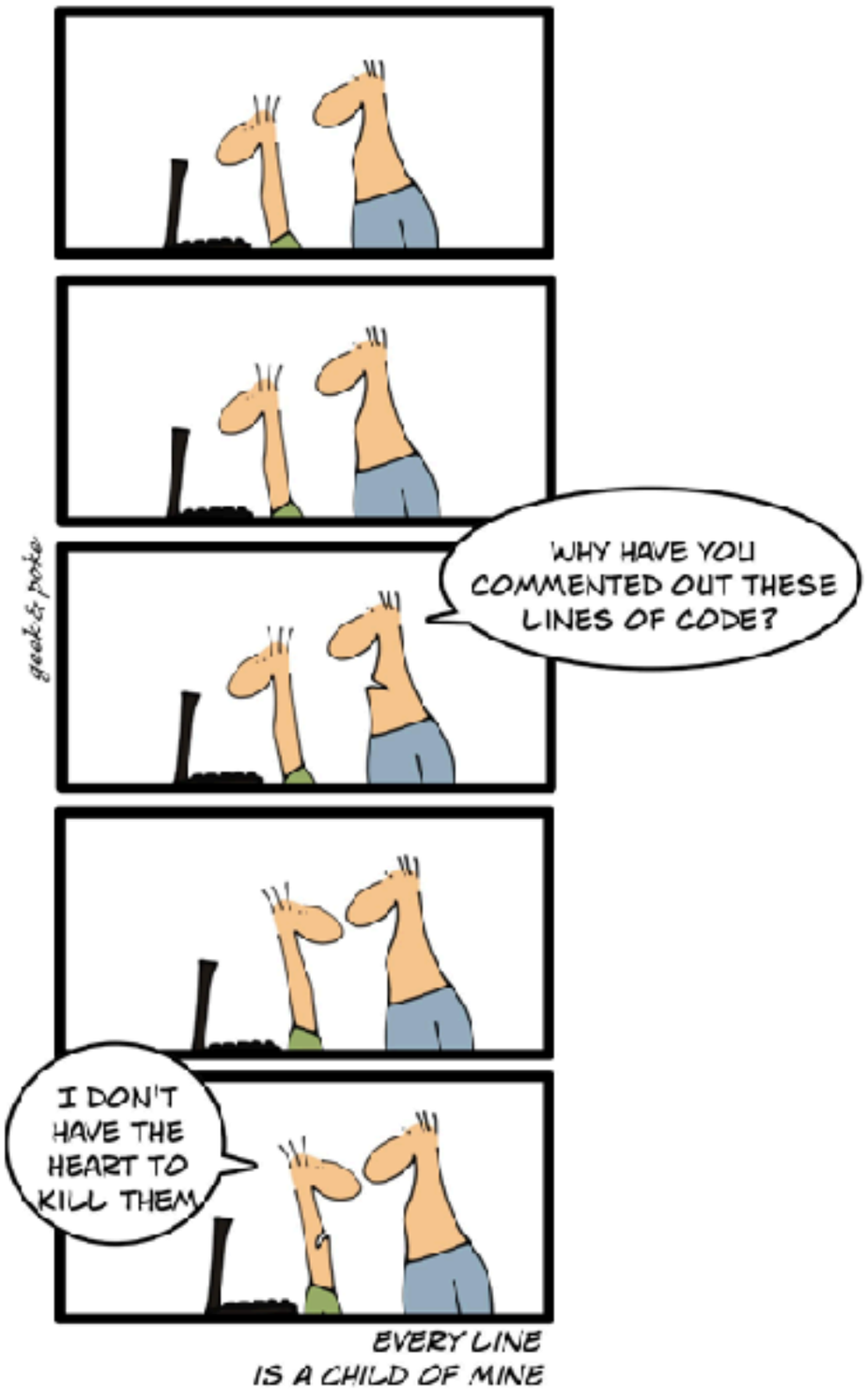
boolean happiness = promotion || hike;

boolean happiness = promotion && hike;

IS THIS COMMENT GOOD & HELPFUL?

```
/*
 * find first satisfactory answer
 *
 *       answer --> +-------------+   ( MESSAGE )
 *                  |    Header   |
 *                  +-------------+
 *                  |  Question   | the question for the name server
 *                  +-------------+
 *                  |   Answer    | RRs answering the question
 *                  +-------------+
 *                  | Authority   | RRs pointing toward an authority
 *                  | Additional  | RRs holding additional information
 *                  +-------------+
 */
```

ftp://ftp.fu-berlin.de/unix/NetBSD/misc/repositories/hg/src.git/lib/libc/net/getnetnamadr.c

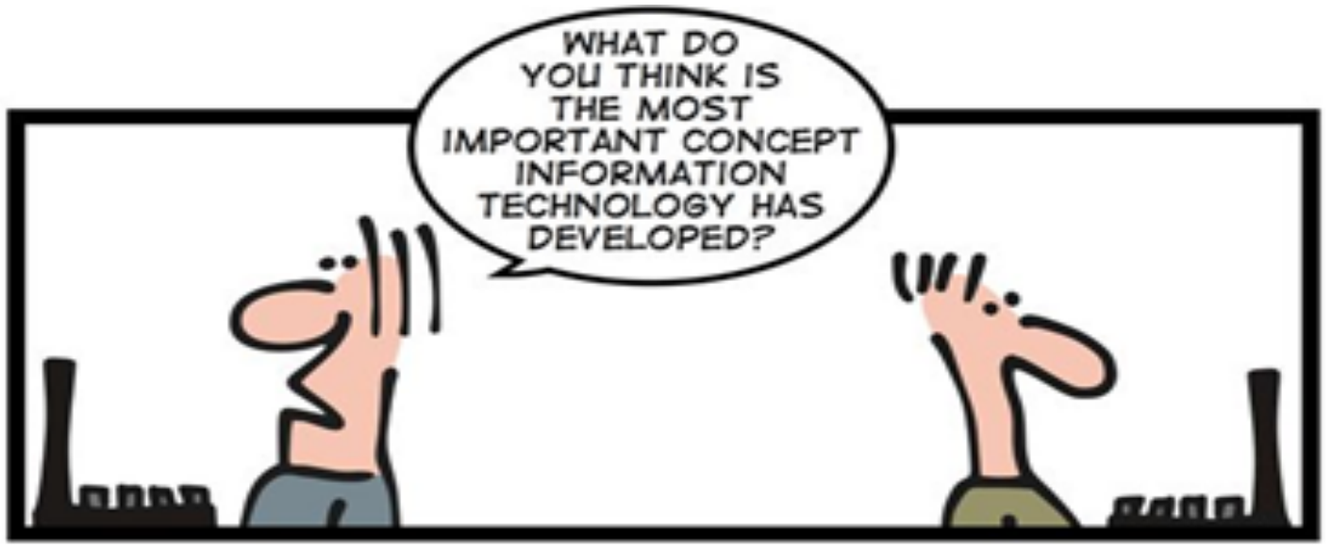

WHY HAVE YOU COMMENTED OUT THESE LINES OF CODE?
I DON'T HAVE THE HEART TO KILL THEM
EVERY LINE IS A CHILD OF MINE

“

“...in programming, the devil hides in the details!”

- Niklaus Wirth

CODE DUPLICATION


WHAT DO YOU THINK IS THE MOST IMPORTANT CONCEPT INFORMATION TECHNOLOGY HAS DEVELOPED?


CTRL-C
CTRL-V

Type 1

- **exactly identical** except for variations in whitespace, layout, and comments

Type 2

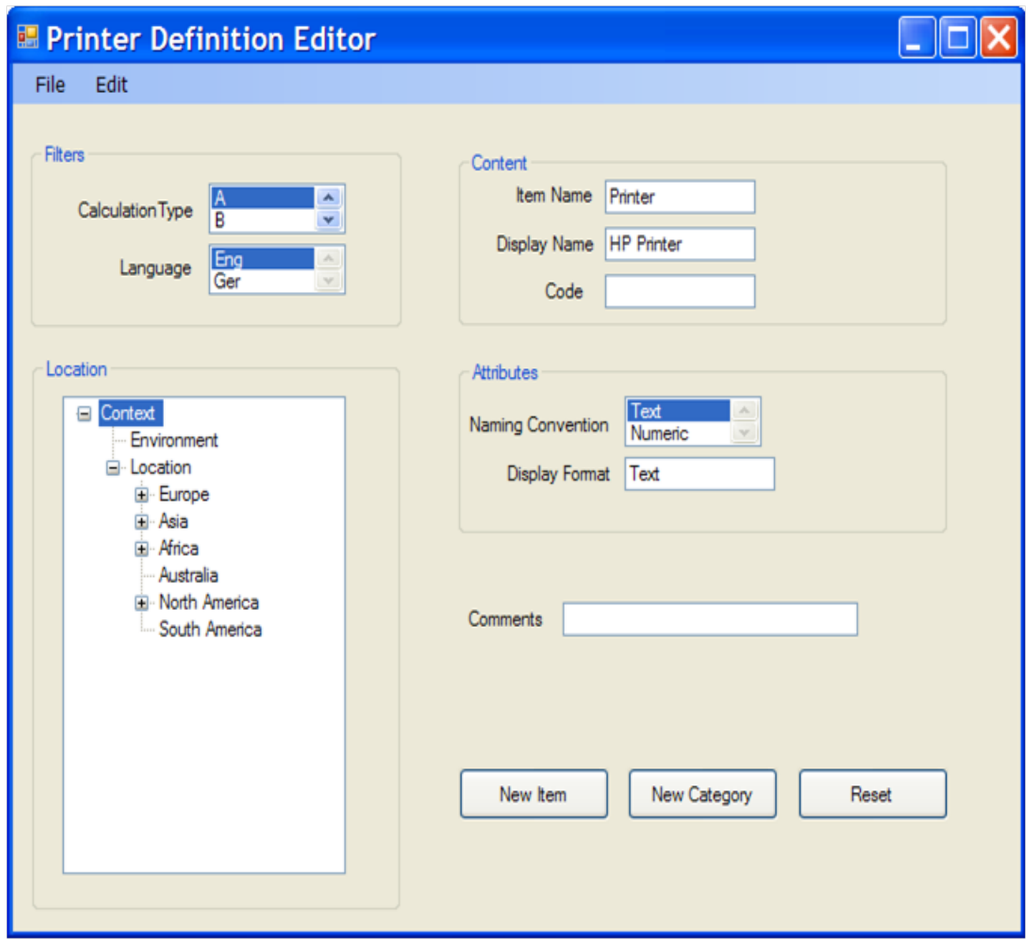
- **syntactically identical** except for variation in symbol names, whitespace, layout, and comments

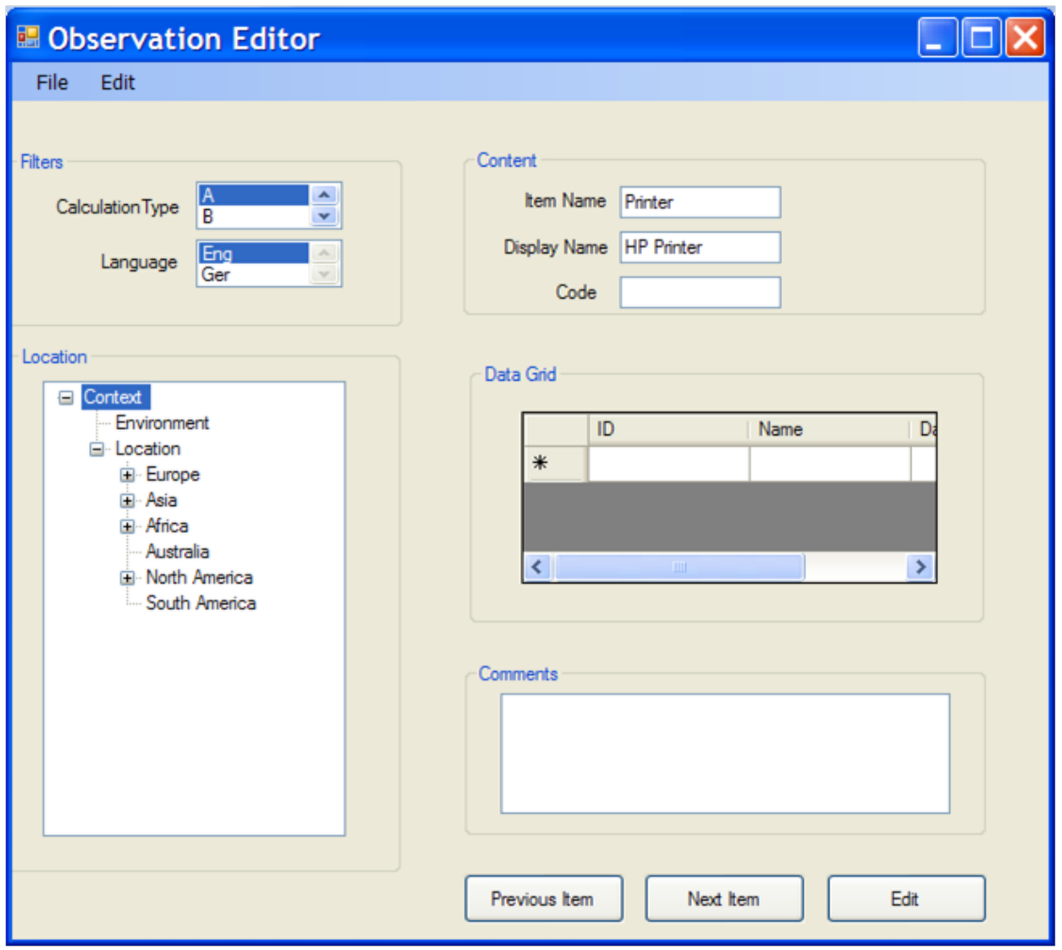
Type 3

- identical except some **statements changed, added, or removed**

Type 4

- when the fragments are **semantically identical** but implemented by syntactic variants


Printer Definition Editor
File
Edit
Filters
Calculation Type
A
B
Language
Eng
Ger
Location
Context
Environment
Location
Europe
Asia
Africa
Australia
North America
South America
Content
Item Name
Printer
Display Name
HP Printer
Code
Attributes
Naming Convention
Text
Numeric
Display Format
Text
Comments
New Item
New Category
Reset


Observation Editor
File
Edit
Filters
Calculation Type
A
B
Language
Eng
Ger
Location
Context
Environment
Location
Europe
Asia
Africa
Australia
North America
South America
Content
Item Name
Printer
Display Name
HP Printer
Code
Data Grid
ID
Name
Comments
Previous Item
Next Item
Edit

MEMORY HANDLING

IS THIS CODE OKAY?

```
#include <iostream>
#include <cstdlib>
using namespace std;

class String {
public:
  String() { cout << "String::String()" << endl; }
  ~String() { cout << "String::~String()" << endl; }
};

int main()
{
  String *stringArray = static_cast<String*>(malloc(10 * sizeof(String)));

  free(stringArray);
}
```

IS THIS CODE OKAY?

```
#include <iostream>
#include <cstdlib>
using namespace std;

class String {
public:
  String() { cout << "String::String()" << endl; }
  ~String() { cout << "String::~String()" << endl; }
};

int main()
{
  String *stringArray = new String[10];

  free(stringArray);
}
```

IS THIS CODE OKAY?

```
#include <iostream>
#include <cstdlib>
using namespace std;

class String {
public:
  String() { cout << "String::String()" << endl; }
  ~String() { cout << "String::~String()" << endl; }
};

int main()
{
  String *stringArray = new String[10];

  delete stringArray;
}
```

ALLOC WITH ‘STRDUP’ AND RELEASE WITH ‘DELETE’?

```
class Word {
    public:
        Word(const char *str);
        Word(const Word &w);
        Word(const Word *w);
        virtual ~Word();
        Word& operator=(const Word &w);
    private:
        char *word;
};
```

Word class definition

Word's data (dynamically allocated)

```
Word::Word(const char *str) {
    if(str)
        word = strdup(str);
    else
        word = 0;
}
```

1 Constructor

Allocate memory

```
Word::Word(const Word &w) {
    if(w.word)
        word = strdup(w.word);
    else
        word = 0;
}
```

2 Copy constructor

Duplicate data

```
Word::~Word() {
    if(word)
        delete word;
}
```

3 Destructor

Dispose allocated memory

```
Word & Word::operator=(const Word &w) {
    if(word) delete word;
    if(w.word)
        word = strdup(w.word);
    else
        word = 0;
    return *this;
}
```

4 Assignment operator (incorrect implementation)

Dispose old data on the LHS

Duplicate the data on the RHS

Source: Diomidis Spinellis. *Code Quality: The Open Source Perspective*. Addison Wesley, 2006

Prefer new and delete to malloc and free (Item #3 in Effective C++)

Use the same form in corresponding uses of new and delete (Item #5 in Effective C++)

- Scott Meyers

WHAT IS THE BUG?

```
#include <cstddef>
#include <iostream>
using namespace std;

// example from Effective C++ book

class EnemyTarget {
public:

  EnemyTarget() {
    cout << "EnemyTarget()" << endl;
    ++numTargets;
  }

  EnemyTarget(const EnemyTarget&) {
    cout << "EnemyTarget(const EnemyTarget&)" << endl;
    ++numTargets;
  }

  ~EnemyTarget() {
    cout << "~EnemyTarget()" << endl;
    --numTargets;
  }

  static size_t numberOfTargets() {
    return numTargets;
  }

private:
  static size_t numTargets;              // object counter
};

size_t EnemyTarget::numTargets;
```

WHAT IS THE BUG? [CONTINUED . . .]

```
class EnemyTank: public EnemyTarget {
public:
  EnemyTank() {
    cout << "EnemyTank()" << endl;
    ++numTanks;
  }

  EnemyTank(const EnemyTank& rhs)
  : EnemyTarget(rhs)
  {
    cout << "EnemyTank(const EnemyTank& rhs)" << endl;
    ++numTanks;
  }

  ~EnemyTank() {
    cout << "~EnemyTank()" << endl;
    --numTanks;
  }

  static size_t numberOfTanks() {
    return numTanks;
  }

private:
  static size_t numTanks;         // object counter for tanks
};

size_t EnemyTank::numTanks;

int main()
{
  EnemyTarget *targetPtr = new EnemyTank;
  delete targetPtr;
}
```

Make sure base classes have virtual destructors (Item #14 in Effective C++)

- Scott Meyers

WHAT'S THE BUG?

```
#include <iostream>
#include <string>
using namespace std;

// example from effective c++ book
class Window {
public:
  string name() const {
    return "WindowName";
  }
  virtual void display() const {
    cout << "Window::display()" << endl;
  }
};

class WindowWithScrollBars: public Window {
public:
  virtual void display() const {
    cout << "WindowWithScrollBars::display()" << endl;
  }
};

void printNameAndDisplay(Window w)
{
  cout << w.name() << endl;
  w.display();
}

int main()
{
  WindowWithScrollBars wwsb;
  printNameAndDisplay(wwsb);
}
```

Prefer pass-by-reference to pass-by-value (Item #22 in Effective C++)

- Scott Meyers

DEFINING CLASSES

WHAT'S WRONG WITH THIS CODE?

```
#include <cstring>

// A poorly designed String class.
class String {

public:

  String(const char *value);
  ~String();
  // no copy ctor or operator=

private:
  char *data;

};
```

```
String::String(const char *value)
{
  if (value) {
    data = new char[strlen(value) + 1];
    strcpy(data, value);
  }
  else {
    data = new char[1];
    *data = '\0';
  }
}


inline String::~String() {
  delete [] data;
}


void doNothing(String localString)
{}
```

Based on: Effective C++: 55 Specific Ways to Improve Your Programs and Designs, Scott Meyers, Addison-Wesley Professional, 2005

EXAMPLES OF PROBLEMS WITH THAT CLASS

```
void problem1()
{
  String a("Hello");
  String b("Hello");
  b = a;
}
```

```
void problem2()
{
  String a("Hello");     // define and construct a

  {                      // open new scope

    String b("World");   // define and construct b

    b = a;               // execute default o=, lose b's memory

  }                      // close scope, call b's destructor

  String c = a;          // c.data is undefined!
                         // a.data is already deleted
}
```

```
void problem3()
{
  String s = "The Truth Is Out There";
  doNothing(s);
}
```

Based on: Effective C++: 55 Specific Ways to Improve Your Programs and Designs, Scott Meyers, Addison-Wesley Professional, 2005

Declare a copy constructor and an assignment operator for classes with dynamically allocated memory (Item #11 in Effective C++)

- Scott Meyers

FIND THE BUG

```
#include <iostream>
using namespace std;

class point {
private:
   int x;
   int y;
public:
   point(int x, int y) {
      x = x;
      y = y;
   }
   friend ostream &operator<<(ostream &output, const point &p) {
          output << "[x: " << p.x << ", y: " << p.y << "]";
          return output;
   }
};

int main() {
   point p(10, 20);
   cout << p <<endl;
}
```

FIND THE BUG

```
#include <iostream>
using namespace std;

class point {
private:
   int x;
   int y;
public:
   point() {
      // chain the constructors ...
      // pass -1 to indicate default invalid x and y values
      point(-1, -1);
   }
   point(int x, int y) {
      this->x = x;
      this->y = y;
   }
   friend ostream &operator<<(ostream &output, const point &p) {
            output << "[x: " << p.x << ", y: " << p.y << "]";
            return output;
   }
};

int main() {
   point p;
   cout << p << endl;
}
```

FIND THE BUG

```
#include <iostream>
#include <cstdlib>
using namespace std;

class String {
   int len;
   const char * data;
public:
   String() {  len = 0; data = 0; }
   ~String() { }
   String(const char *str) : data(str), len(strlen(data)) { }
   friend ostream& operator<<(ostream& s, const String& aString);
};

ostream& operator<<(ostream& s, const String& aString)
{
   s << aString.data << " of length " << aString.len;
   return s;
}

int main()
{
   String aString("Hello");
   cout << aString << endl;
}
```

> List members in an initialiser list in the order in which they are declared (Item #13 in Effective C++)
>
> *- Scott Meyers*

DETERMINE THE BEHAVIOUR

```
#include <iostream>
using namespace std;

void f(int) { cout << "f(int)" << endl; }

void f(char) { cout << "f(char)" << endl; }

int main()
{
  double d = 6.02;

  f(d);
}
```

Based on: Effective C++: 55 Specific Ways to Improve Your Programs and Designs, Scott Meyers, Addison-Wesley Professional, 2005

DETERMINE THE BEHAVIOUR

```
#include <iostream>
using namespace std;

class Base1 {
public:
  int doIt() { cout << "Base1::doIt()" << endl; }
};

class Base2 {
private:
  void doIt() { cout << "Base2::doIt()" << endl; }
};

class Derived: public Base1,          // Derived doesn't declare
               public Base2 {         // a function called doIt
};

int main()
{
  Derived d;

  d.doIt();
}
```

Based on: Effective C++: 55 Specific Ways to Improve Your Programs and Designs, Scott Meyers, Addison-Wesley Professional, 2005

“

Guard against potential ambiguity (Item #26 in Effective C++)

- Scott Meyers

FIND 6 DIFFERENCES BETWEEN THE TWO CLASSES!

```
class Nothing {}
```

```
class Nothing {
public:
  Nothing();
  Nothing(const Nothing& arg);
  ~Nothing();
  Nothing& operator= (const Nothing &arg);
};
```

HOW TO PROHIBIT CERTAIN OPERATIONS?

To Prohibit the ...	Declare ...
Direct instantiation of a class's object	The class's constructors protected
Derivation of a class	The class's constructors private
Copying of objects	A private assignment operator and copy constructor
Dynamic allocation of objects	A private operator `new()`
Static or automatic (stack) allocation of objects	A private destructor

Source: Diomidis Spinellis. *Code Quality: The Open Source Perspective*. Addison Wesley, 2006

EXCEPTION HANDLING

IS THIS CODE OKAY?

```
#include <vector>
#include <iostream>
using namespace std;

void print_vec(std::vector<int> v) {
  try {
    for(int i = 0; i <= v.size(); i++) {
      std::cout << v.at(i) << endl;
    }
  }
  catch(const std::exception& e) {
    std::cout << e.what() << endl;
  }
}

int main() {
  vector<int> avector = { 1, 4, 9, 16, 25 };
  print_vec(avector);
}
```

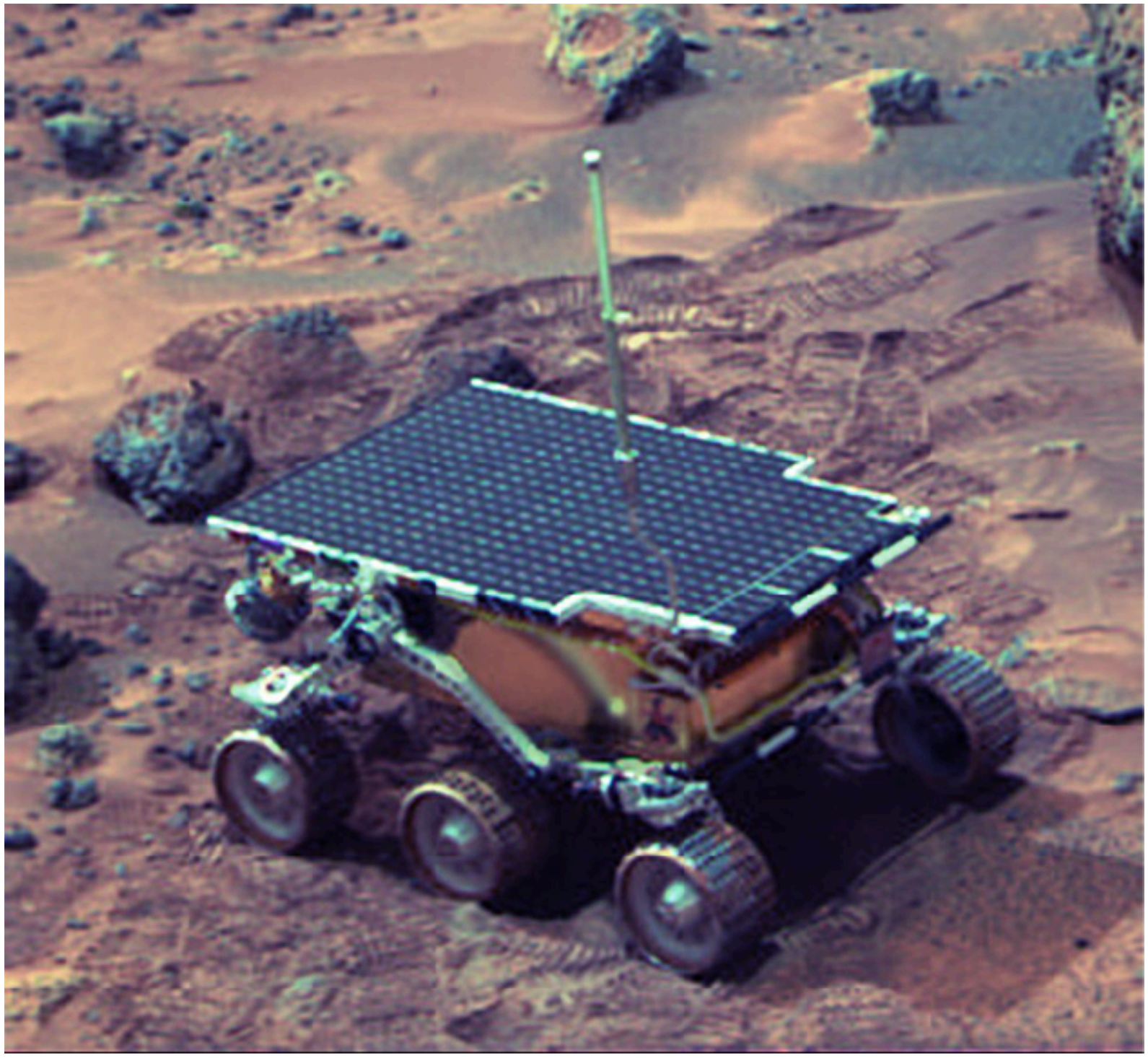
MULTI-THREADING

CONCURRENCY / MULTI-THREADING

I really really hate concurrency problems

EXAMPLE: DEADLOCKS

NASA'S MARS ROVER (SOJOURNER)



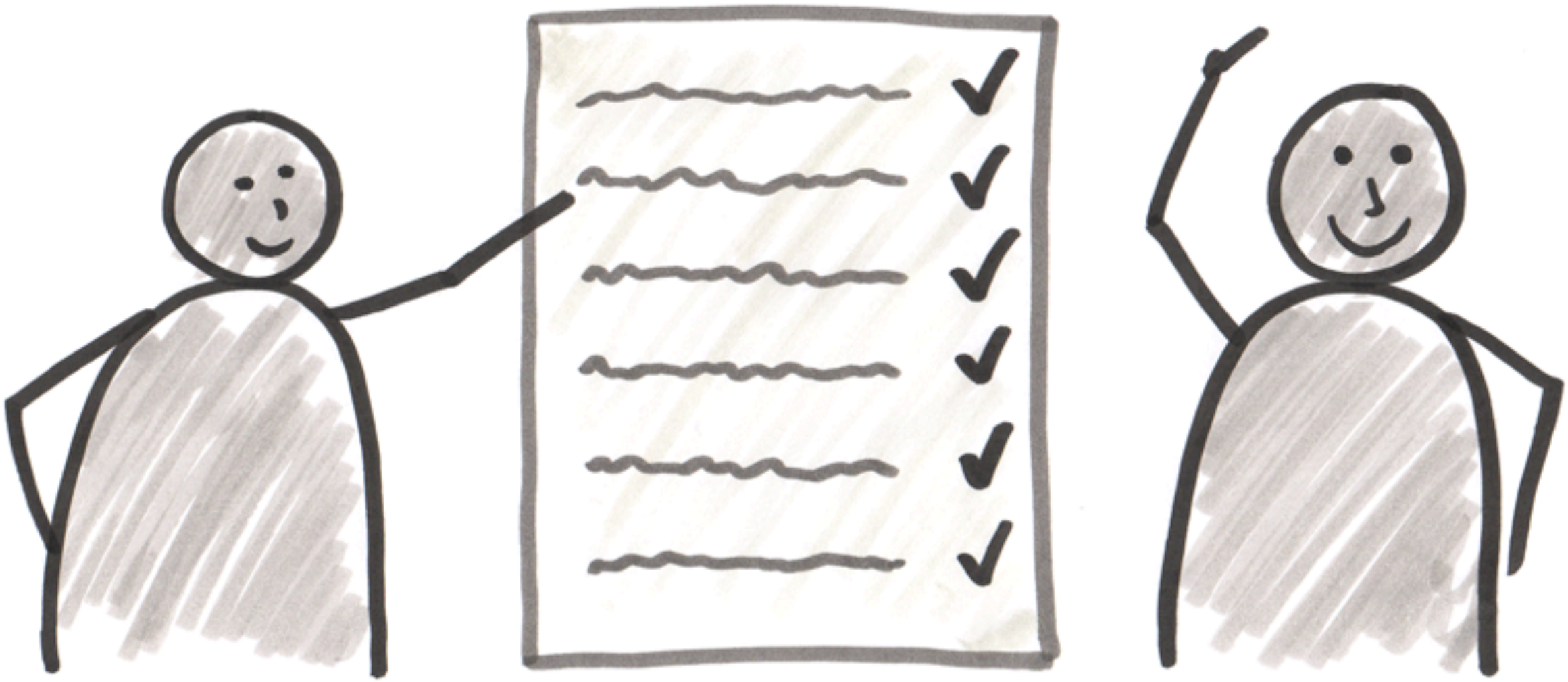
PRIORITY INVERSION

- In the Martian spacecraft, various devices communicated over a data bus.
- Activity on this bus was managed by a pair of high-priority tasking threads BM1 and BM2. One of the bus manager tasks that BM1 communicated through a pipe was a low-priority meteorological science task, MS1. The communication pipe was protected by a mutex L1. BM1 and MS1 needed to acquire the mutex L1 before they could send data over the pipe.

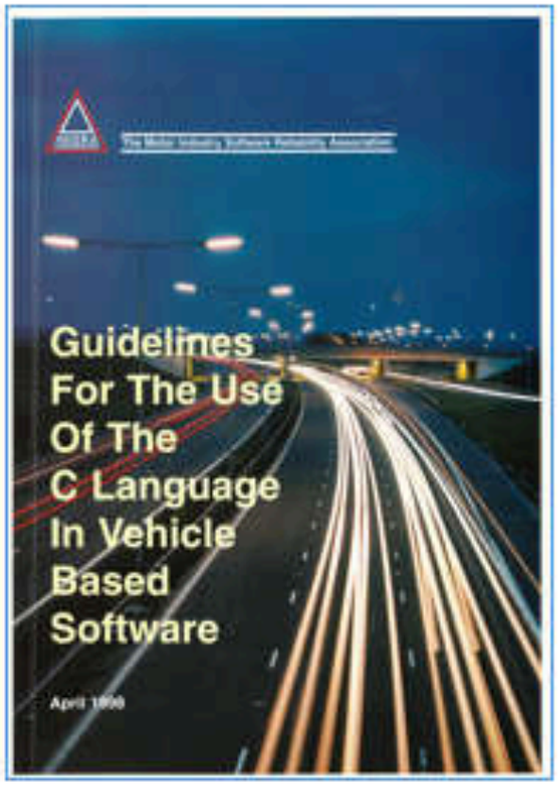
PRIORITY INVERSION

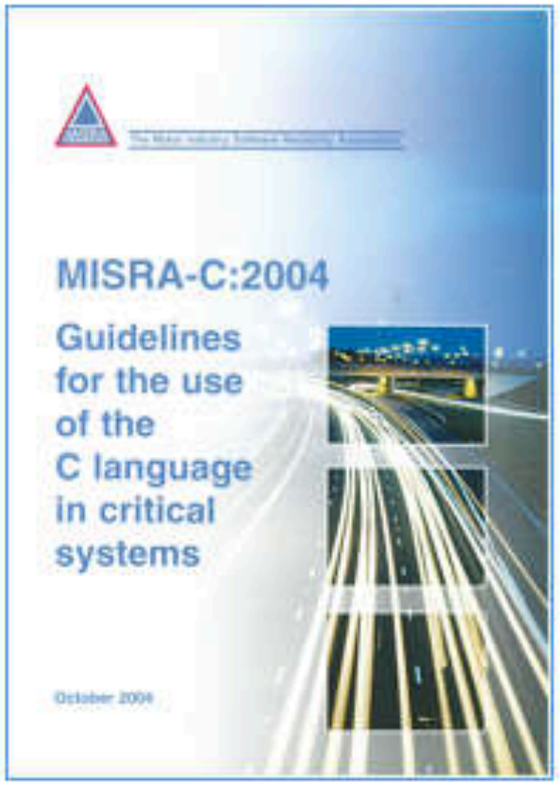
- The sequence of events leading to each reset began when the low-priority task MS1 was preempted by a couple of medium-priority tasks while it held the mutex L1.
- While the low-priority task MS1 was preempted, the high-priority bus distribution manager thread, BM1, tried to send more data to it over the same pipe. Because the mutex L1 was still held by MS1, the bus distribution manager was made to wait.
- Shortly thereafter, the other bus scheduler thread BM2 became active. It noticed that the distribution manager BM1 had not completed its work for that bus cycle and forced a system reset.
- VxWorks was the RTOS used on PathFinder. Fortunately, there was a priority inversion workaround already there in VxWorks. Using that, NASA engineers were able to solve the issue remotely.

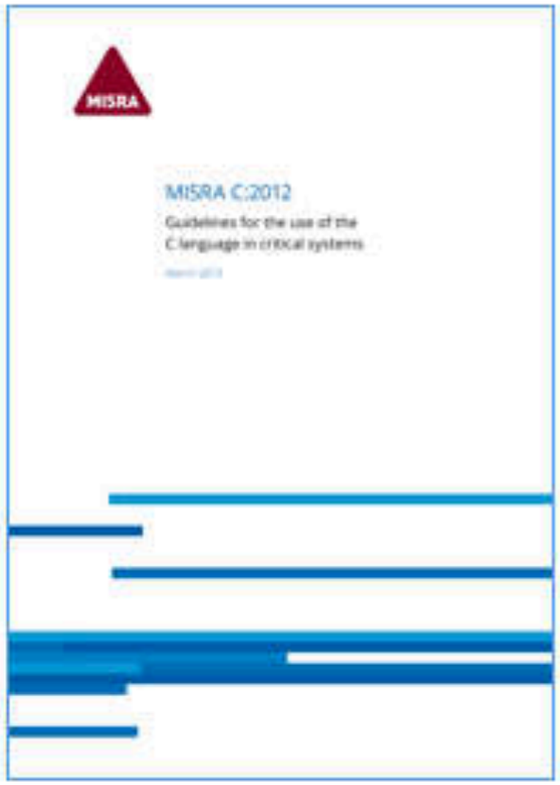
CODING GUIDELINES

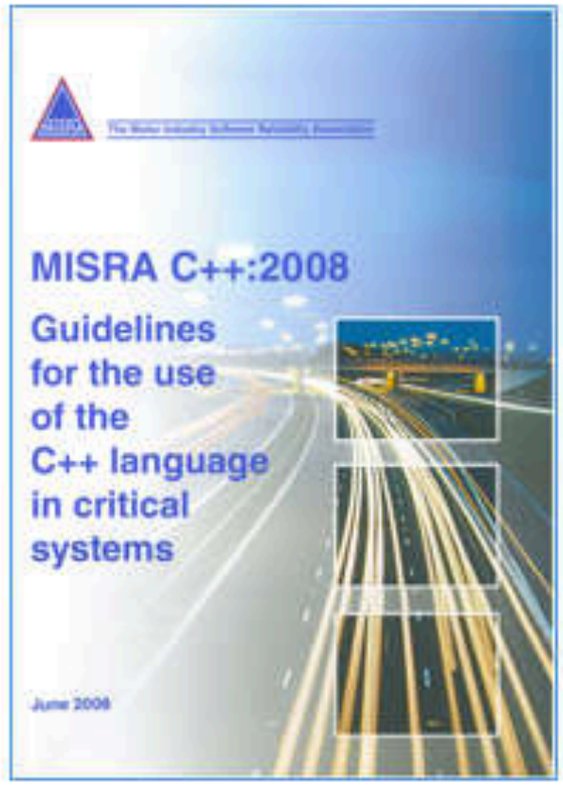


MISRA C/C++



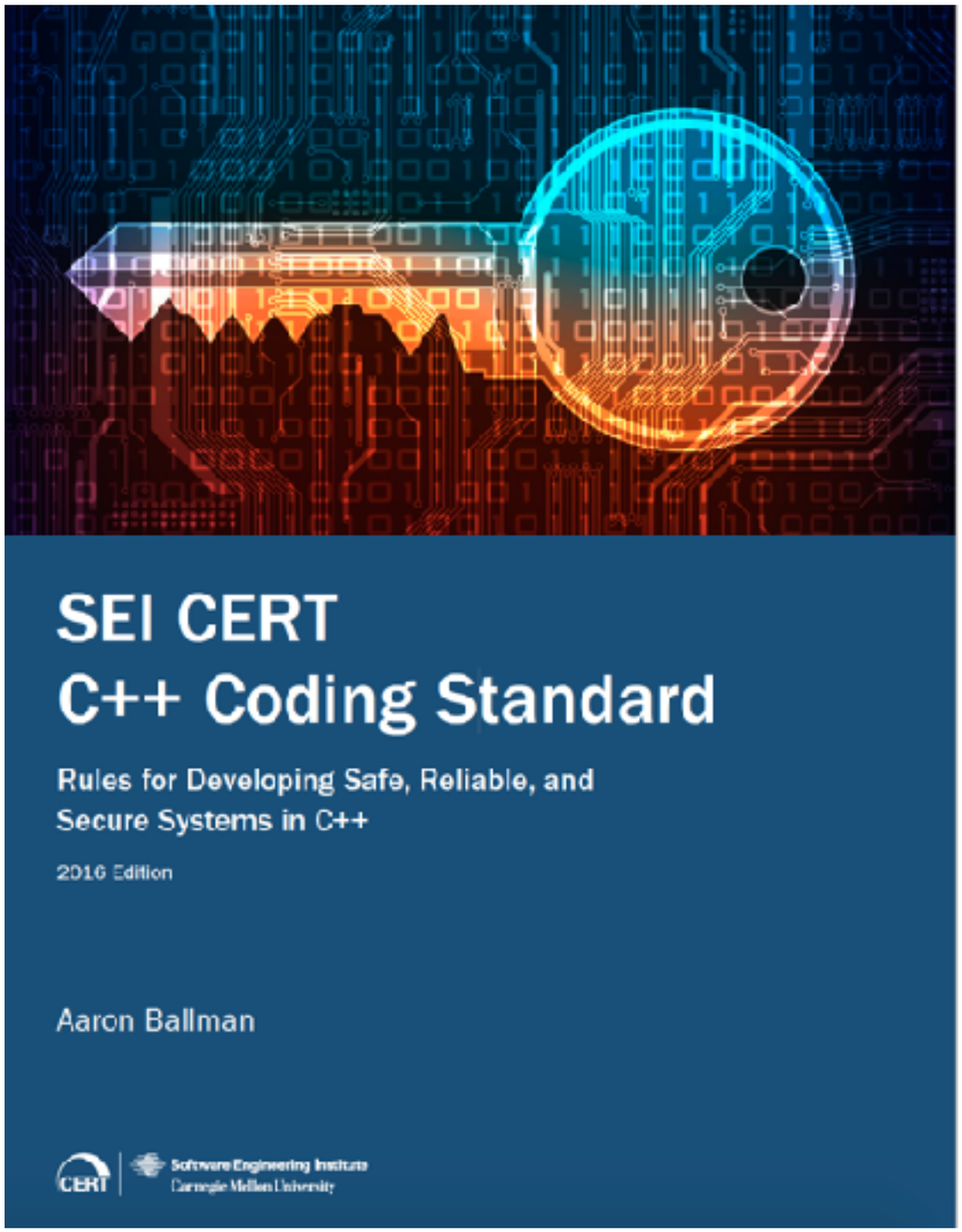






https://www.misra-cpp.com/MISRACHome/tabid/128/Default.aspx

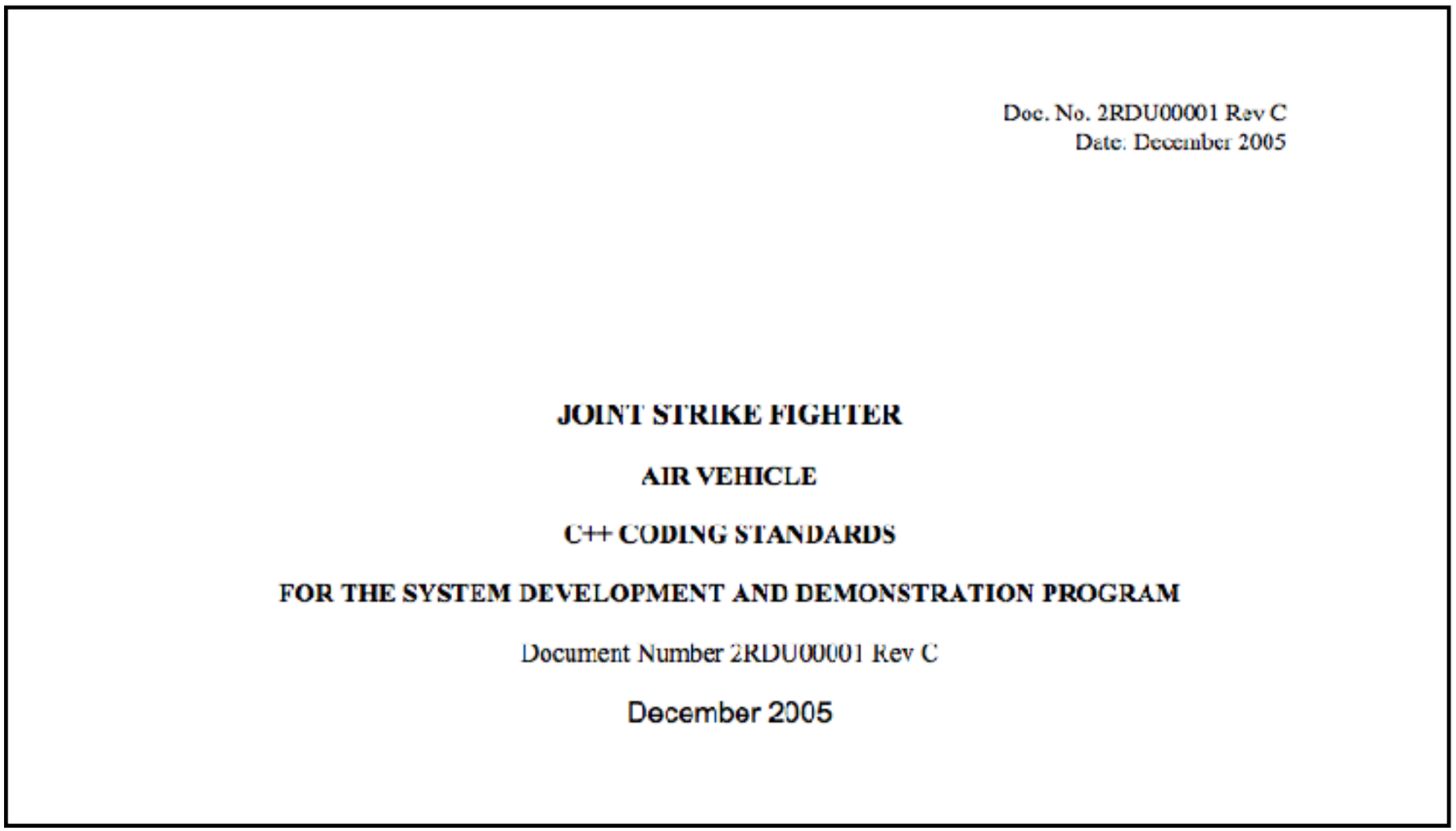
CERT C++



https://www.cert.org/downloads/secure-coding/assets/sei-cert-cpp-coding-standard-2016-v01.pdf

JSF++

Designed by Lockheed Martin for Joint Strike Fighter - Air Vehicle coding standard for C++



Doc. No. 2RDU00001 Rev C
Date: December 2005

JOINT STRIKE FIGHTER

AIR VEHICLE

C++ CODING STANDARDS

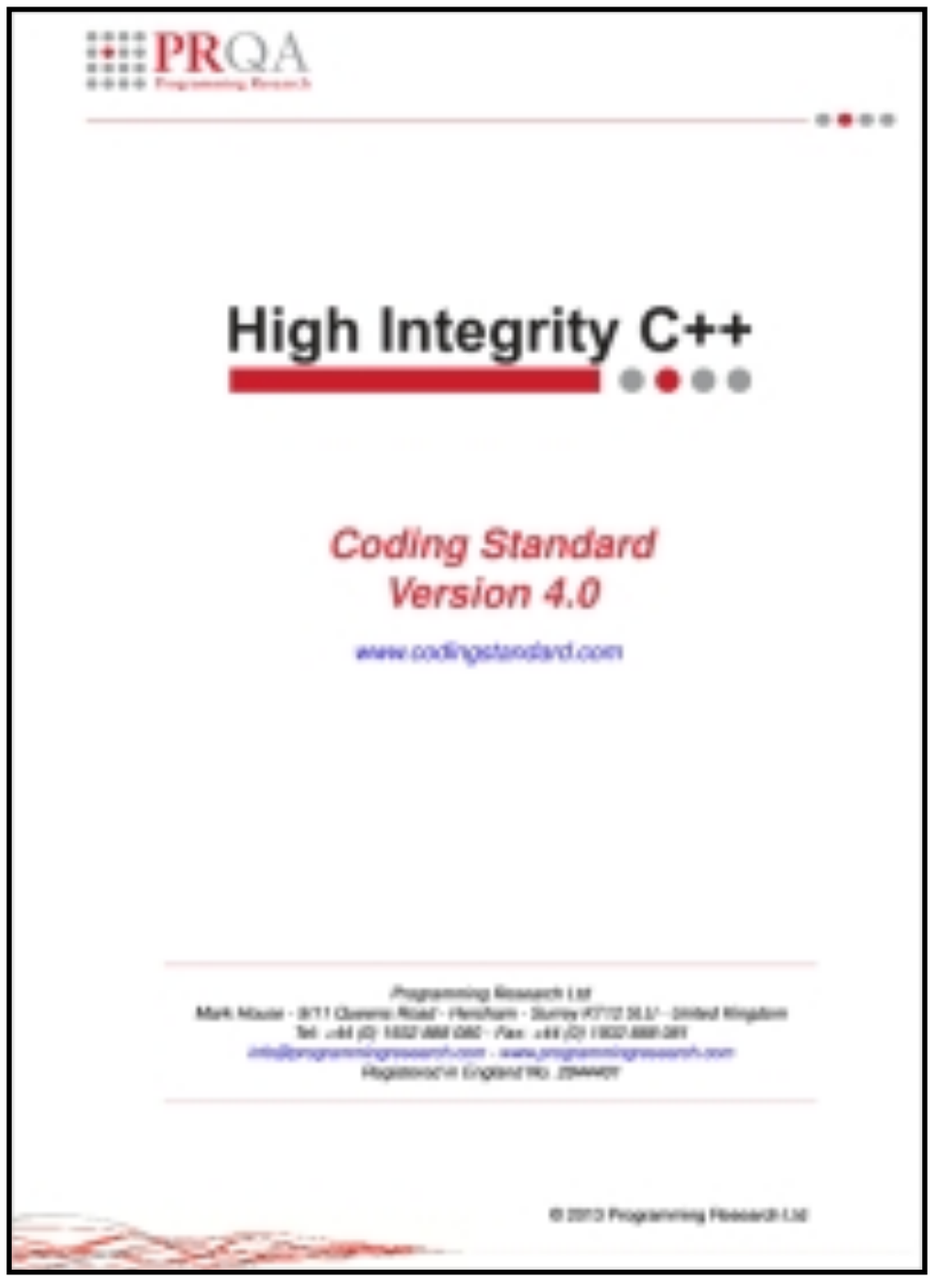
FOR THE SYSTEM DEVELOPMENT AND DEMONSTRATION PROGRAM

Document Number 2RDU00001 Rev C

December 2005

http://www.stroustrup.com/JSF-AV-rules.pdf

HIGH-INTEGRITY C++ CODING STANDARD



http://www.codingstandard.com/section/index/

GOOGLE C++ STYLE GUIDE

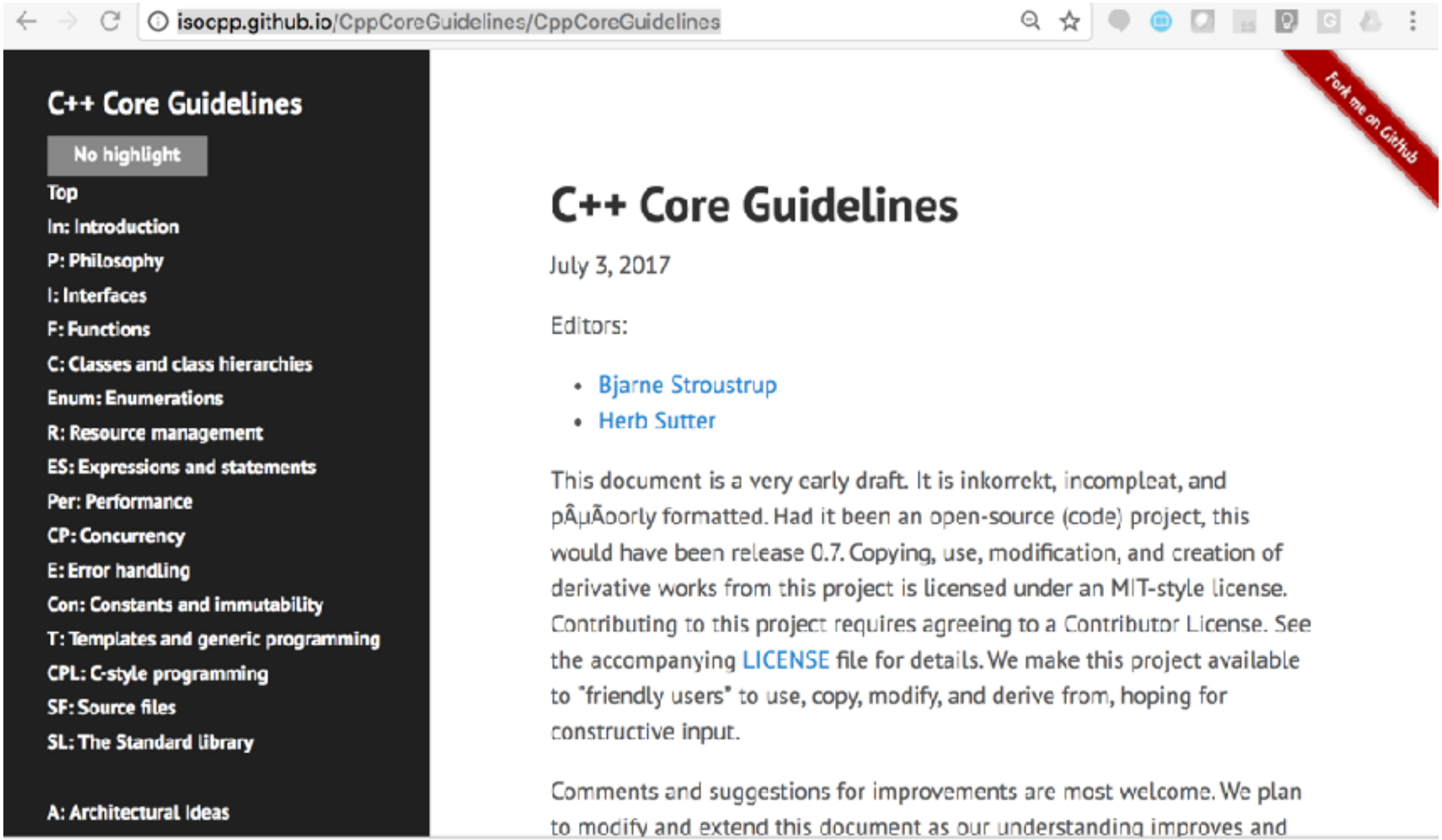
Google C++ Style Guide

https://google.github.io/styleguide/cppguide.html

Table of Contents

https://google.github.io/styleguide/cppguide.html

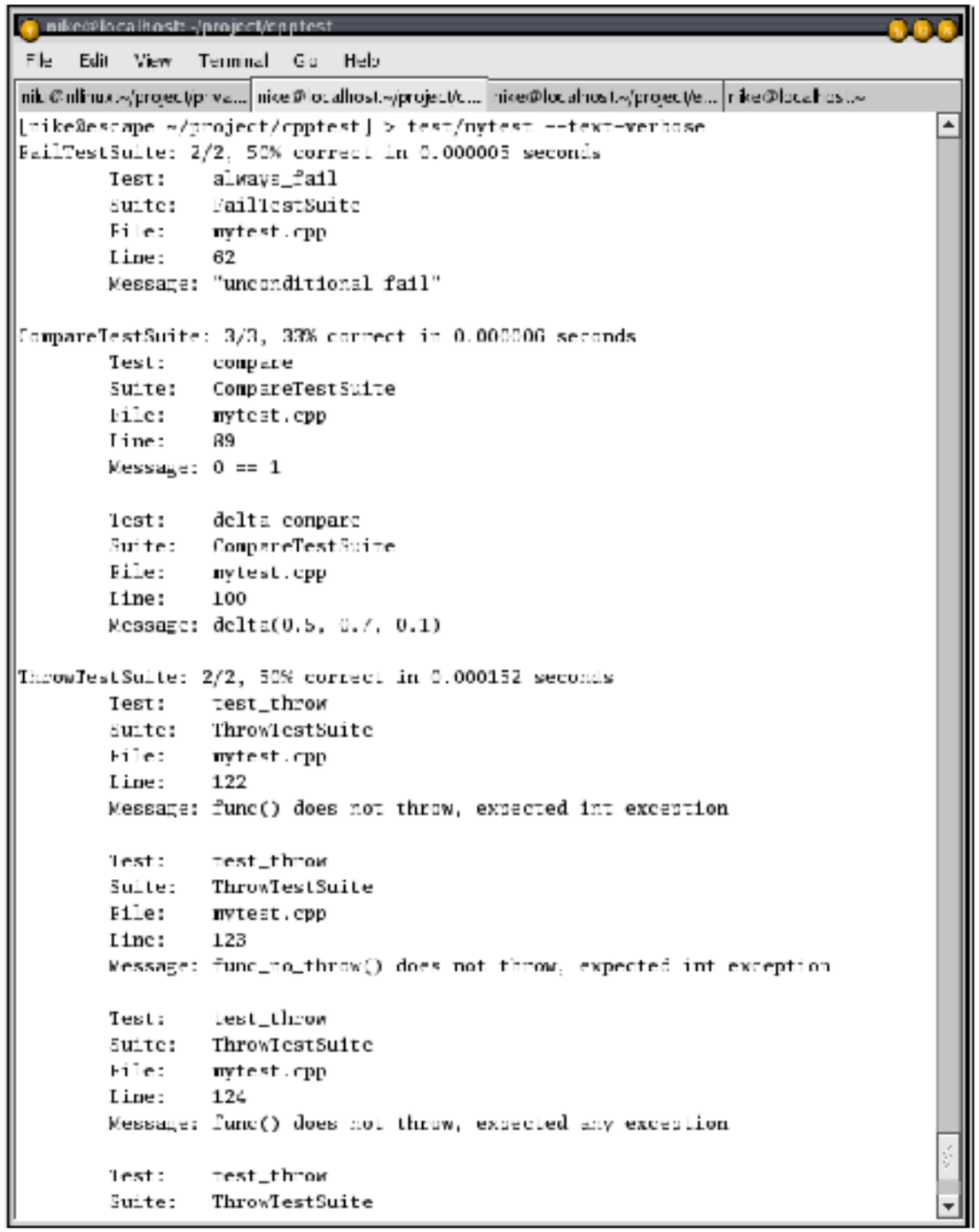
C++ CORE GUIDELINES



http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines

TOOLS TO USE

CPPTEST



http://cpptest.sourceforge.net/

SIMIAN

```
$ java -jar simian-2.3.34.jar -language=cpp /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/*.cpp
Similarity Analyser 2.3.34 - http://www.harukizaemon.com/simian
Copyright (c) 2003-2013 Simon Harris.  All rights reserved.
Simian is not free unless used solely for non-commercial or evaluation purposes.
{failOnDuplication=true, ignoreCharacterCase=true, ignoreCurlyBraces=true, ignoreIdentifierCase=true, ignoreModifiers=true,
ignoreStringCase=true, language=C++, threshold=6}
Found 6 duplicate lines in the following files:
 Between lines 352 and 360 in /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_networkstatistics.cpp
 Between lines 252 and 260 in /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_networkstatistics.cpp
Found 6 duplicate lines in the following files:
 Between lines 2519 and 2529 in /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_networking.cpp
 Between lines 2369 and 2379 in /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_networking.cpp
Found 6 duplicate lines in the following files:
 Between lines 1156 and 1167 in /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_networking.cpp
 Between lines 1134 and 1145 in /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_networking.cpp
Found 6 duplicate lines in the following files:
 Between lines 888 and 897 in /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_io.cpp
 Between lines 255 and 264 in /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_console.cpp
```

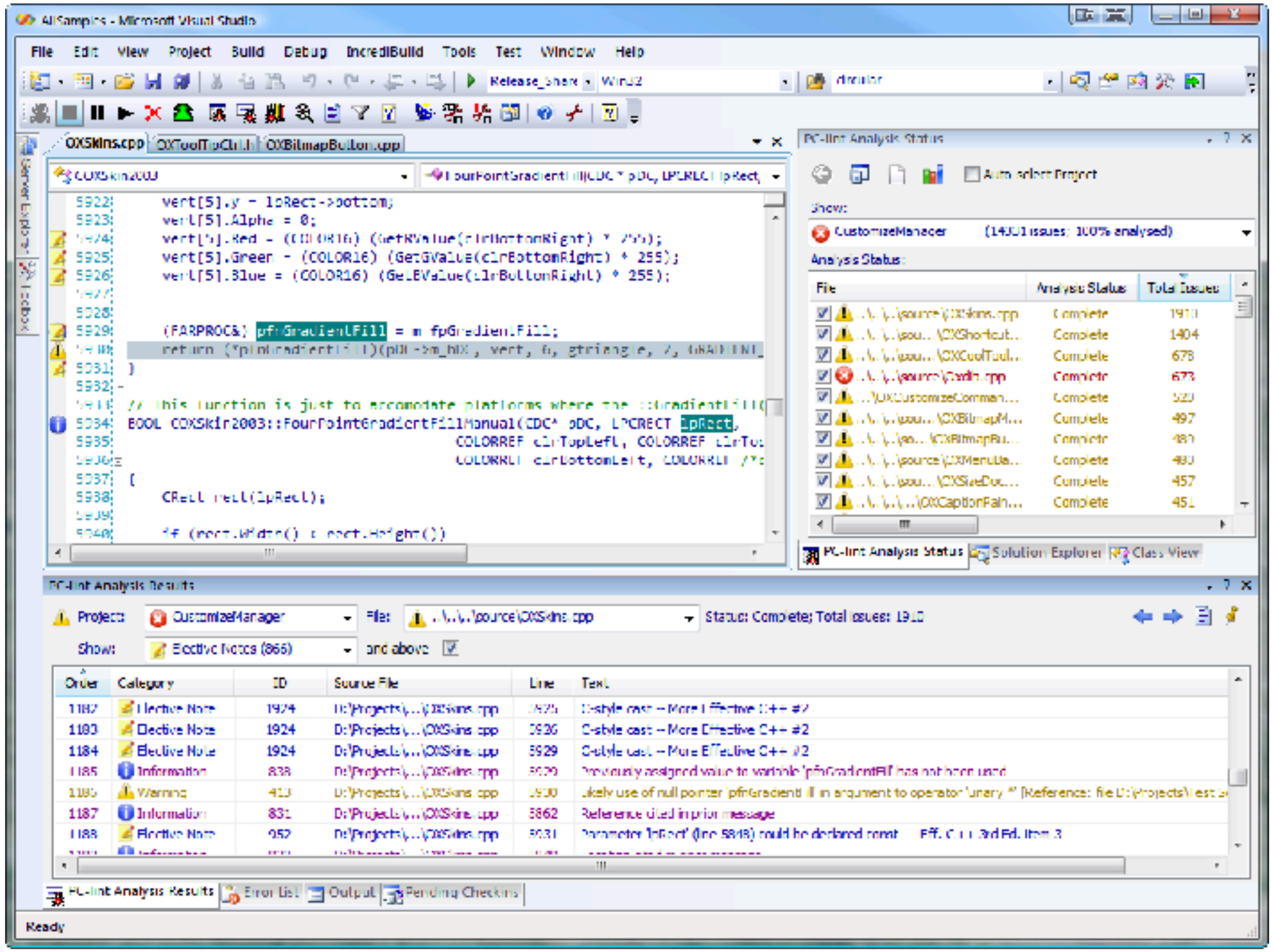
http://www.harukizaemon.com/simian/

PMD CPD (COPY PASTE DETECTOR)

```
$ ./cpd.sh /Users/gsamarthyam/corefx/src/Native/Unix/System.Native
Found a 22 line (117 tokens) duplication in the following files:
Starting at line 262 of /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_networkstatistics.cpp
Starting at line 362 of /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_networkstatistics.cpp

    while (sysctlbyname("net.inet.tcp.pcblist", buffer, &estimatedSize, newp, newlen) != 0)
    {
        delete[] buffer;
        size_t tmpEstimatedSize;
        if (!multiply_s(estimatedSize, static_cast<size_t>(2), &tmpEstimatedSize) ||
            (buffer = new (std::nothrow) uint8_t[estimatedSize]) == nullptr)
        {
            errno = ENOMEM;
            return -1;
        }
        estimatedSize = tmpEstimatedSize;
    }

    int32_t count = static_cast<int32_t>(estimatedSize / sizeof(xtcpcb));
    if (count > *infoCount)
    {
        // Not enough space in caller-supplied buffer.
        delete[] buffer;
        *infoCount = count;
        return -1;
    }
    *infoCount = count;
=====================================================================
Found a 20 line (105 tokens) duplication in the following files:
Starting at line 546 of /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_networking.cpp
Starting at line 625 of /Users/gsamarthyam/corefx/src/Native/Unix/System.Native/pal_networking.cpp

    assert(hostname != nullptr);
    assert(entry != nullptr);

    size_t scratchLen = 512;

    for (;;)
    {
        size_t bufferSize;
        uint8_t* buffer;
        if (!add_s(sizeof(hostent), scratchLen, &bufferSize) ||
            (buffer = reinterpret_cast<uint8_t*>(malloc(bufferSize))) == nullptr)
        {
            return PAL_NO_MEM;
        }

        hostent* result = reinterpret_cast<hostent*>(buffer);
        char* scratch = reinterpret_cast<char*>(&buffer[sizeof(hostent)]);

        int getHostErrno;
        int err = gethostbyname_r(reinterpret_cast<const char*>(hostname), result, scratch, scratchLen, entry, &getHostErrno);
```

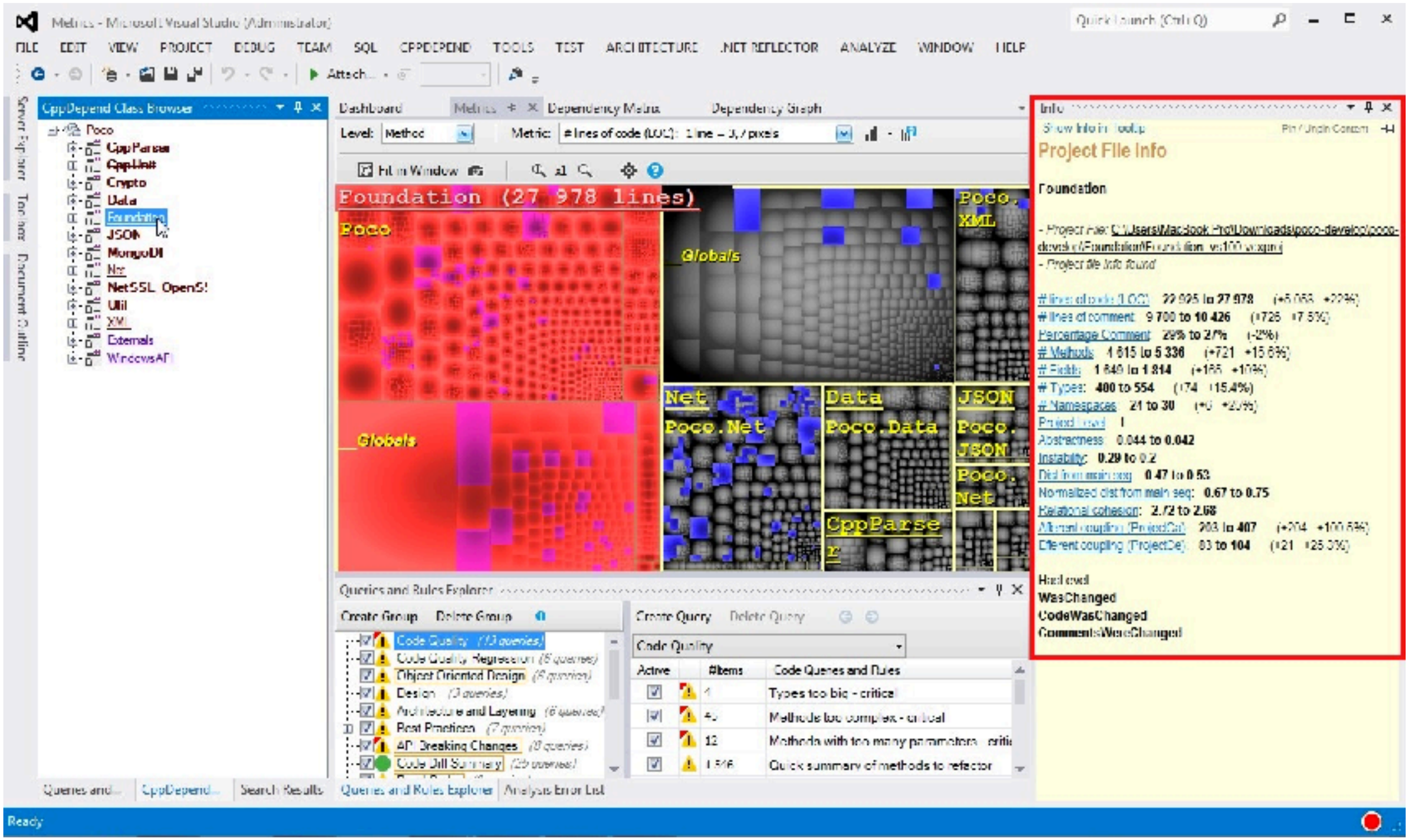
pmd.sourceforge.net/pmd-4.3.0/cpd.html

PC-LINT



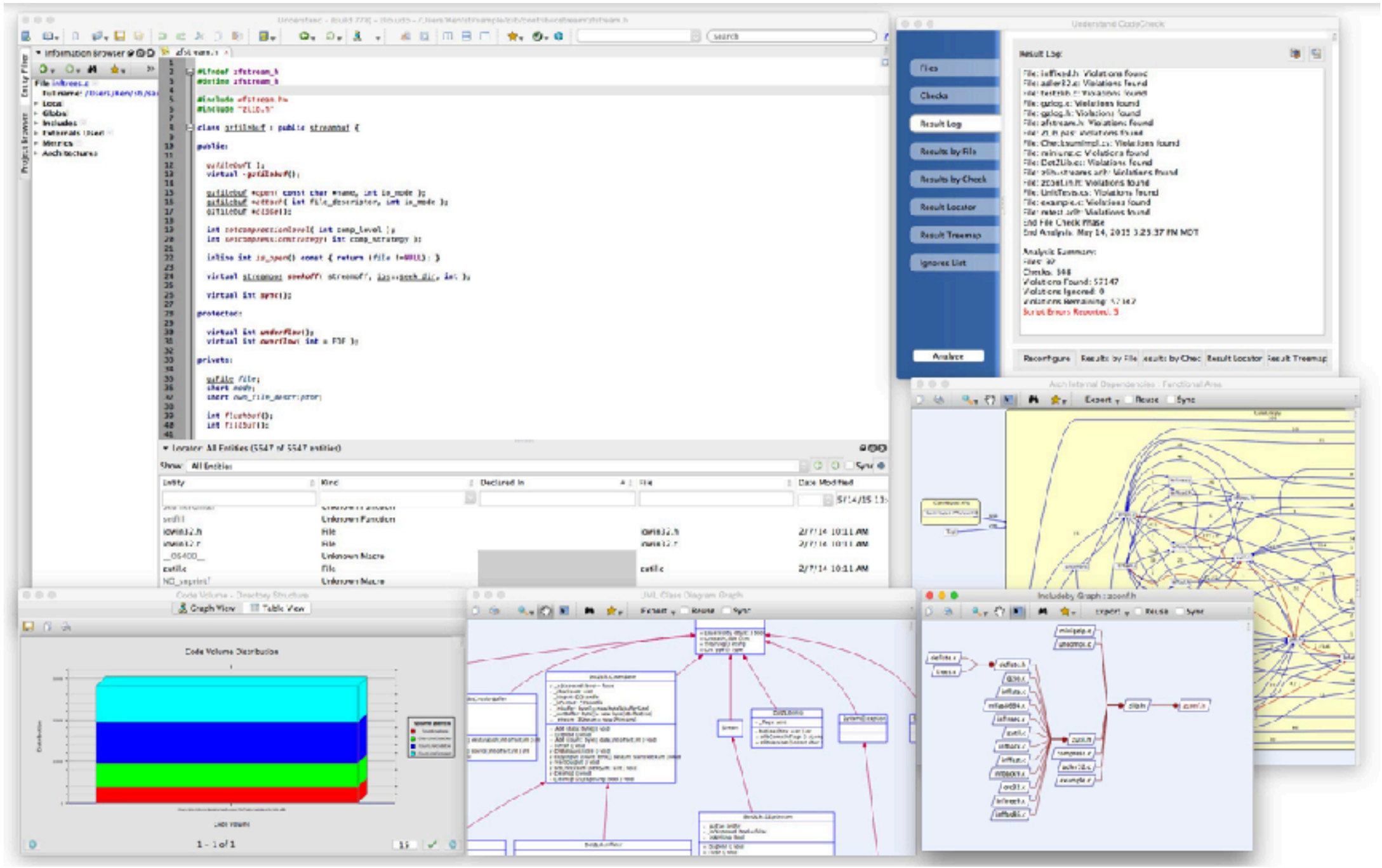
http://www.gimpel.com/html/pcl.htm

CPPDEPEND



http://www.cppdepend.com/

UNDERSTAND FOR C++



https://scitools.com/

BUT BE CAREFUL WHEN TOOLS REPORT ERRORS

	revision 140, Tue May 2 16:25:19 2006 UTC	revision 141, Tue May 2 16:34:53 2006 UTC
#	**Line 271**	**Line 271**
271	else	else
272	MD_Update(&m,&(state[st_idx]),j);	MD_Update(&m,&(state[st_idx]),j);
273		
274		/*
275		* Don't add uninitialised data.
276	MD_Update(&m,buf,j);	MD_Update(&m,buf,j);
277		*/
278	MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));	MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
279	MD_Final(&m,local_md);	MD_Final(&m,local_md);
280	md_c[1]++;	md_c[1]++;
#	**Line 465**	**Line 468**
468	MD_Update(&m,local_md,MD_DIGEST_LENGTH);	MD_Update(&m,local_md,MD_DIGEST_LENGTH);
469	MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));	MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
470	#ifndef PURIFY	#ifndef PURIFY
471		/*
472		* Don't add uninitialised data.
473	MD_Update(&m,buf,j); /* purify complains */	MD_Update(&m,buf,j); /* purify complains */
474		*/
475	#endif	#endif
476	k=(st_idx+MD_DIGEST_LENGTH/2)-st_num;	k=(st_idx+MD_DIGEST_LENGTH/2)-st_num;
477	if (k > 0)	if (k > 0)

OpenSSL change that caused security vulnerability in Debian

BOOKS TO READ

Effective C++
Third Edition

55 Specific Ways to Improve Your Programs and Designs

Scott Meyers

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

More Effective C++

35 New Ways to Improve Your Programs and Designs

Scott Meyers

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Effective STL

50 Specific Ways to Improve Your Use of the Standard Template Library

Scott Meyers

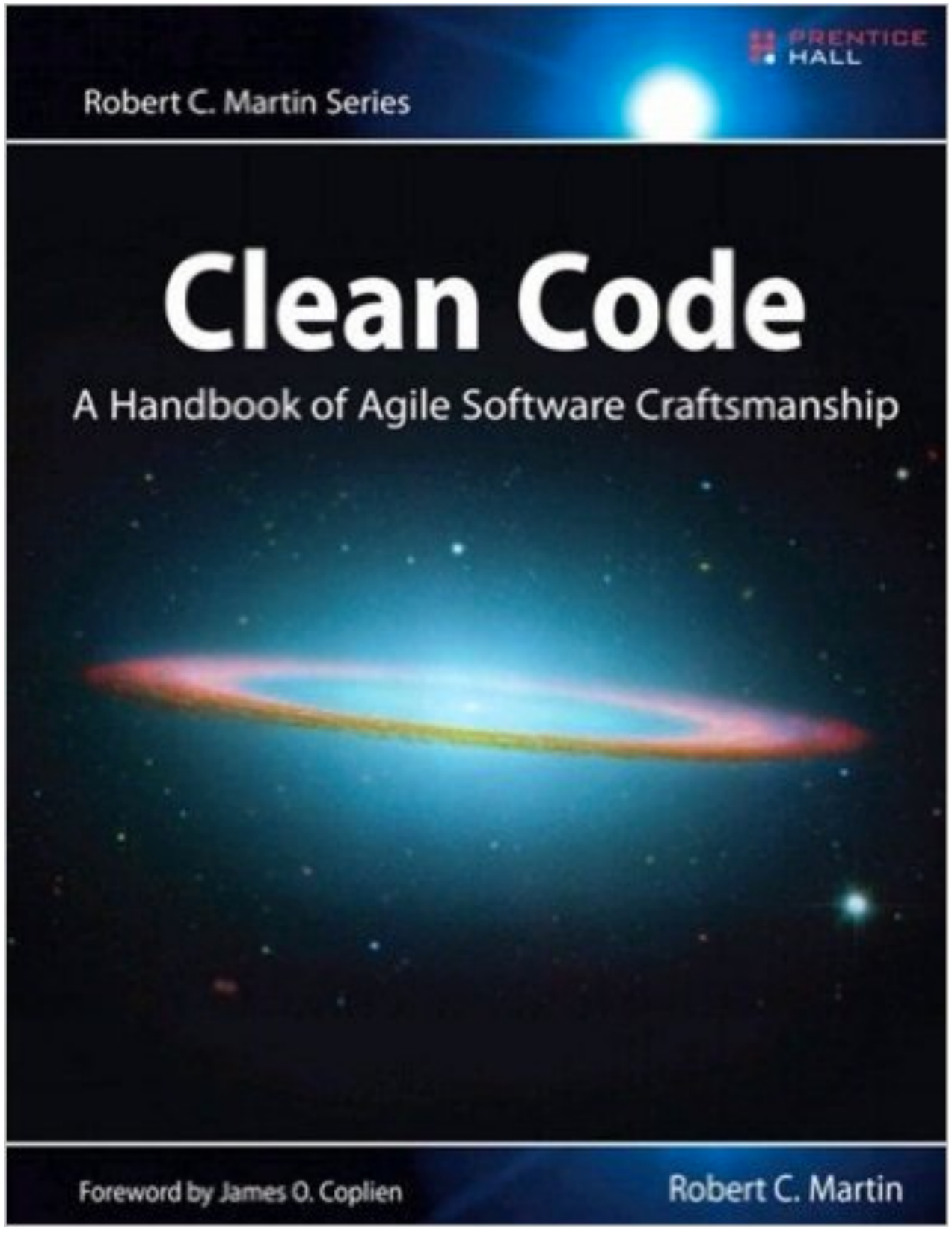
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

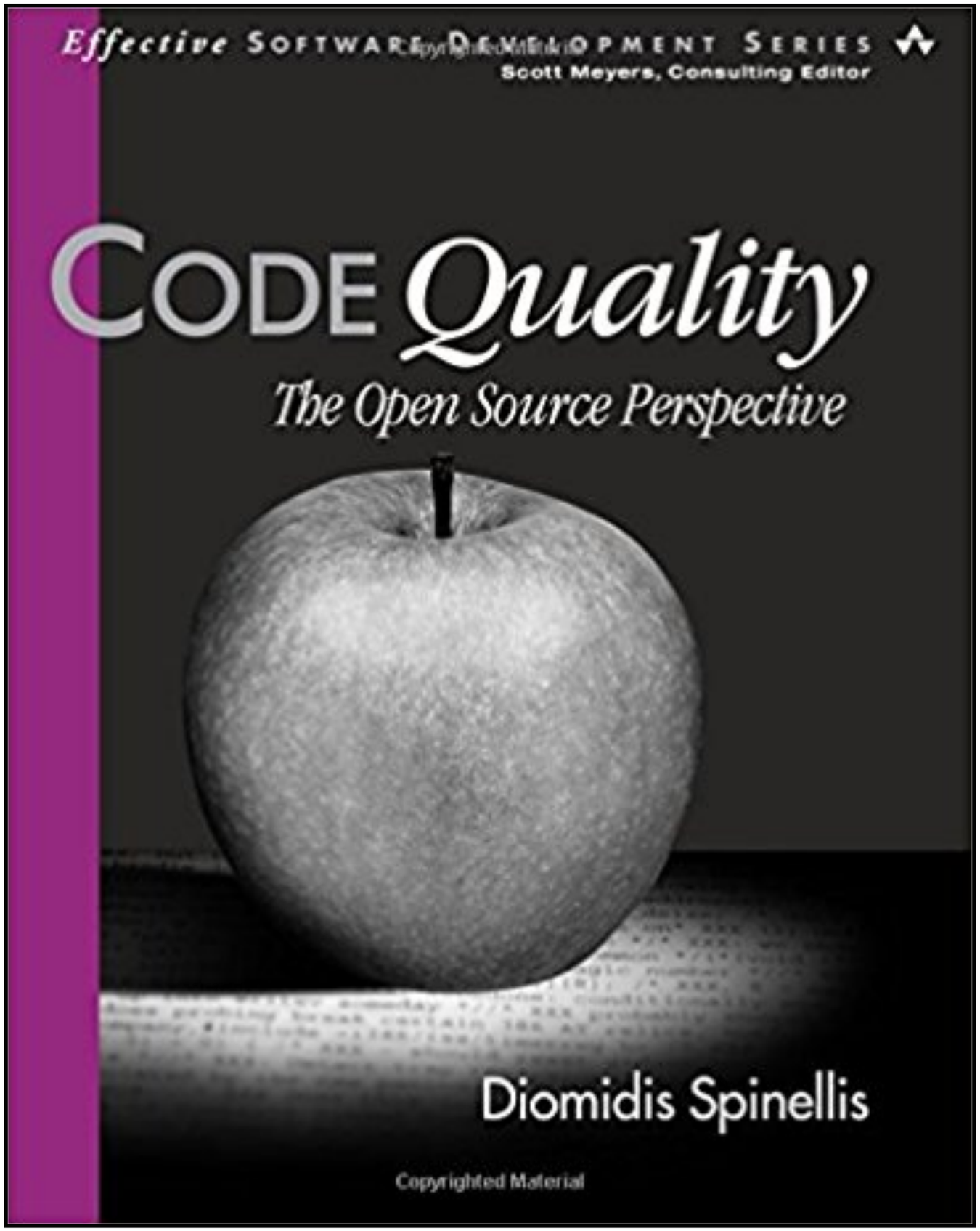
C++ Coding Standards

101 Rules, Guidelines, and Best Practices

Herb Sutter
Andrei Alexandrescu

C++ In-Depth Series • Bjarne Stroustrup


PRENTICE HALL
Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin


Effective Software Development Series
Scott Meyers, Consulting Editor
Code Quality
The Open Source Perspective
Diomidis Spinellis
Copyrighted Material

WRITING SOLID CODE

Microsoft's Techniques for Developing Bug-Free C Programs

STEVE MAGUIRE

Foreword by Dave Moore
Director of Development, Microsoft Corporation

Microsoft Press

Microsoft

CODE COMPLETE

2

Second Edition

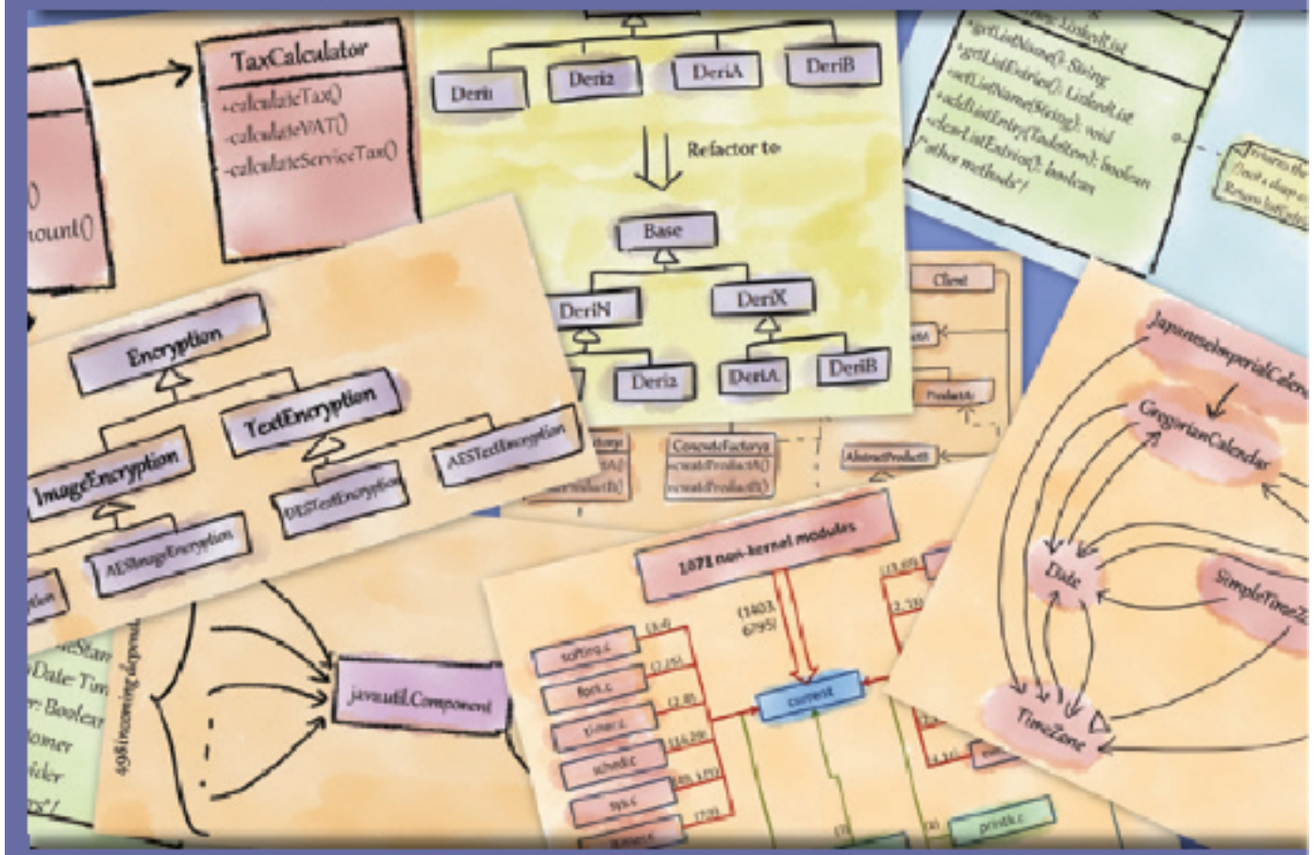
A practical handbook of software construction

Steve McConnell

Two-time winner of the *Software Development* Magazine Jolt Award

Refactoring for Software Design Smells

Managing Technical Debt



Girish Suryanarayana,
Ganesh Samarthyam, Tushar Sharma

Forewords by Grady Booch and Stéphane Ducasse

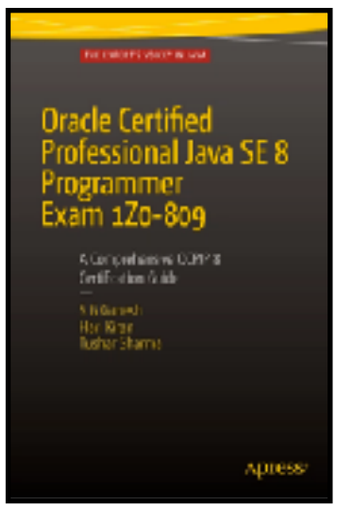
MK
MORGAN KAUFMANN

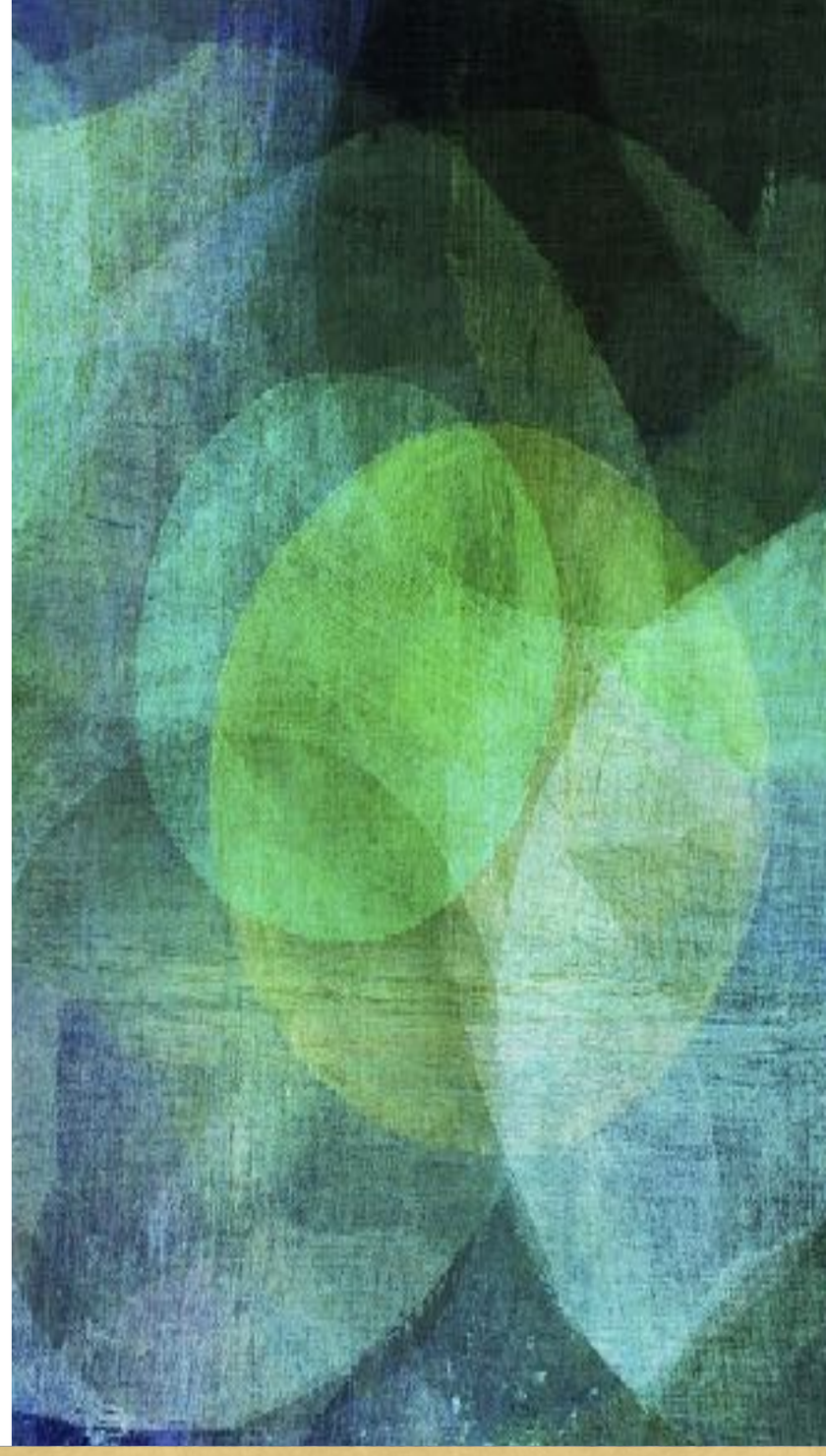
IMAGE CREDITS

- http://www.informit.com/ShowCover.aspx?isbn=0321334876
- https://www.pearsonhighered.com/assets/bigcovers/0/1/3/2/013279747X.jpg
- http://images.pearsoned-ema.com/jpeg/large/9780201749625.jpg
- https://images-na.ssl-images-amazon.com/images/I/41TINACY3hL._SX384_BO1,204,203,200_.jpg
- https://images-na.ssl-images-amazon.com/images/I/51esm%2BZYfDL._SX395_BO1,204,203,200_.jpg
- https://images-na.ssl-images-amazon.com/images/I/719IV-Xyr1L.jpg
- https://images-na.ssl-images-amazon.com/images/I/51MEZDyKvZL._SX409_BO1,204,203,200_.jpg
- https://pixabay.com/en/drill-milling-milling-machine-444499/
- http://l1.yimg.com/bt/api/res/1.2/SDyAe8VtF3AF9pu7KB3Klw--/YXBwaWQ9eW5ld3M7cT04NTt3PTY1MA--/http://globalfinance.zenfs.com/en_us/Finance/US_AFTP_SILICONALLEY_H_LIVE/Amazing_images_show_what_looks-28cee28fe2273975438300e102d050c
- https://scitools.com/wp-content/uploads/2015/05/Understand-knows-a-lot.jpg
- http://www.riverblade.co.uk/images/blog/2011/cppcheck_analysis_example_pclint_comparison.png
- https://cdn-images-1.medium.com/max/1600/1*J2mKSLBEp_jUbMtOWXTTjQ.png
- https://wlion.com/wp-content/uploads/2017/04/CleanCode.jpg

IMAGE CREDITS

- http://allsquash.co.uk/wp-content/uploads/2015/11/Warm-Up-Cartoon.gif
- https://pbs.twimg.com/profile_images/514432418330075136/ie5DoP1U.png
- http://geekandpoke.typepad.com/.a/6a00d8341d3df553ef0120a6d65b8a970b-pi
- https://static1.squarespace.com/static/518f5d62e4b075248d6a3f90/t/5868efc159cc6829cf3677be/1483272141992/
- http://cpptest.sourceforge.net/screenshot-text-verbose.png
- https://s-media-cache-ak0.pinimg.com/originals/d9/39/14/d93914819b10bc9c21508c0145f439c3.jpg
- https://www.socialpsychology.org/thumb/4111/0/md.jpg
- https://i1.wp.com/www.languageties.com/sites/default/files/images/lexical/003-When-in-Rome-do-as-the-Romans-do.jpg
- https://www.qa-systems.de/fileadmin/Tools/Static_analysis/QA-MISRA/MISRA_Covers.PNG
- http://343f11.pbworks.com/f/1317750936/memory.gif
- http://ecx.images-amazon.com/images/I/81bVWBzcLJL.jpg





ganesh@codeops.tech

www.codeops.tech

+91 98801 64463

@GSamarthyam

slideshare.net/sgganesh

bit.ly/sgganesh

CodeOps