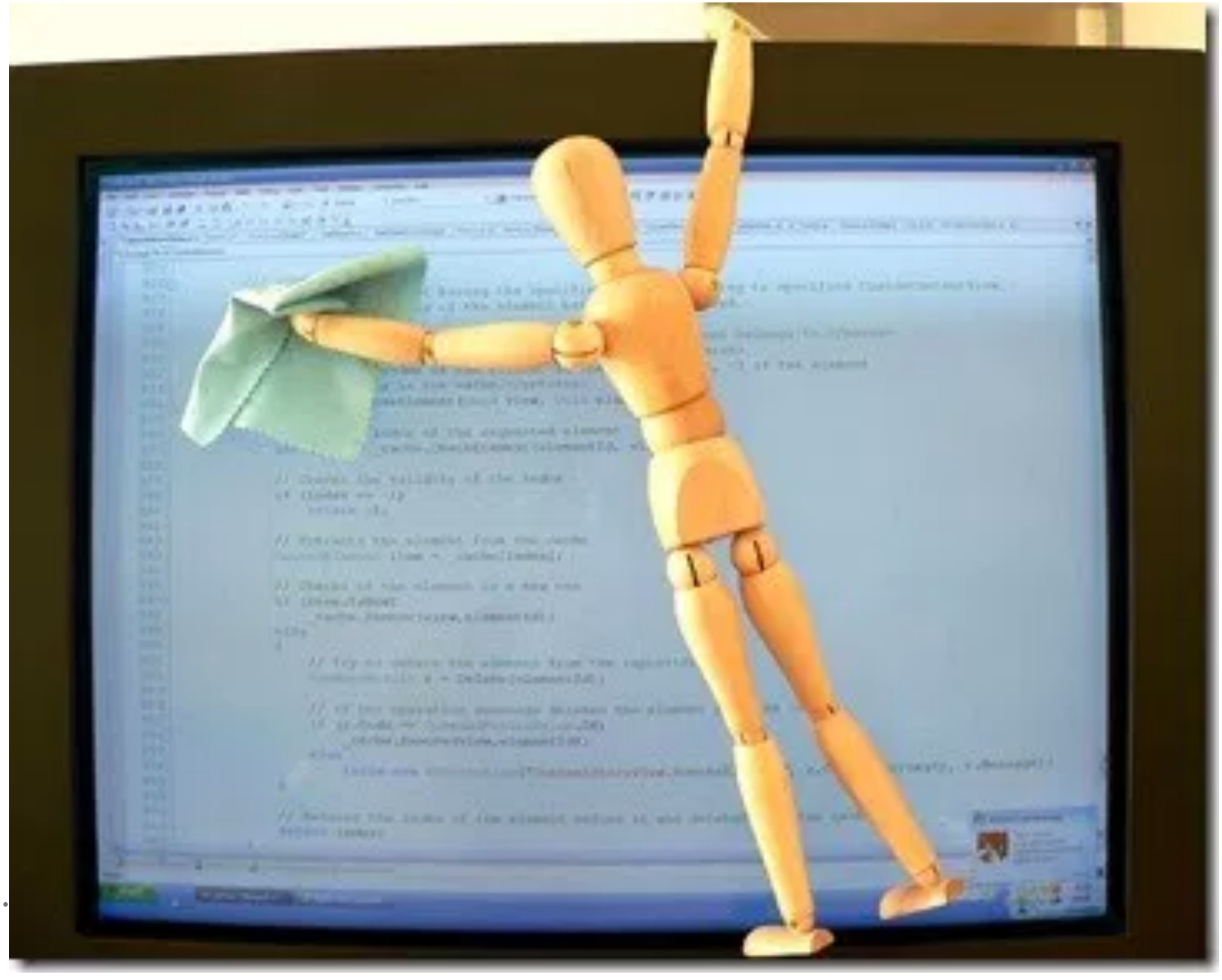


Ganesh Samarthyam
ganesh@codeops.tech

“

Keeping “clean” is a habit - it’s not to do with “skills” or “ability”!

WARM-UP EXERCISE

ASSUME MANY PLACES YOU HAVE SUCH CODE- HOW TO IMPROVE?

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

bool validate_temperature_measures(vector<int> &measures) {
        return find_if(measures.begin(),
                        measures.end(),
                       [](const int value) { return (value < 0) || (value > 100); })
                == measures.end();
        for(vector<int>::iterator i = measures.begin(); i != measures.end(); ++i ) {
                if( *i < 0 || *i > 100) {
                        return false;
                }
        }
        return true;
}

int main() {
        vector<int> measures = { 23, 66, 25, 85, 110, 45, 34, 90 };
        cout << "valid measures? " << std::boolalpha <<
validate_temperature_measures(measures) << endl;
}
```

ATTEMPT #1

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

template<typename T>
class is_outside_range : public unary_function<T, bool> {
public:
  is_outside_range(const T& low, const T& high) : low_(low), high_(high) {}
  bool operator()( const T& val ) const {return val < low_ || val > high_; }
private:
  T low_, high_;
};

bool validate_temperature_measures(vector<int> &measures) {
   vector<int>::iterator result = find_if(measures.begin(), measures.end(),
is_outside_range<int>(0, 100));
   return result == measures.end();
}

int main() {
   vector<int> measures = { 23, 66, 25, 85, 10, 45, 34, 90 };
   cout << "valid measures? " << std::boolalpha <<
validate_temperature_measures(measures) << endl;
}
```

ATTEMPT #2

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

bool validate_temperature_measures(vector<int> &measures) {
   function<bool(int)> temperature_validator =
         [](const int value) {
            const int low = 0;
            const int high = 100;
            return (value < low) || (value > high);
         };
   vector<int>::iterator result = find_if(measures.begin(), measures.end(),
temperature_validator);
   return result == measures.end();
}

int main() {
   vector<int> measures = { 23, 66, 25, 85, 10, 45, 34, 90 };
   cout << "valid measures? " << std::boolalpha <<
validate_temperature_measures(measures) << endl;
}
```

GETTING STARTED

ESSENTIALS

- Ensure traceability within code
- Do not write “sloppy code”
- Provide necessary copyright notice, change logs, etc. in each source and header files

FORMATTING

- “Consistency” is key to formatting
  - Example: placement of curly braces - “{“ and “}”
  - Another obvious example: tabs vs. whitespaces
- Keep lines within 80 characters
- Ensure “density” is not too high
  - E.g., leave out enough a linespace between group of statements
- Have an ordering of member functions, data members, etc.
- Ensure proper intendation

MEANINGFUL NAMES

- Use intention-revealing names

```
int d; // elapsed time in days
int elapsedTimeInDays;
```

- Use searchable names

```
for(int j = 0; j < 34; j++)
   s+= (t(j)*4)/5;
```

```
int realDaysPerIdealDay = 4;
const int WORK_DAYS_PER_WEEK = 5;
int sum = 0;
for(int task; task < NUMBER_OF_TASKS; j++) {
   int realTaskDays = taskEstimate[task] * realDaysPerIdealDay;
   int realTaskWeeks = (realDays / WORK_DAYS_PER_WEEK);
   sum += realTaskWeeks;
}
```

Clean Code: A Handbook of Agile Software Craftsmanship, Robert C. Martin, Prentice Hall, 2008

NAMING

- Avoid Hungarian notation and member prefixes
- Class names should have noun or noun phrases (and not be a verb)
- Method names should have verb or verb phrase names
- Avoid typos and smelling mistakes in the code

COMMENTS

- Provide good “internal documentation”
- Comments do not make up for bad code
- Explain yourself in code
  - Explain intent
- Avoid commenting-out code
- Avoid redundant comments
- Provide mandated comments
- Use position markers sparingly (e.g., // Actions ////////)

Clean Code: A Handbook of Agile Software Craftsmanship, Robert C. Martin, Prentice Hall, 2008

FUNCTIONS / METHODS

- Avoid macros - use inline functions instead
- Method names - neither too long, nor too short
- Keep parameter list small (not more than 4 or so parameters)
- Avoid “out” parameters - prefer return values instead
- Avoid friend functions
- Prefer const methods
- Ensure proper return from all paths
  - Strive for exception safety

FUNCTIONS / METHODS

- Keep nesting level of functions low (not more than 3 or 4 levels deep)
- Keep Cyclomatic complexity low (less than 10 per method)
- Keep lines size small (more than 10 or so becomes difficult to grasp)
- Avoid functions doing many things - do one thing (e.g., realloc does too many things)
- Avoid “stateful-functions” (e.g., strtok)

CLASSES

- Prefer one class per file
- Limit the number of members in a class
- Avoid public data members
- Limit the accessibility of the class members
- Have a ordering for the methods and data members; order public, private, and protected members
- Keep the complexity of classes low - i.e., Weighted Methods per Class (WMC) keep it low like 25 or 30
- Avoid “struct-like” classes

HARDCODING

- Avoid “hard-coded logic”
- Avoid “magic numbers”
- Avoid “magic strings”

“

Adopt a coding standard and adhere to it (as a team!)

Code without unit tests is legacy code!

HOW TO REFACTOR?

HOW TO REFACTOR LARGE CLASSES?

Problem	How to Refactor	Refactoring Name
There is one class that does the work of two	move relevant fields and methods to a new class	Extract Class
A class has members that are only used by only few instances	create a subclass and move those members to that	Extract Subclass
A method in a class contains code for conditional execution logic (say switch)	create a Command for each action and replace the conditional execution logic with the execution of commands	Replace Conditional Dispatcher with Command
A method in class has complex conditional expressions controlling object's state	replace the conditions with State classes that would internally handle the states and transitions between them	Replace State-Altering Conditionals with State
There are numerous methods in a class that have elements of an implicit language	define classes for that implicit language so that instances can be combined to for interpretable expressions	Replace Implicit Language with Interpreter

HOW TO REFACTOR LONG METHODS?

Problem	How to Refactor	Refactoring Name
The method's logic is difficult to understand	transform the steps of the method to the same level so that each step expresses the intent	Compose Method
There is a independent logical block of statements present in the method	introduce a new method with descriptive name explaining the purpose of that method	Extract Method
There is a set of statements that accumulates information to a local variable	accumulate results to a Collecting Parameter that gets passed to extracted methods	Move Accumulation to Collecting Parameter
method accumulates information to a local variable from heterogeneous classes	move the accumulation task to a visitor that can visit each class and get that information	Move Accumulation to Visitor
method has conditional logic that does several variants of calculation	create a Strategy for each variant and use a Strategy instance to do the actual calculation	Replace Conditional Logic with Strategy
method contains code for conditional execution logic	create a Command for each action and replace the conditional execution logic with the execution of commands	Replace Conditional Dispatcher with Command

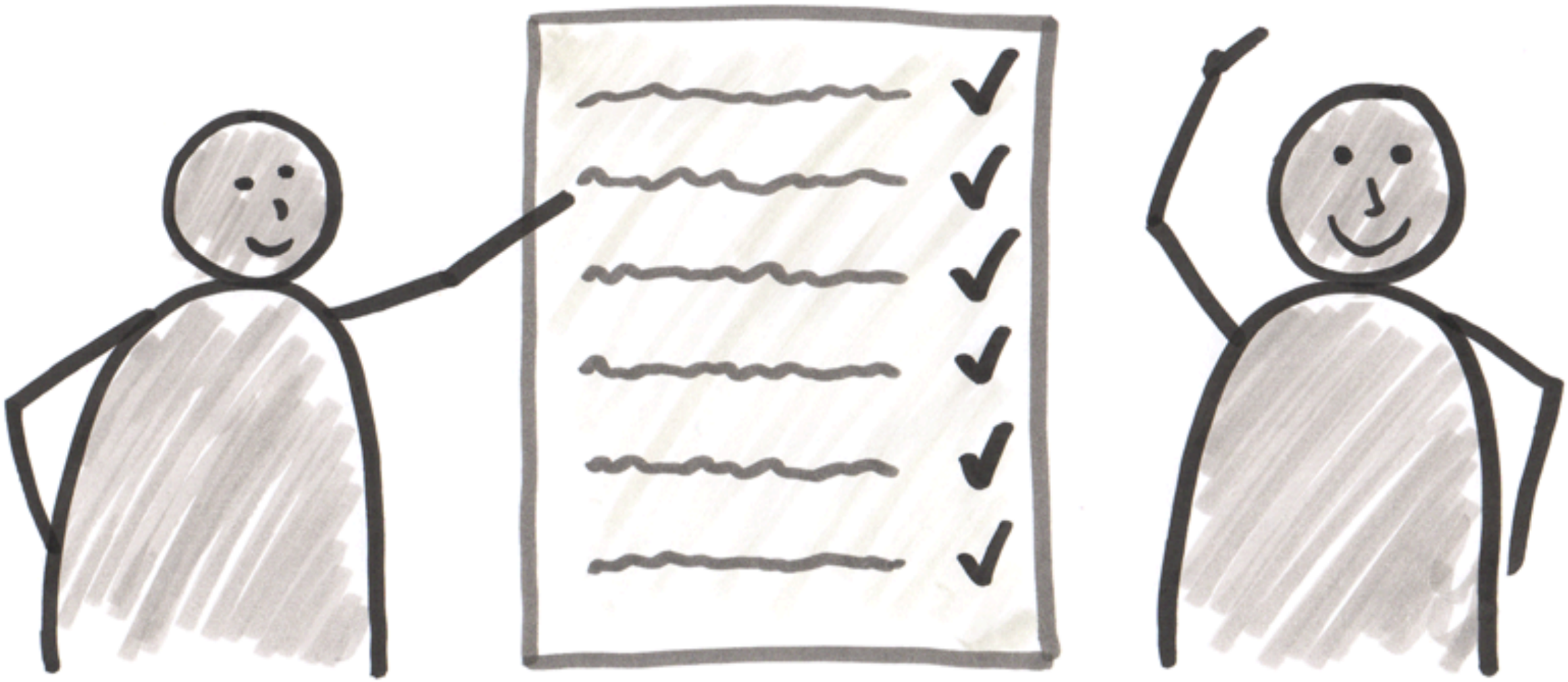
HOW TO REFACTOR COMPLEX CONDITIONALS?

Problem	How to Refactor	Refactoring Name
complex conditional logic that does several variants of calculation	create a Strategy for each variant and use a Strategy instance to do the actual calculation	Replace Conditional Logic with Strategy
conditions check for many special case conditions that should be executed in addition to the core behaviour	move the special condition to a Decorator	Move Embellishment to Decorator
complex conditional expressions control object's state	replace the conditions with State classes that would internally handle the states and transitions between them	State-Altering Conditionals with State
Condition check for null condition is repeated throughout the code because the object can be null	Introduce a Null Object and remove that null check	Introduce Null Object
large switch has logic for executing different actions	create a Command for each action and replace the conditional execution logic with the execution of commands	Replace Conditional Dispatcher with Command
using conditional logic, a method accumulates information to a local variable from heterogeneous classes	move the accumulation task to a visitor that can visit each class and get that information	Move Accumulation to Visitor

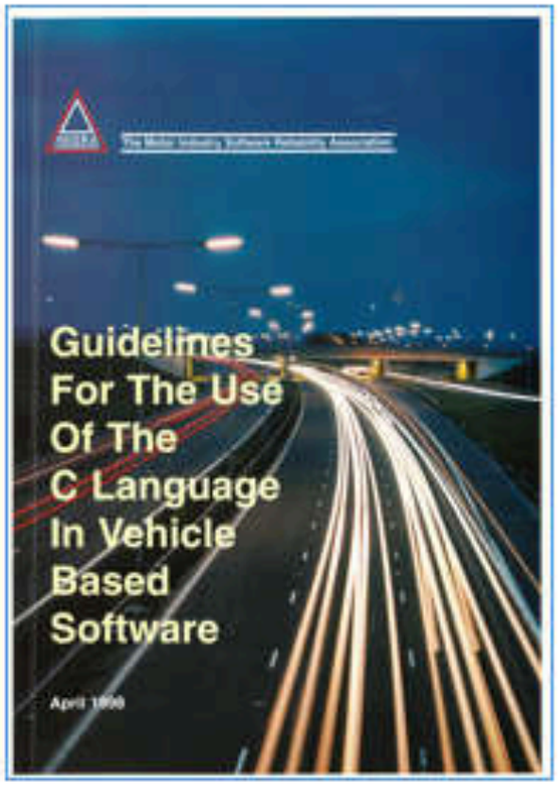
HOW TO REFACTOR DUPLICATE CODE?

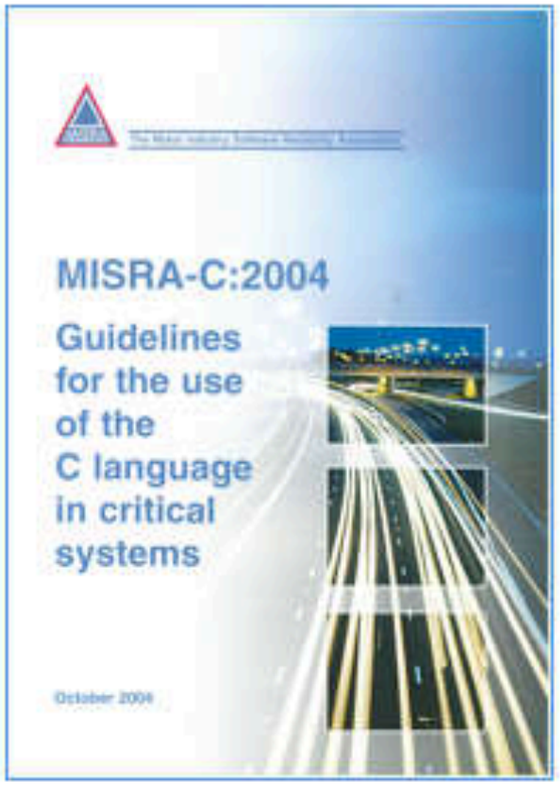
Code Duplication Type	How to Refactor	Refactoring Name
Methods in subclasses have similar code with steps done in same order; still the steps are different	Generalize the method by extracting the steps with same method name and pull up the method to the base class as a Template Method	Form Template Method
Methods in derived classes are same except for the object creation step	Create a super class version of the method that calls a Factory Method to do that instantiation	Introduce Polymorphic Creation with Factory Method
Constructors of a class have duplicate code	Chain the constructors and remove the duplicate code	Chain Constructors
There is separate code for handling single object or a collection of objects	replace that condition with single code segment for handling both single and collection of objects	Replace One/Many Distinctions with Composite
Subclasses in a hierarchy implement their own collection of objects and their implementation is identical	extract a super class that implements the collection of objects	Extract Composite
Processing logic is different just because the objects have different interfaces	provide adaptor classes so that classes have same interfaces	Unify Interfaces with Adapter
The check for null condition is repeated throughout the code because the object can be null	Introduce a Null Object and remove that null check	Introduce Null Object

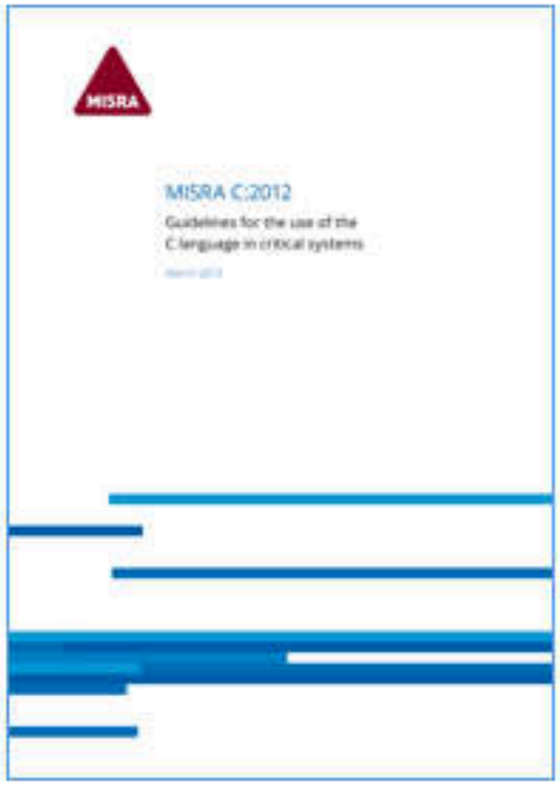
CODING GUIDELINES

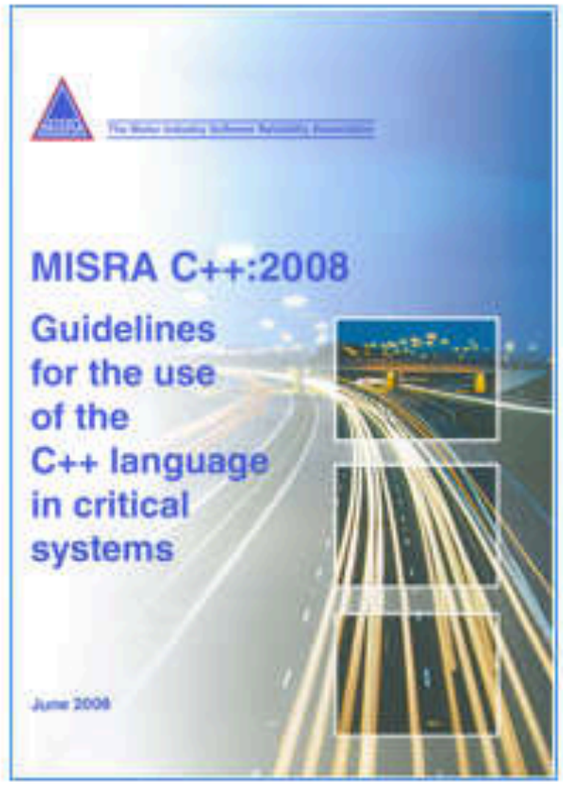


MISRA C/C++



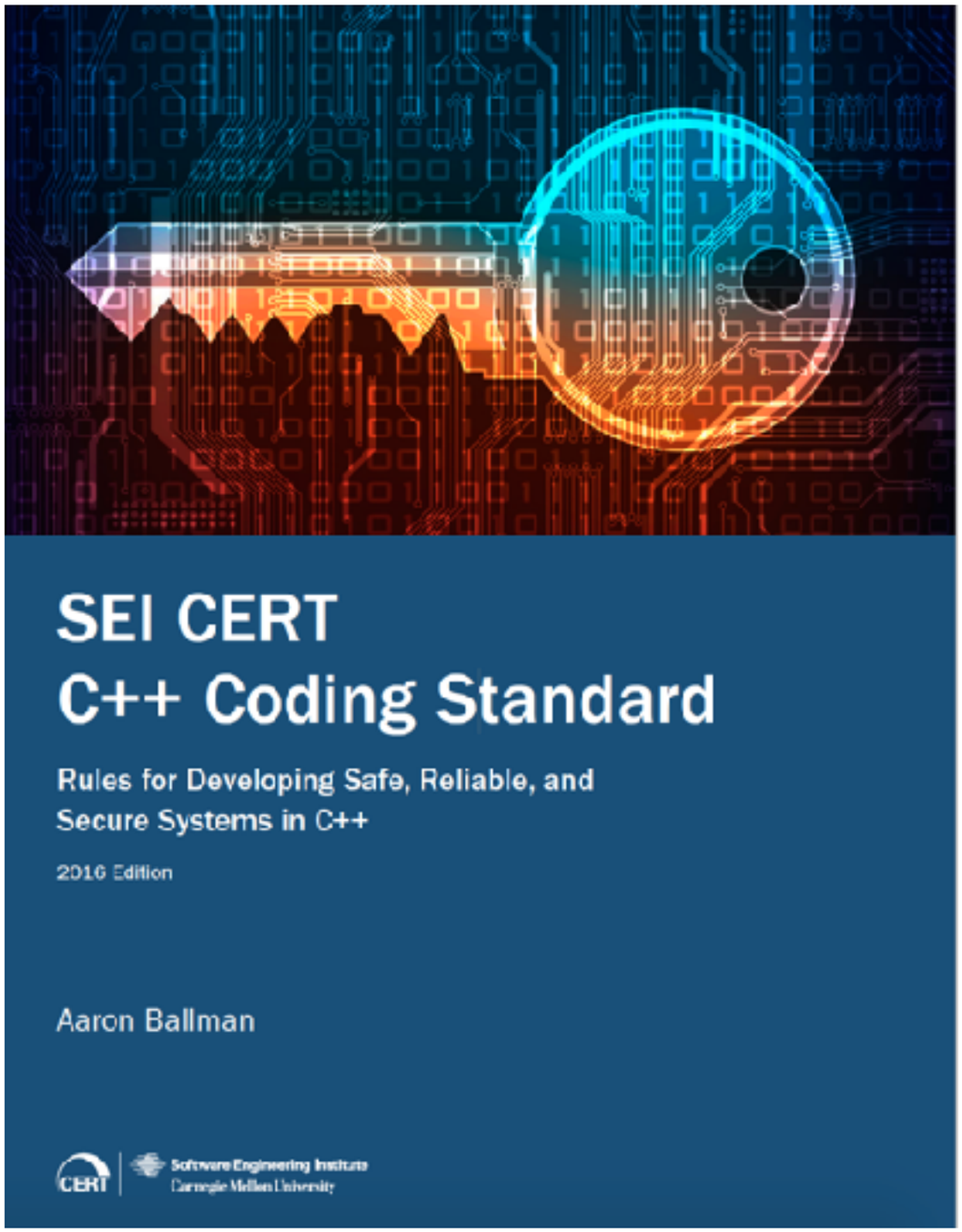






https://www.misra-cpp.com/MISRACHome/tabid/128/Default.aspx

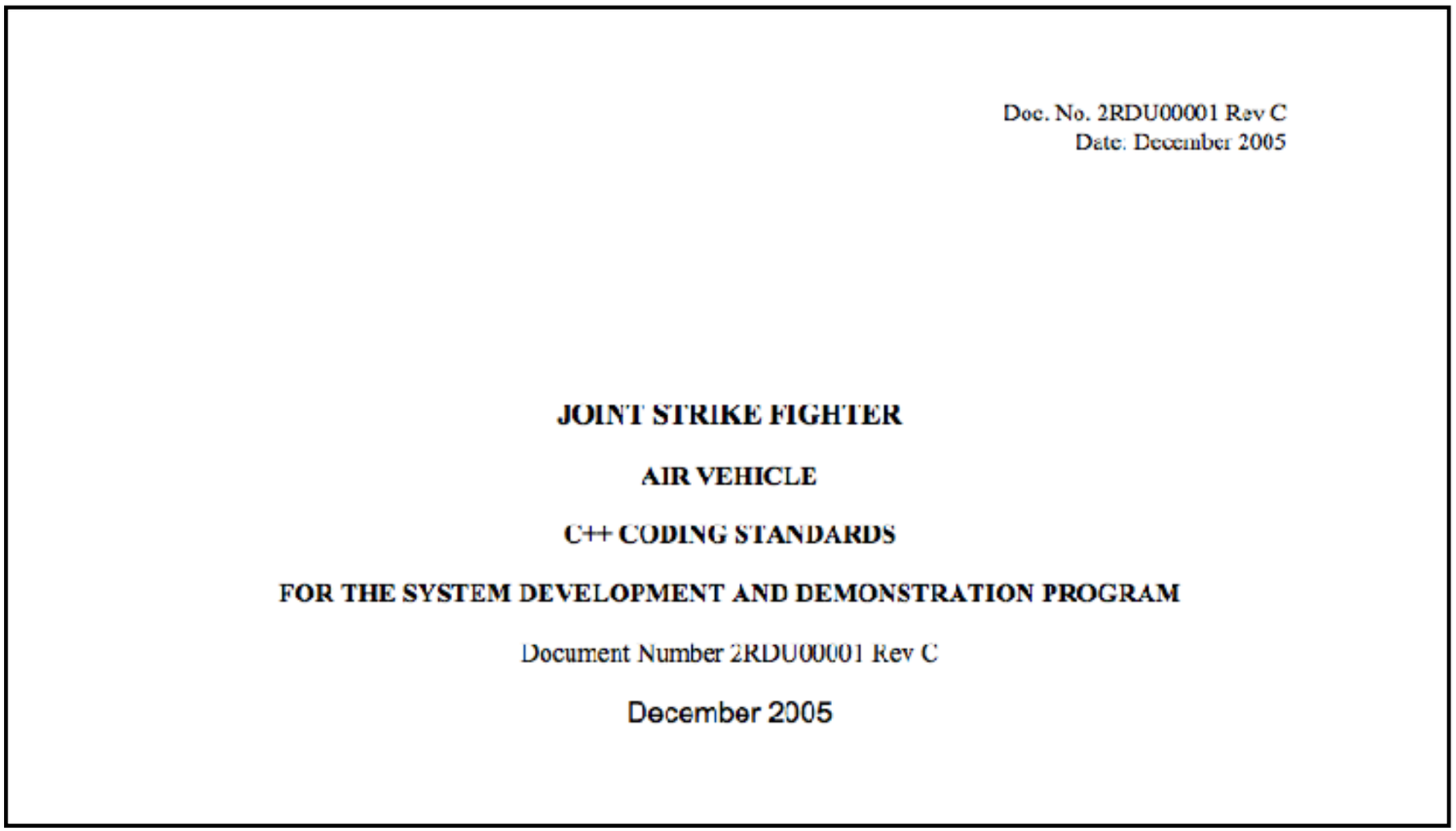
CERT C++



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

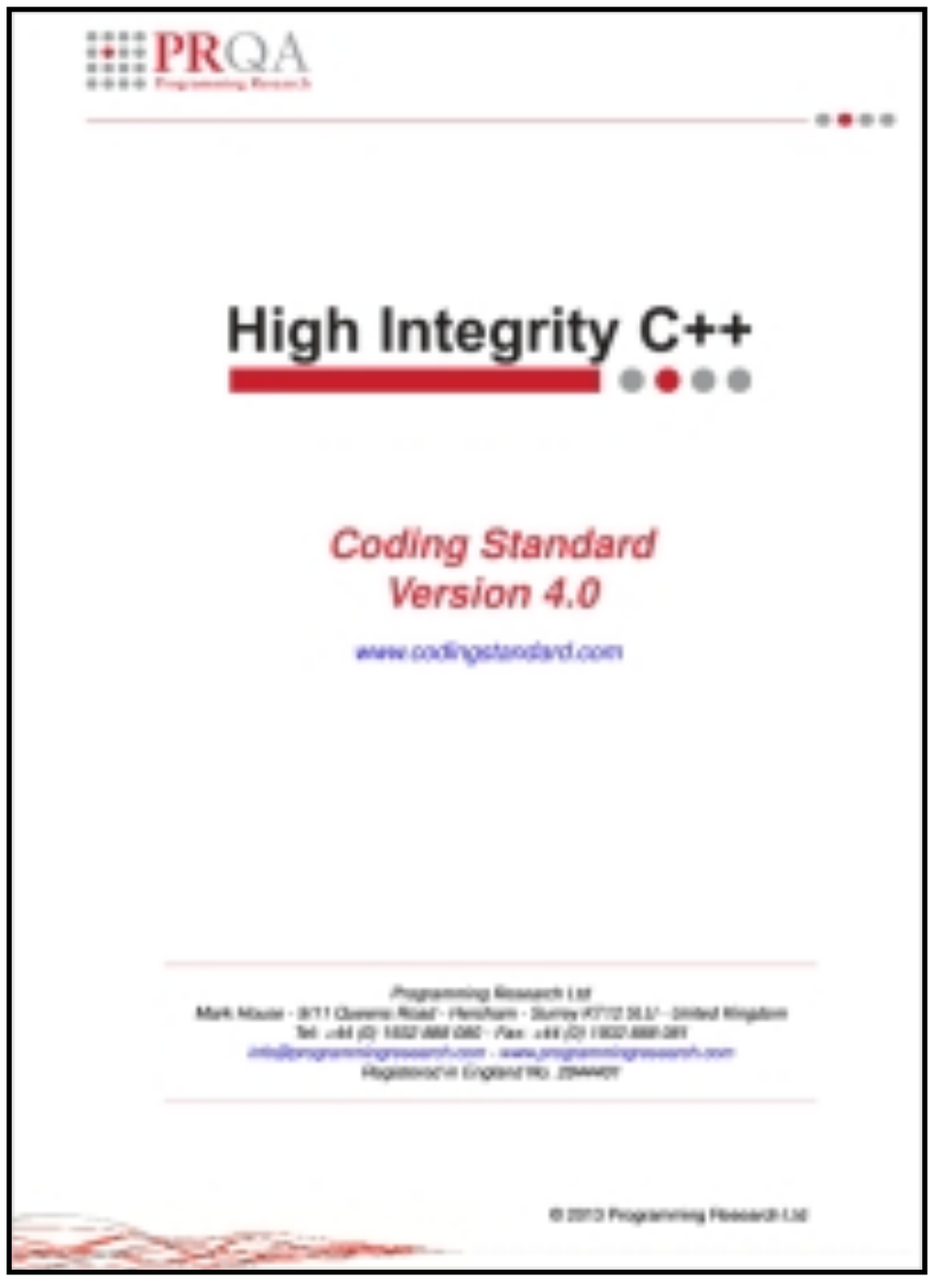
FOR THE SYSTEM DEVELOPMENT AND DEMONSTRATION PROGRAM

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

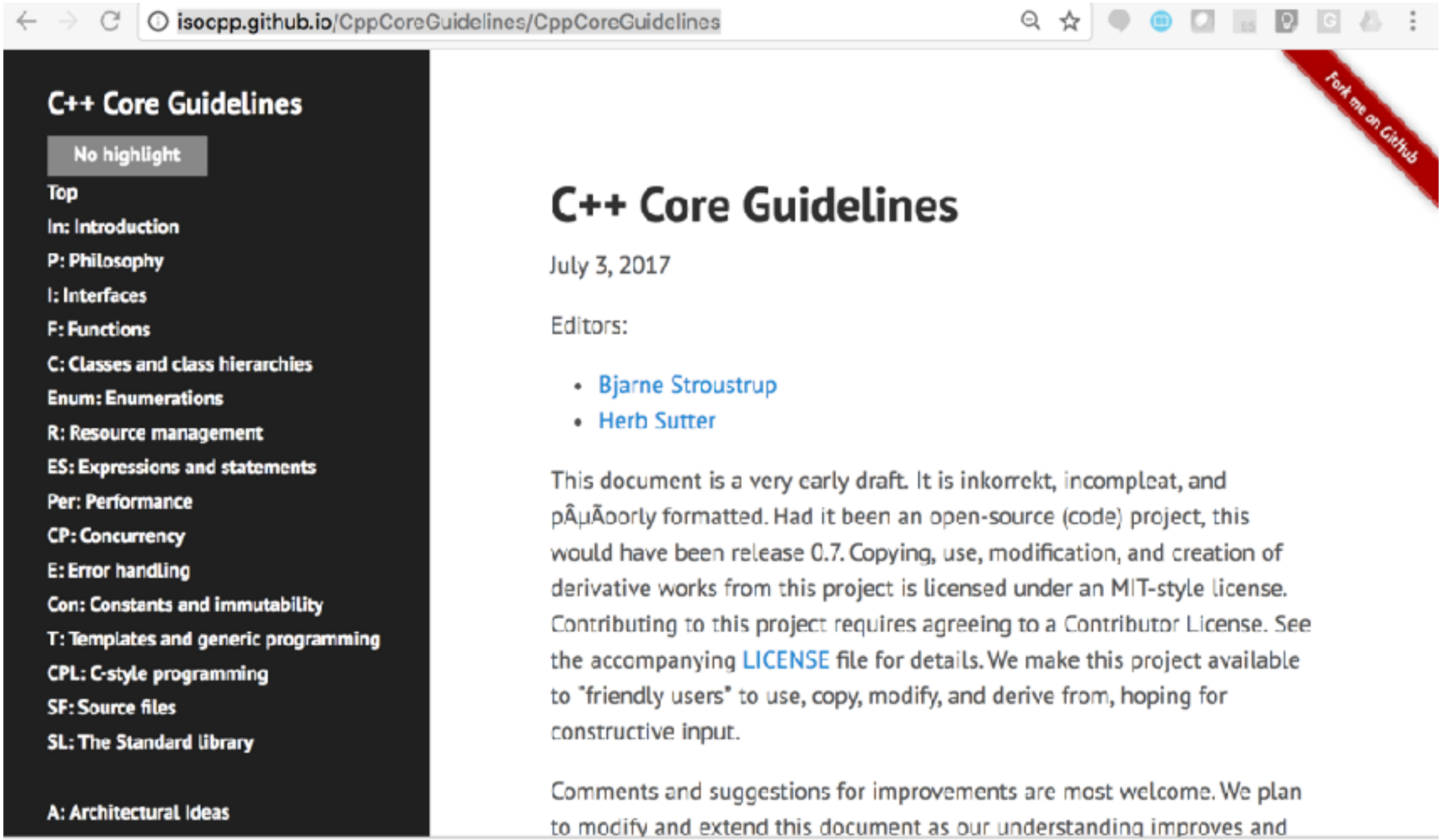
Google C++ Style Guide

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

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
```

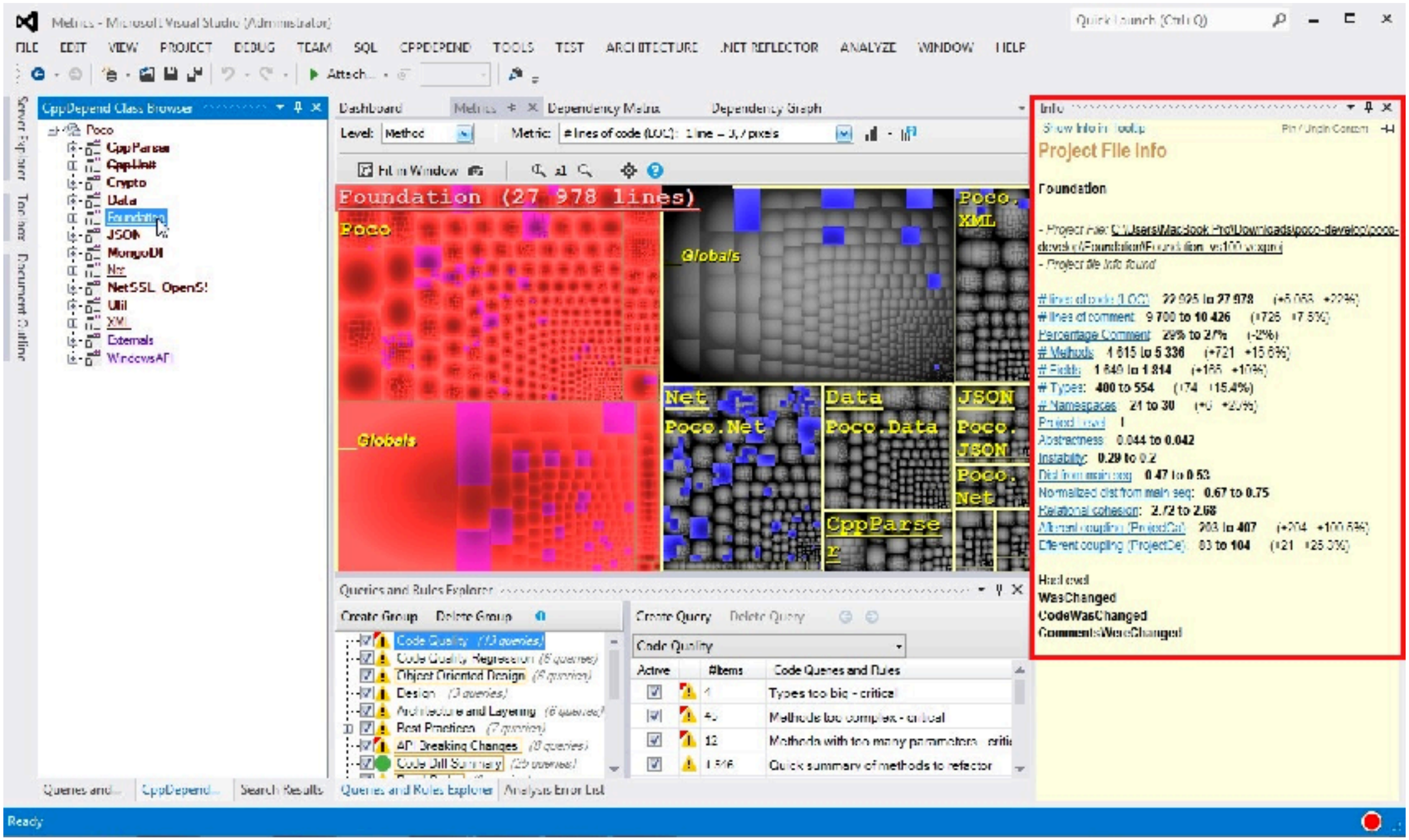
http://www.harukizaemon.com/simian/

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);
```

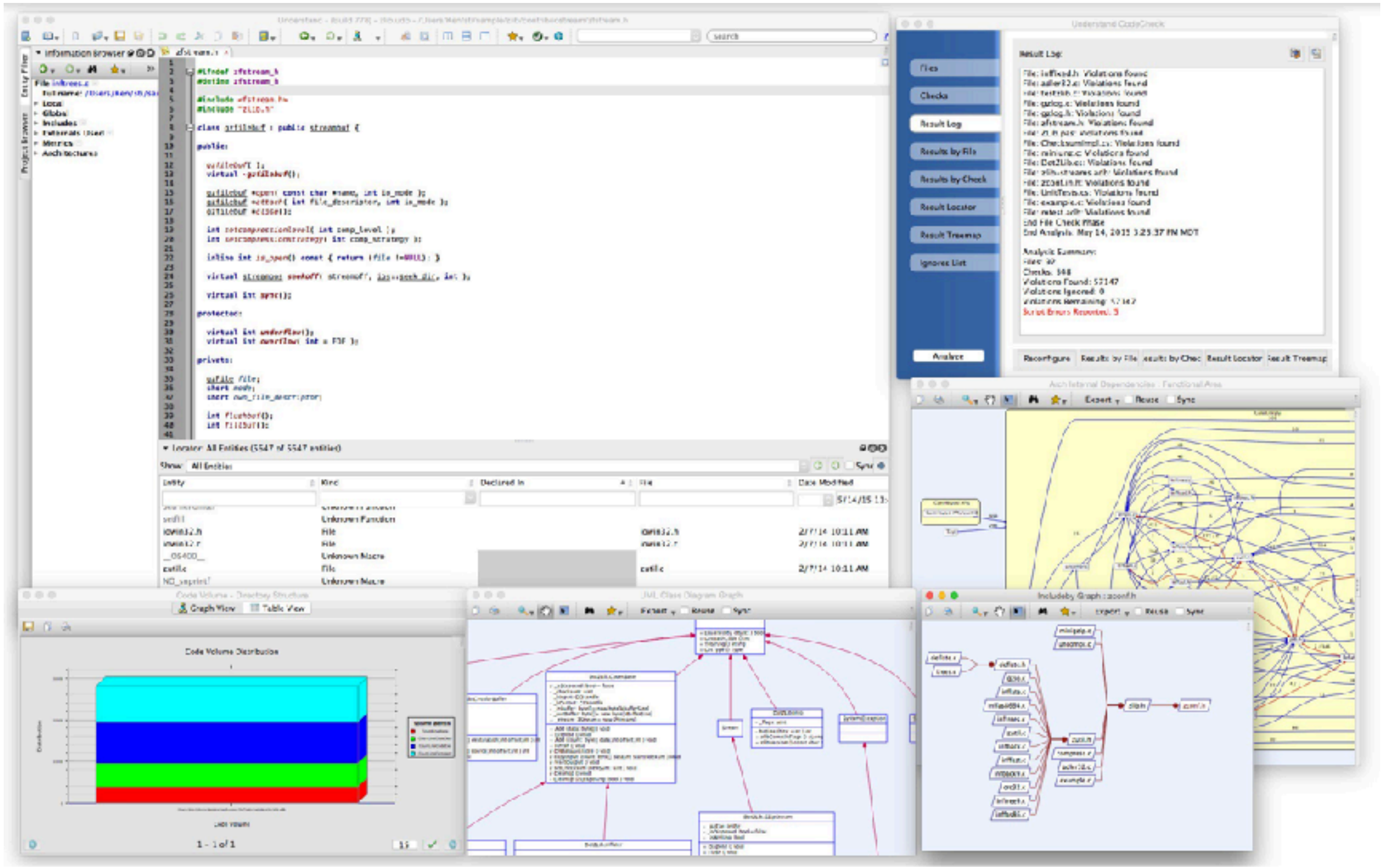
pmd.sourceforge.net/pmd-4.3.0/cpd.html

CPPDEPEND



http://www.cppdepend.com/

UNDERSTAND FOR C++



https://scitools.com/

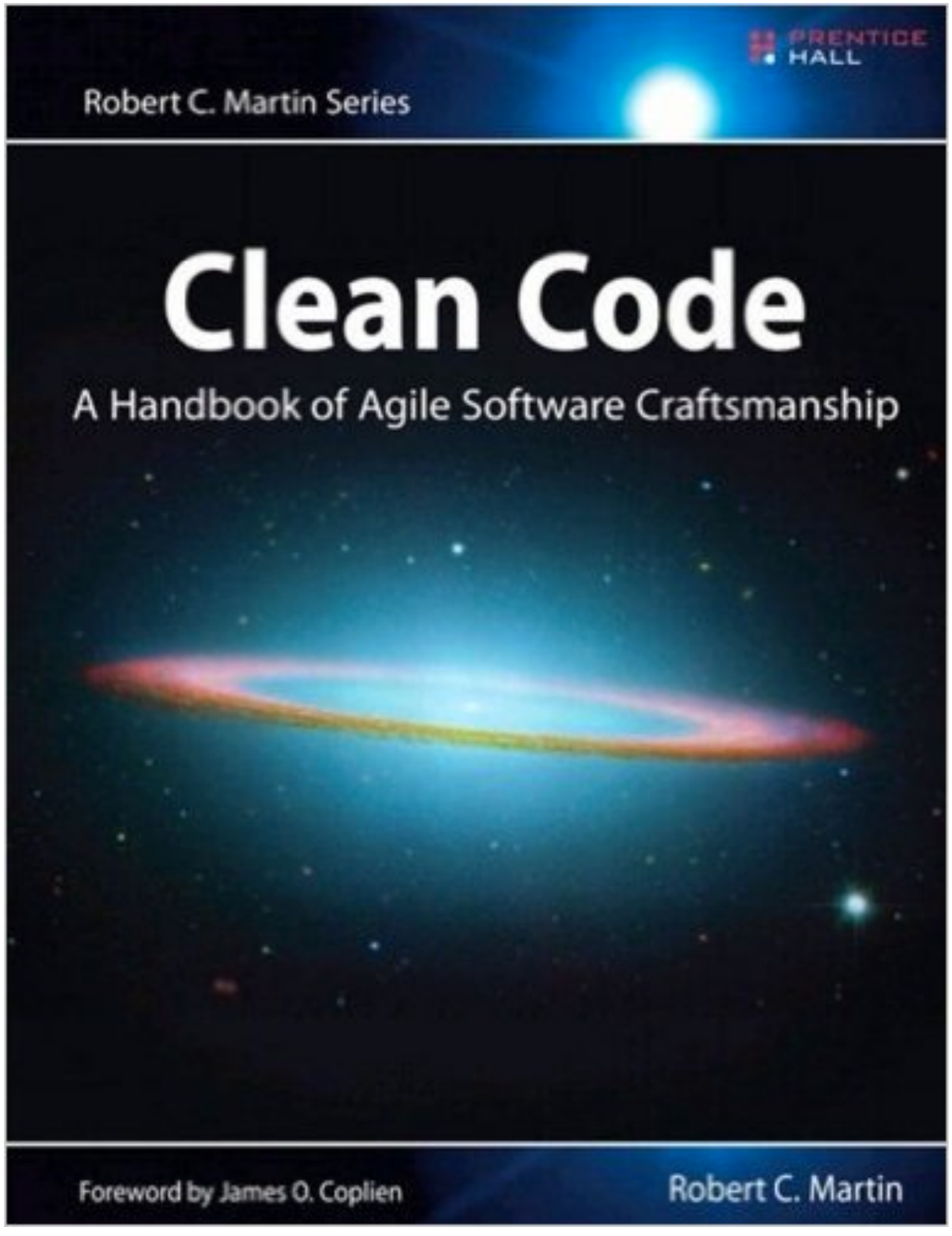
BOOKS TO READ

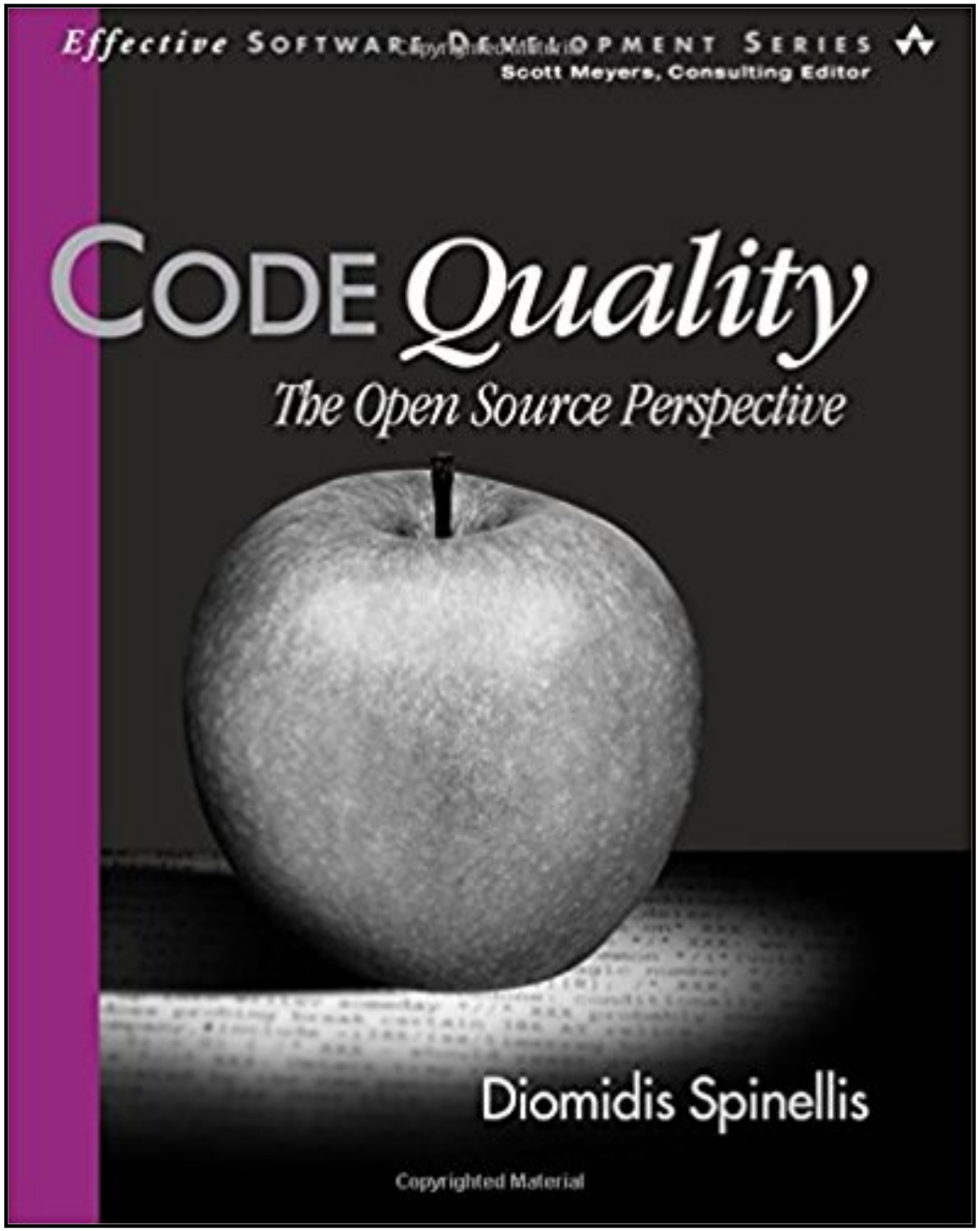
C++ Coding Standards

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

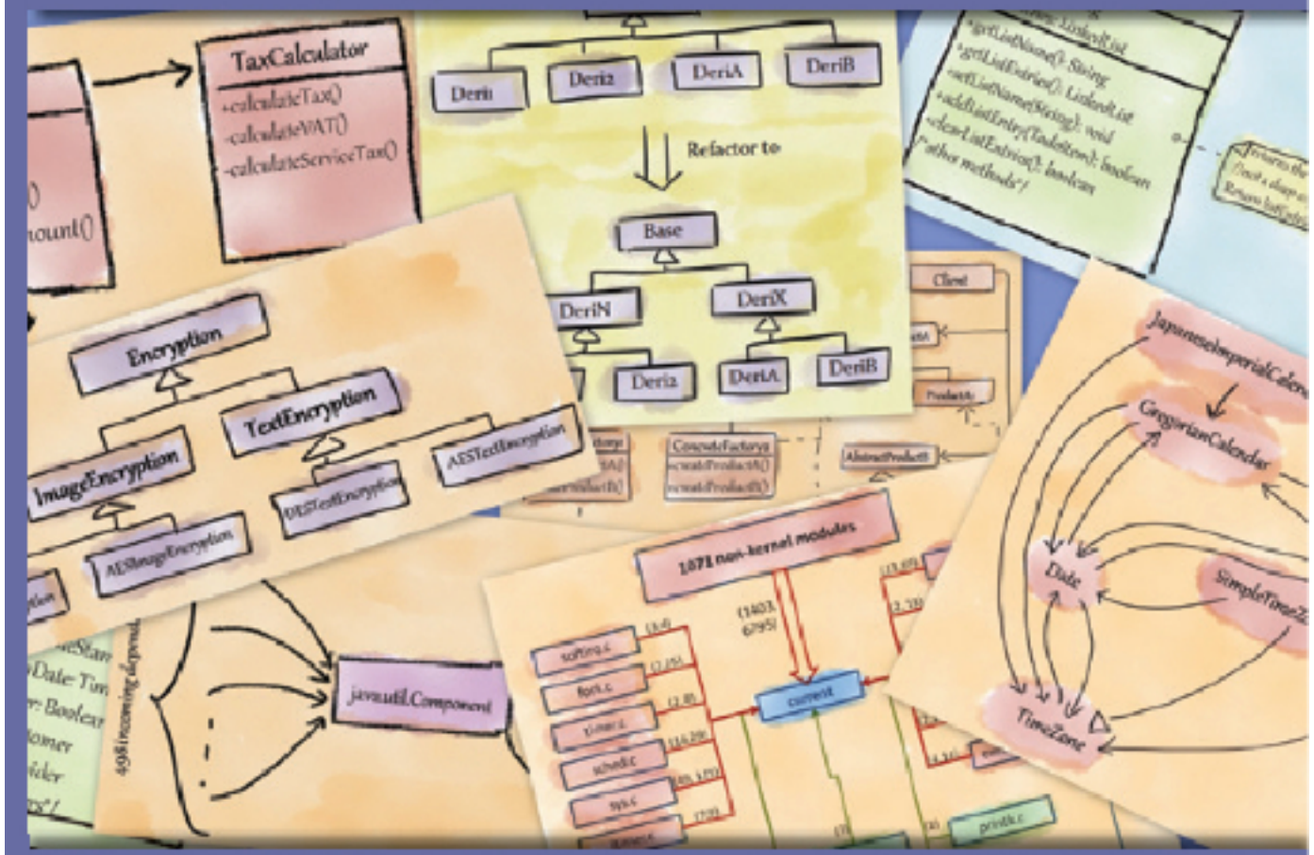
A practical handbook of software construction

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

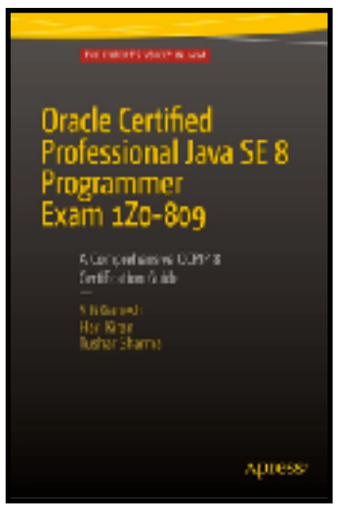
MK
MORGAN KAUFMANN

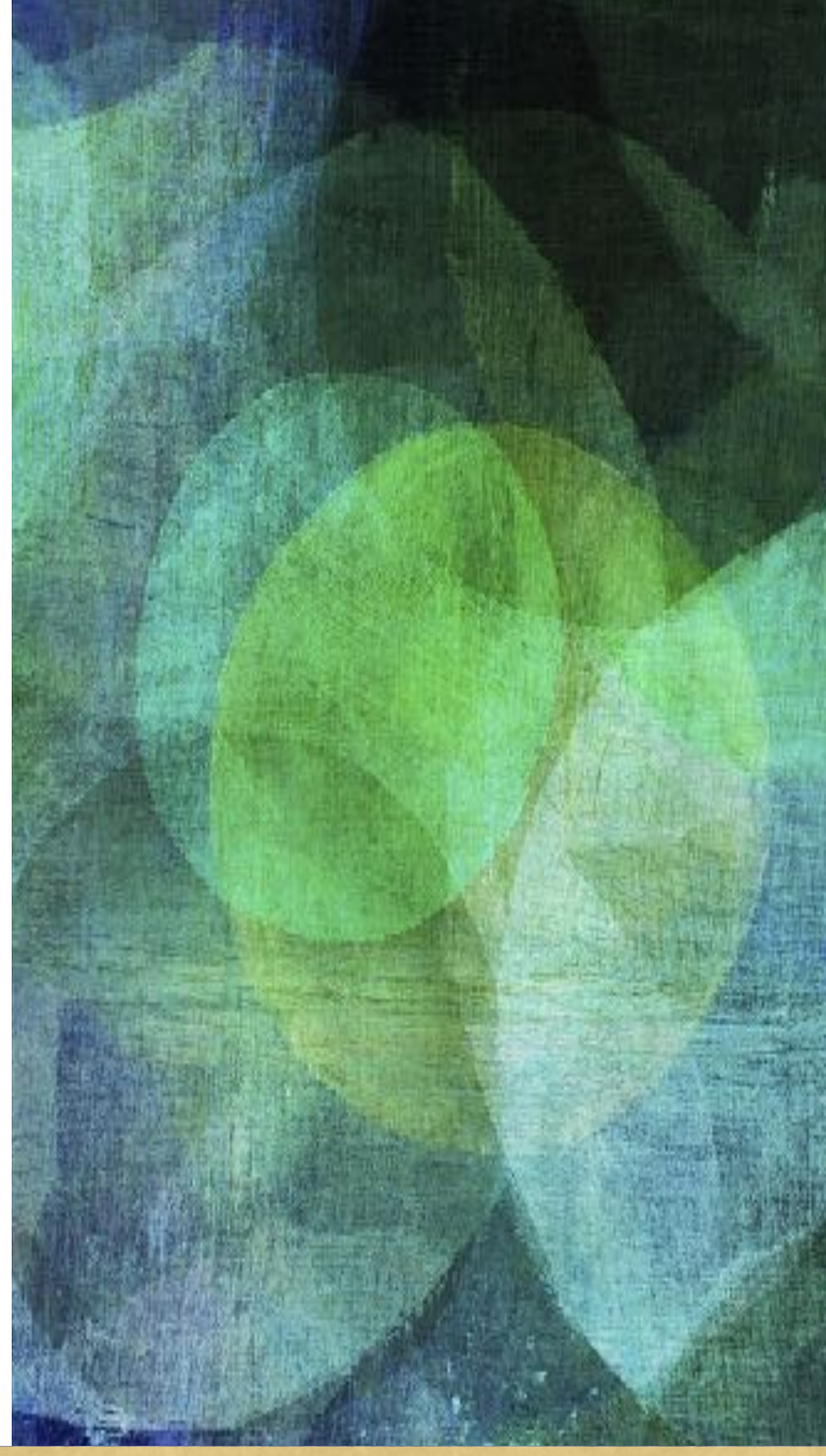
IMAGE CREDITS

- 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