Desire to be proud of what we do

SOFTWARE CRAFTSMANSHIP

PREVENT CODE ROT

BHARAT MANE

LANGLEY AND TEAM'S MINDSET

HAVE TO DO IT

WRITE BROTHERS MINDSET

WANT TO DO IT

IN 1982, A THEORY WAS INTRODUCED BY SOCIAL SCIENTISTS

JAMES Q. WILSON

WINDOW

LITTER

GRAFFITI

STRUCTURE

SENSE OF ABANDONMENT BECOMES REALITY

NEGLECTION ***ACCELERATES*** THE ROT FASTER THAN ANY OTHER FACTOR.

HOW IS THIS RELATED WITH THE SOFTWARE?

Are the next few questions
sounds familiar?

CHANGES ARE **HARD** TO DO?

SIMPLE CHANGE **IMPACTS**
NUMEROUS MODULES?

IMPLEMENTING SIMPLE CHANGE TAKES **FOREVER**?

CHANGE IN ONE PLACE
HARMS SOMEWHERE
COMPLETELY UNEXPECTED
AREA?

FIXING A BUG **CAUSES** <N>
MORE?

MODULES ARE
NOT REUSABLE
BECAUSE OF THEIR
DEPENDENCIES?

IT'S EASIER
TO DO **HACKS**
THAN BY THE BOOK?


BINGO!

```
public static String testableH
   PageData pageData,
   boolean includeSuiteSetup
 ) throws Exception {
   WikiPage wikiPage = pageDat
   StringBuffer buffer = new S
   if (pageData.hasAttribute("
     if (includeSuiteSetup) {
       WikiPage suiteSetup =
         PageCrawlerImpl.getIn
                 SuiteResponde
         );
       if (suiteSetup != null)
         WikiPagePath pagePath
           suiteSetup.getPageC
         String pagePathName =
```

```
public List<ir
  List<int[]>
  for (int[] x
    if (x[0] =
      list1.ac
  return list1
}
```

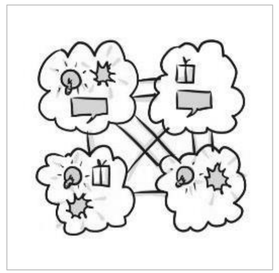
```
int a = 1;
if (0 == 1
  a = 01;
else
  1 = 01;
```

```
/** The child Containers belongi
protected HashMap children = new

/** The processor delay for this
protected int backgroundProcesso

/** The lifecycle event support
protected LifecycleSupport lifec

/** The container event listener
protected ArrayList listeners =

/** The Loader implementation wi
protected Loader loader = null;
```

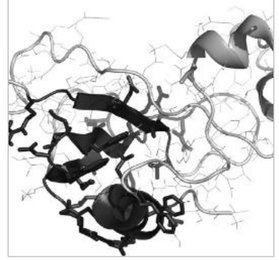
CODE ROT
HAS BEGUN?

SOFTWARE
EROSION

SOFTWARE
DECAY



SOFTWARE
ENTROPY



SOFTWARE
ROT

TECHNICAL
DEBT

WE KNOW IT BY MANY NAMES

HOW CAN WE **STOP** IT?

KEEP THE
CODE CLEAN
AT ALL TIMES

WHAT'S THE **CLEAN CODE**?

...DOES **ONE THING** WELL

-- Bjarne Stroustrup

Inventor of C++

...READS LIKE WELL-WRITTEN **PROSE**

-- Grady Booch

Inventor of UML

...IT WAS WRITTEN BY SOMEONE

WHO CARED

-- Michael Feathers

Author of Working effectively with Legacy Code

...FEW PROGRAMMERS KNOW HOW TO WRITE CODE THAT **HUMAN UNDERSTAND**

-- Martin Fowler

Author of Refactoring

...IS WHEN EACH METHOD YOU READ TURNS OUT TO BE PRETTY MUCH WHAT **YOU EXPECTED**

-- Ward Cunningham

Inventor of Wiki

Coinventor of Extreme Programming

OUR GOAL IS

NOT TO CODE

BUT TO

COMMUNICATE

1

WHILE MAKING
CHANGES
TO EXISTING CODE

2

WHILE ADDING
NEW CODE

1

WHILE MAKING
CHANGES
TO EXISTING CODE

2

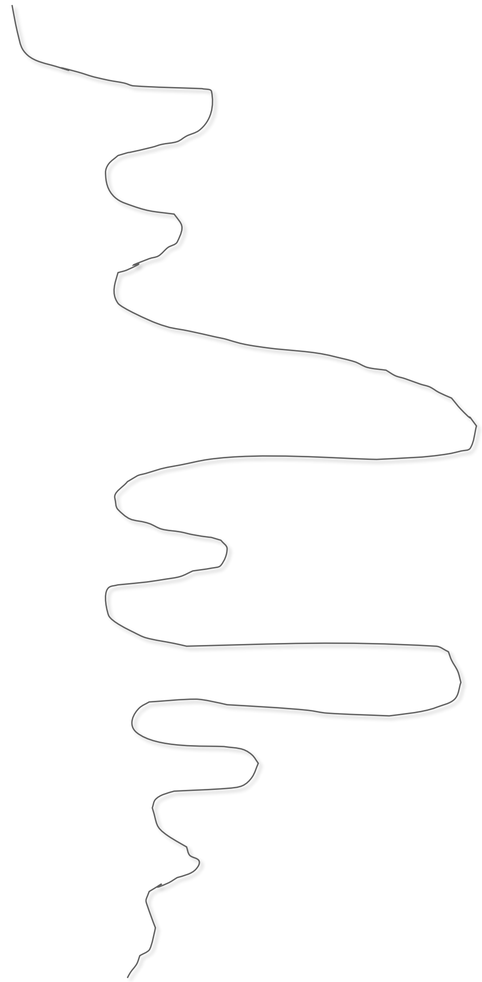
WHILE ADDING
NEW CODE

ANY IDEA WHEN
EXISTING CODE
BECOMES **LEGACY?**

THE MOMENT WE
STOP REFACTORING
OR **STOP CARING**

MAINTENANCE **START** THE MOMENT
WE **WRITE** THE CODE

WHAT HAPPENS WHEN NOT CARED



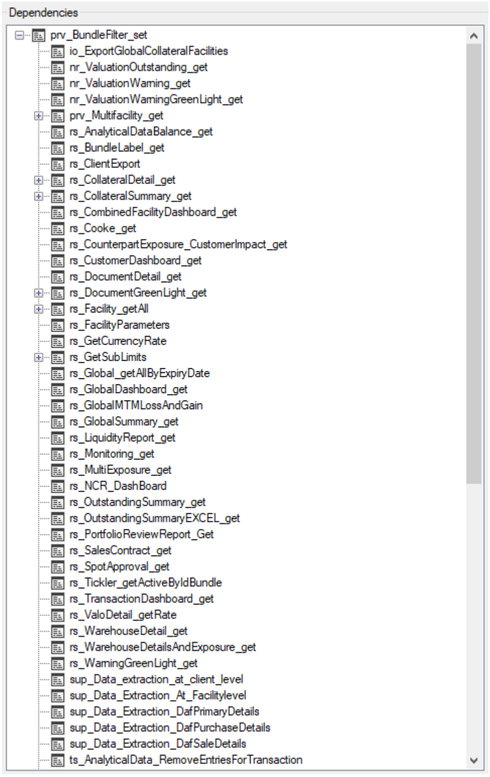
“DON’T PANIC
I KNOW LITTLE REFACTORING
LET’S DO IT!!”

```
CREATE PROCEDURE dbo.prv_BundleFilter_set_old
 @valueDate DATETIME ,
    @asOfDate DATETIME ,
    @idBundle INT = NULL ,
    @localizationSG INT = NULL ,
    @idFOCode INT = NULL ,
    @isMigrated BIT = NULL ,
    @idNCRStatus INT = NULL ,
    @idCollateralStructure INT = NULL ,
    @isActive INT = 1 ,
    @idProfile INT = NULL
AS

DECLARE @foCodesTable TABLE
    (
      idFrontOfficeCode INT
    )

INSERT  INTO @foCodesTable
        SELECT  *
        FROM    fn_GetDistinctFOCodes(@idProfile)

DECLARE @CombinedFacility TABLE
    (
      idBundle INT NULL INDEX IX_idbundle,
      idBBMultiFacilityHeader INT NULL ,
      isActive BIT
    )
```

```
CREATE FUNCTION dbo.ts_Bundle_create
(
)

-- the @summary variable table is central to the function and being filled step by step
RETURNS @Summary table (
        idBundle int null INDEX idx_idBundle
        ,idBundleUp int null
        ,lft int null
        ,rgt int null
        ,idTree int null
        ,idOutstandingValo int null
        ,idOutstandingValoEOD int null
        ,idValo int null
        ,valueDate datetime null
        ,asOfDate datetime null
        ,idItemType int null
        ,type varchar(50) null
        ,idKey varchar(50) null
        ,label varchar(255) null
        ,ca1isodev char(3) null
        ,ca1isodevEOD char(3) null
        ,idLocalBL int null
        ,localBL varchar(255) null
        ,idFrontOfficeCode int null
        ,FrontOfficeCode varchar(255) null
        ,isMigrated bit null
        ,idNCRStatus int null

        -- Global Summary Information
        ,limitAmount float null
        ,limitAmountEOD float null
        ,collateralAmount float null
        ,grossCollateralAmount float null
        ,outstandingEODAmount float null
        -- UNDONE by AT : valo outstanding(to be removed) ,outstandingIntradayAmount float nu
        ,outstandingCurrentAmount float null
        ,outstandingTheoricalAmount float null -- relatif aux BB
        ,facilityAvailableEODAmount float null -- (limit - outstanding)
        ,multifacilityAvailableEODAmount float null -- (limit - outstanding)
        ,collatPositionEODAmount float null -- (collat - outstanding)
        ,availableForDrawingEODAmount float null -- min(facilityAvailableAmount, availableFor
        ,facilityAvailableCurrentAmount float null -- (limit - outstanding)
        ,multifacilityAvailableCurrentAmount float null -- (limit - outstanding)
        ,collatPositionCurrentAmount float null -- (collat - outstanding)
        ,availableForDrawingCurrentAmount float null -- min(facilityAvailableAmount, availabl

        -- SG Summary Information
        ,SGLimitAmount float null
        ,SGLimitAmountEOD float null
        ,SGCollateralAmount float null
        ,SGGrossCollateralAmount float null
        ,SGCurrentCollateralAmount float null -- just for shared BB current collat
        ,SGCurrentGrossCollateralAmount float null -- just for shared BB current collat
        ,SGoutstandingEODAmount float null
        -- UNDONE by AT : valo outstanding(to be removed) ,SGoutstandingIntradayAmount float
        ,SGoutstandingCurrentAmount float null
        ,SGOutstandingTheoricalAmount float null
        ,SGFacilityAvailableEODAmount float null -- (limit - outstanding)
        ,SGmultiFacilityAvailableEODAmount float null -- (limit - outstanding)
        ,SGCollatPositionEODAmount float null -- (collat - outstanding)
        ,SGAvailableForDrawingEODAmount float null -- min(facilityAvailableAmount, availableF
        ,SGFacilityAvailableCurrentAmount float null -- (limit - outstanding)
        ,SGmultiFacilityAvailableCurrentAmount float null -- (limit - outstanding)
        ,SGCollatPositionCurrentAmount float null -- (collat - outstanding)
        ,SGAvailableForDrawingCurrentAmount float null -- min(facilityAvailableAmount, availa

        ,minNCR float null
        ,imperfectlySecuredLimit float null

        ,perfectlySecuredAmount float null
        ,imperfectlySecuredAvailable float null
        ,unsecuredAmount float null

        ,perfectlySecuredAmountEOD float null
        ,imperfectlySecuredAvailableEOD float null
        ,unsecuredAmountEOD float null

        ,SGperfectlySecuredAmount float null
        ,SGimperfectlySecuredAvailable float null
        ,SGunsecuredAmount float null

        ,SGperfectlySecuredAmountEOD float null
        ,SGimperfectlySecuredAvailableEOD float null
        ,SGunsecuredAmountEOD float null

        ,MarginCallAmount float null

        ,hasMarginCallThreshold int null
        ,thresholdType char(1) null
        ,thresholdValue float null
        ,idMarginCallAlertScope int null
        ,idNCRAlertScope int null

        ,grossEODRatio float null
        ,netEODRatio float null
        ,grossCurrentRatio float null
        ,netCurrentRatio float null

        ,idCollateralStructure int null,
        isOutstandingSGAmount int null,
        isLimitSGAmount int null,
        sharingSG float null,
        sharingSGEOD float null,
        sharingSGCurrent float null,
        sharedBB int null,

        idBBSnapshot int null,
        idSGRole int null,
        Cooke float null,
        SyndicatedCooke float null,
        OutstandingCooke float null,
        UnusedPortionCooke float null,
        commited int null,
        snapshotdate datetime null,

        expiryDate datetime,
        maturityDate datetime,
        effectiveDate DATETIME

        --Added by Anant for itrack 1284178
    ,initialOutstandingSGAmount float null
    ,initialOutstandingGlobalAmount float null
      ,grossIncUSGGCollateralRatio FLOAT NULL
        ,grossIncUSGGlobalCollateralRatio FLOAT NULL
        ,grossIncUSGGCollateralAmount FLOAT NULL
        ,grossIncUSGGlobalCollateralAmount FLOAT NULL
)
as
begin
 return
end
```

102
COLUMNS

100+
DEPENDENCIES



I WAS STILL SCRATCHING THE **SURFACE**

FOLLOWING THE **EXISTING PATTERN**	**UNFAMILIAR** CODE	**INEXPERIENCED** DEVELOPERS
TIME **PRESSURE**	**NEW** TO TEAM	MAKING **SMALLEST** POSSIBLE CHANGE
LACK OF REGULATIONS	DON'T KNOW **WHEN** IT HAPPENED	**HARD** TO UNDERSTAND

WHY THIS STORED PROCEDURE BECAME LIKE THIS?

ATTEMPT 1

SMALL CHANGE & IT FAILED

CODE IS RIGID

ATTEMPT 20+

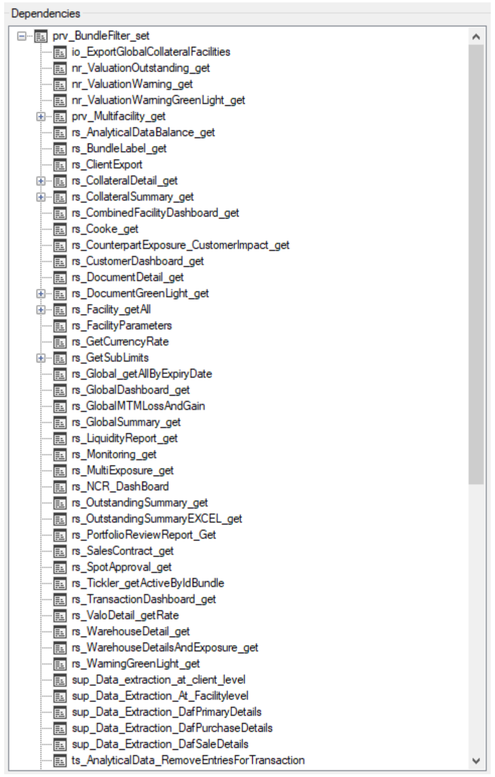
Made changes, it worked!!!
Checked in.

WE RELEASED ☺

After **2 Days** in production
It broke the reports
from completely different part of application

CODE IS FRAGILE

100+
DEPENDENCIES



When looked deeper into the code, dependencies have grown because this module

couldn’t be reused partly

CODE IS IMMOBILE

Module was full of
technical names than
business/domain names

CODE IS NOT READABLE

PERFECT INGREDIENTS FOR?

FEAR


TESTS
FEAR

TEST **AUTOMATION**

UNIT TESTING

FUNCTIONAL TESTING

ZJSONPATCH

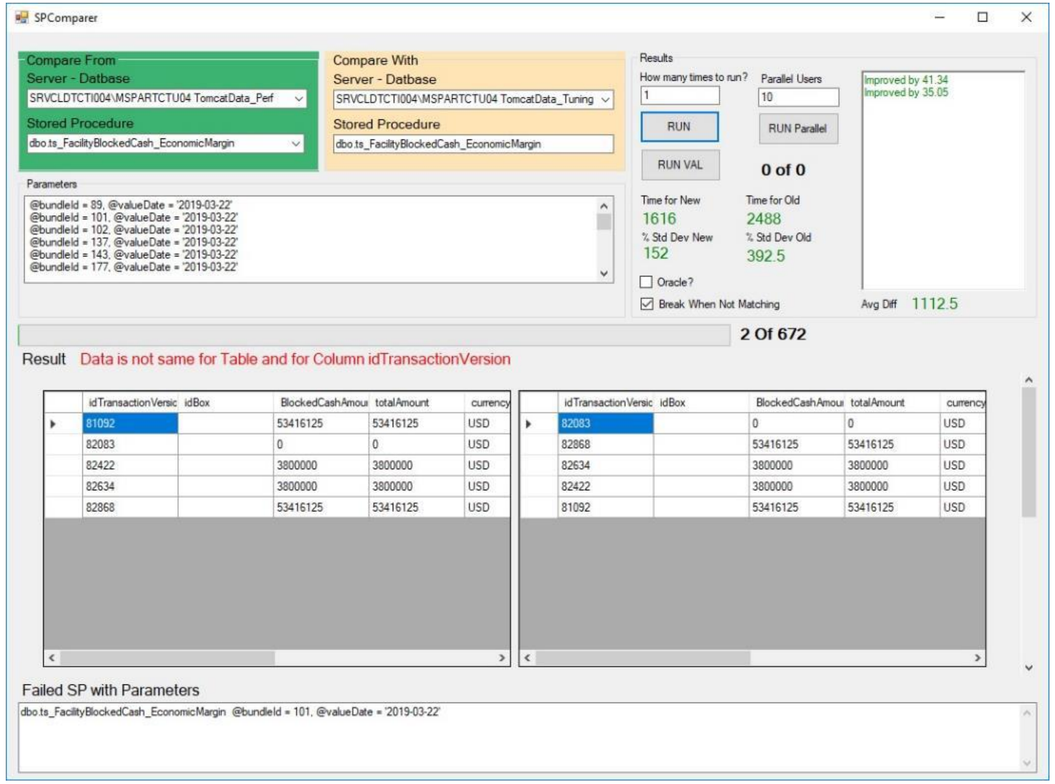
DIFFY

DATABASE NONREG PACK

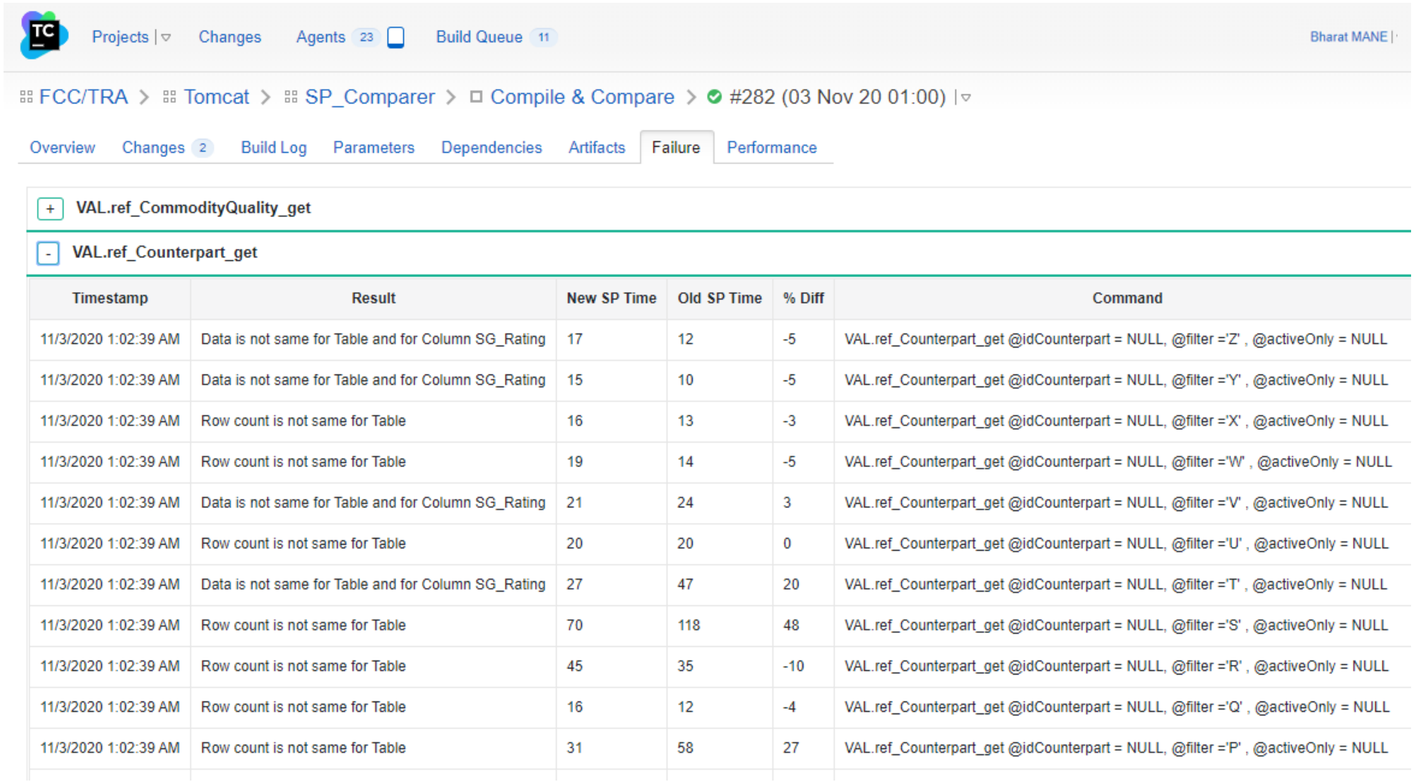
Finds potential bugs in your system comparing the output of ***new code*** and your ***old code***.

STORED PROCEDURE OUTPUT **COMPARER**

STAND ALONE

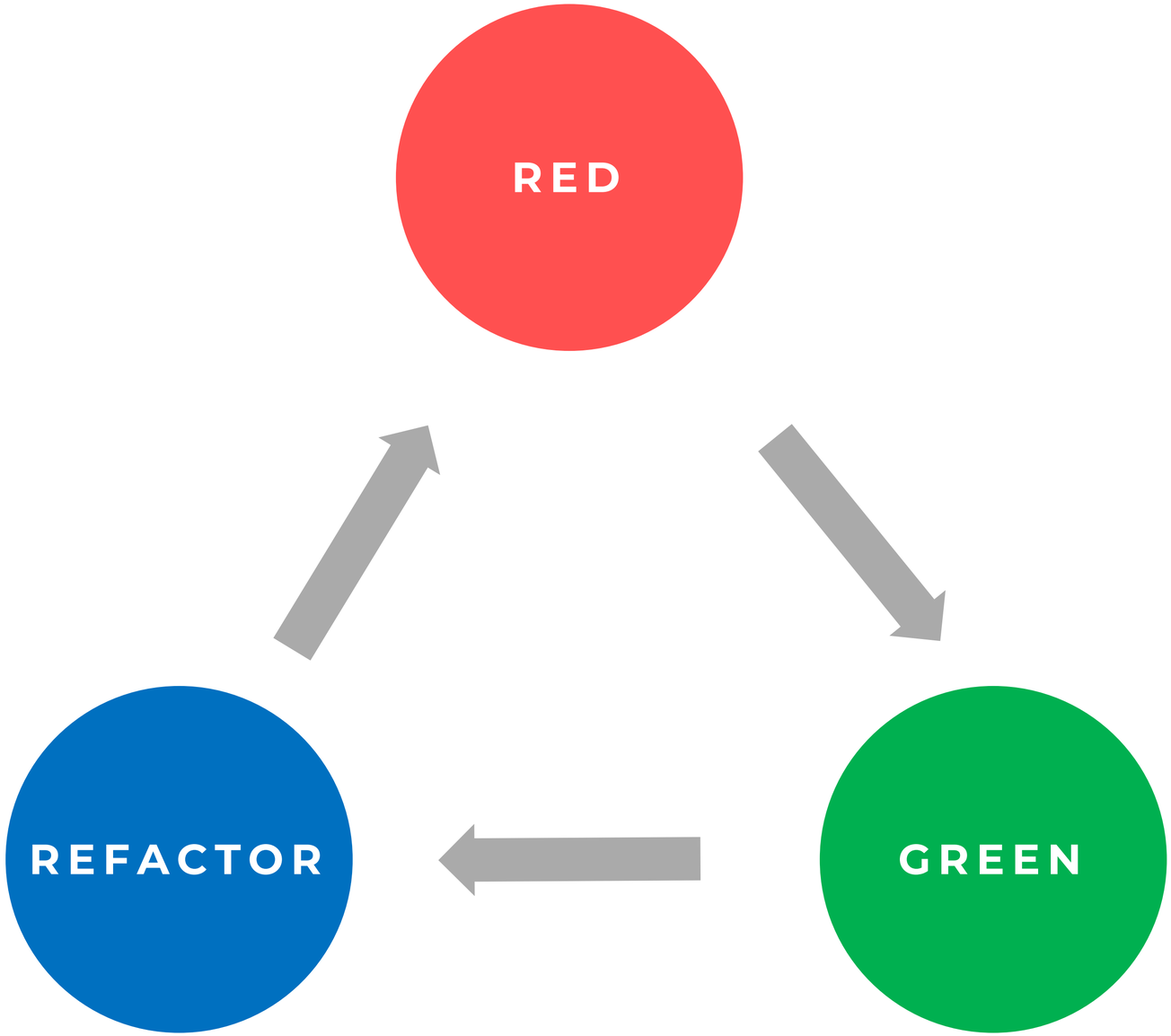


INTEGRATED WITH CI/CD



Timestamp	Result	New SP Time	Old SP Time	% Diff	Command
11/3/2020 1:02:39 AM	Data is not same for Table and for Column SG_Rating	17	12	-5	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='Z' , @activeOnly = NULL
11/3/2020 1:02:39 AM	Data is not same for Table and for Column SG_Rating	15	10	-5	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='Y' , @activeOnly = NULL
11/3/2020 1:02:39 AM	Row count is not same for Table	16	13	-3	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='X' , @activeOnly = NULL
11/3/2020 1:02:39 AM	Row count is not same for Table	19	14	-5	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='W' , @activeOnly = NULL
11/3/2020 1:02:39 AM	Data is not same for Table and for Column SG_Rating	21	24	3	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='V' , @activeOnly = NULL
11/3/2020 1:02:39 AM	Row count is not same for Table	20	20	0	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='U' , @activeOnly = NULL
11/3/2020 1:02:39 AM	Data is not same for Table and for Column SG_Rating	27	47	20	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='T' , @activeOnly = NULL
11/3/2020 1:02:39 AM	Row count is not same for Table	70	118	48	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='S' , @activeOnly = NULL
11/3/2020 1:02:39 AM	Row count is not same for Table	45	35	-10	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='R' , @activeOnly = NULL
11/3/2020 1:02:39 AM	Row count is not same for Table	16	12	-4	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='Q' , @activeOnly = NULL
11/3/2020 1:02:39 AM	Row count is not same for Table	31	58	27	VAL.ref_Counterpart_get @idCounterpart = NULL, @filter ='P' , @activeOnly = NULL

REFACTOR
FEARLESSLY



1

WHILE MAKING
CHANGES
TO EXISTING CODE

2

WHILE ADDING
NEW CODE


WHEN YOUR CODE WORKS THE FIRST TIME

IT'S JUST **HALF** THE JOB DONE

THE TIME THAT THE CODE FIRST WORKS...

...IT'S THE MOMENT IN WHICH YOU START **BUILDING YOUR CAREER**

…**IT'S THE MOMENT** IN WHICH YOU START **BUILDING YOUR DESIGN SKILLS**

DOUBTS

REMORSE

MICRO DECISIONS

SELF REVIEW

IT'S PERFECT MOMENT TO

CLEAN THE CODE

NAMES	FUNCTIONS	COMMENTS
FORMATTING	OBJECTS & DATA STRUCTURES	ERROR HANDLING
CONTINUOUS REFACTORING	PAIR PROGRAMMING	MONITORING

CLEAN CODE TOPICS

NAMES

THE **POWER** INVESTED IN OUR FINGERTIPS


local variable
private function
instance variable
class
argument
directory
global variable
namespace
public function
module
library
package
program
file

…BECAUSE WE NAME **SO MUCH**

…WE MUST USE THIS **POWER WISELY**
WE MUST **DO IT WELL**

TIM OTTINGER

COINED THE PHRASE,
“**IMPLICITY** OF A CODE”

USE **INTENTION-REVEALING** NAMES

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

```
int elapsedTimeInDays;
```

```
public List<int[]> getThem() {
        List<int[]> list1 = new ArrayList<int[]>();
        for (int[] x : theList)
                if (x[0] == 4)
                        list1.add(x);
        return list1;
}
```

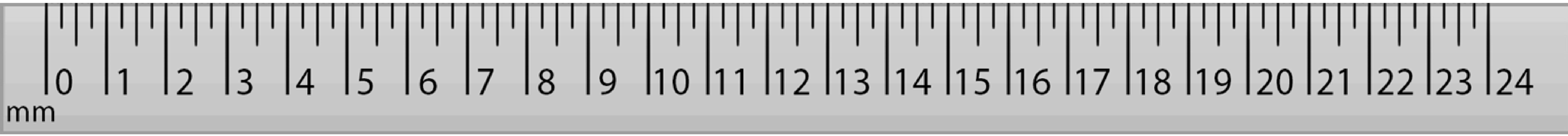


```
public List<int[]> getFlaggedCells() {
        List<int[]> flaggedCells = new ArrayList<int[]>();
        for (int[] cell : gameBoard)
                if (cell[STATUS_VALUE] == FLAGGED)
                        flaggedCells.add(cell);
        return flaggedCells;
}
```

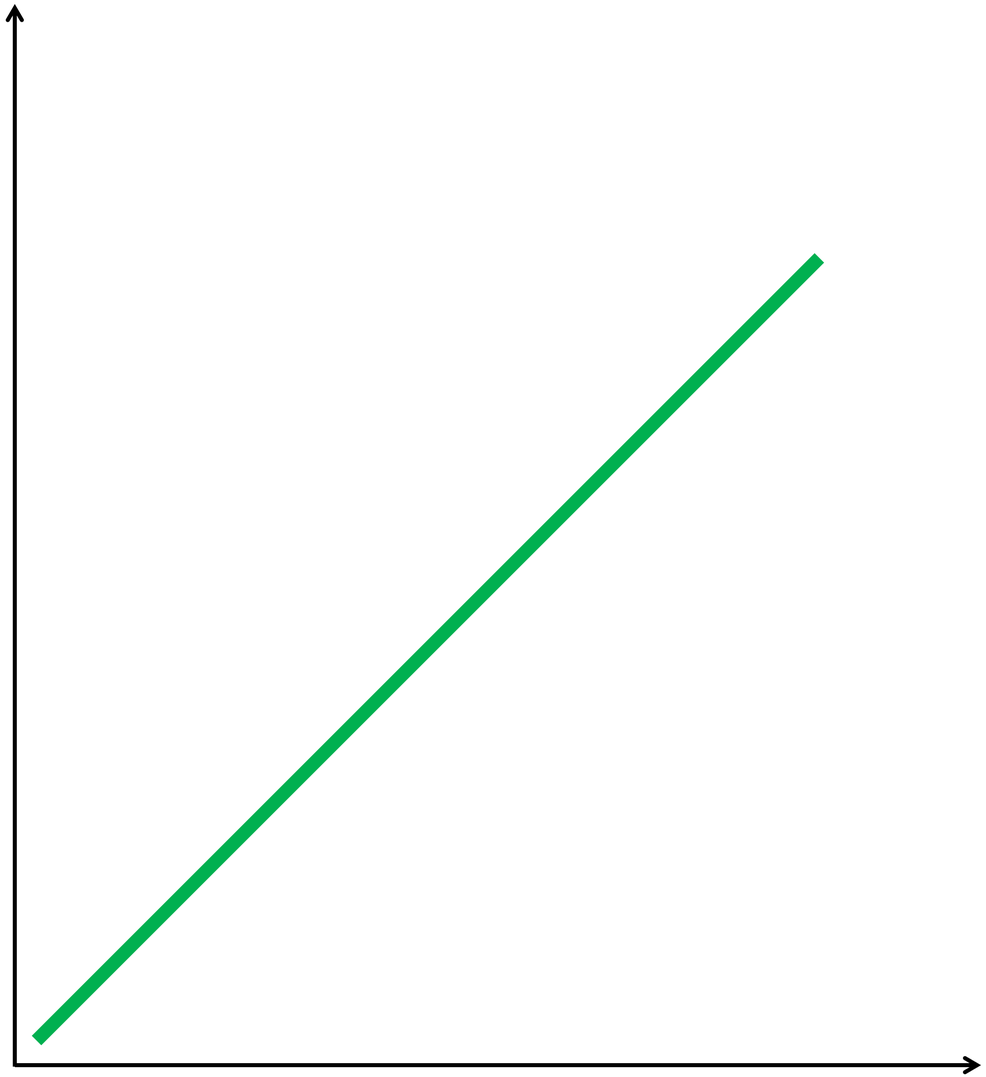
```
public List<Cell> getFlaggedCells() {
        List<Cell> flaggedCells = new ArrayList<Cell>();
        for (Cell cell : gameBoard)
                if (cell.isFlagged())
                        flaggedCells.add(cell);
        return flaggedCells;
}
```

WE **COMMUNICATE**

LENGTH

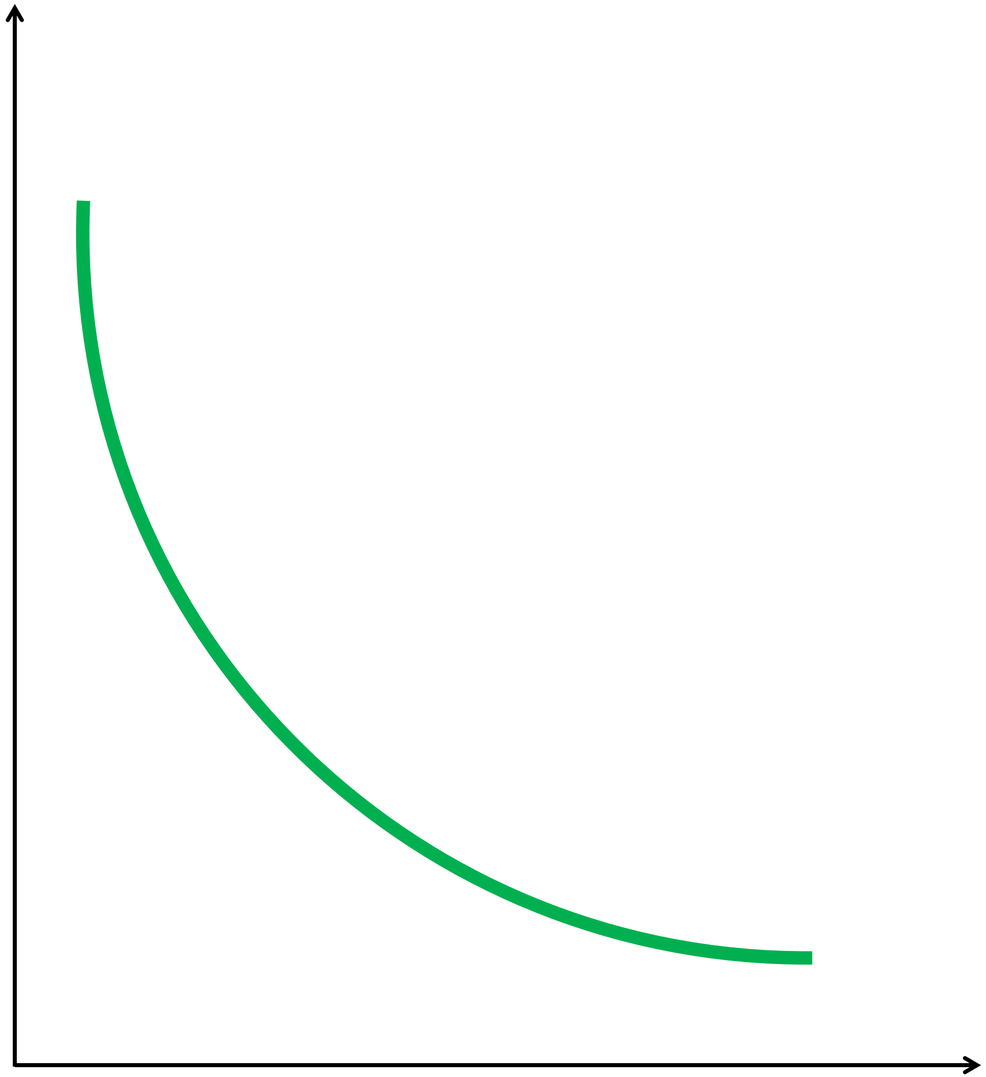


VARIABLE NAME LENGTH



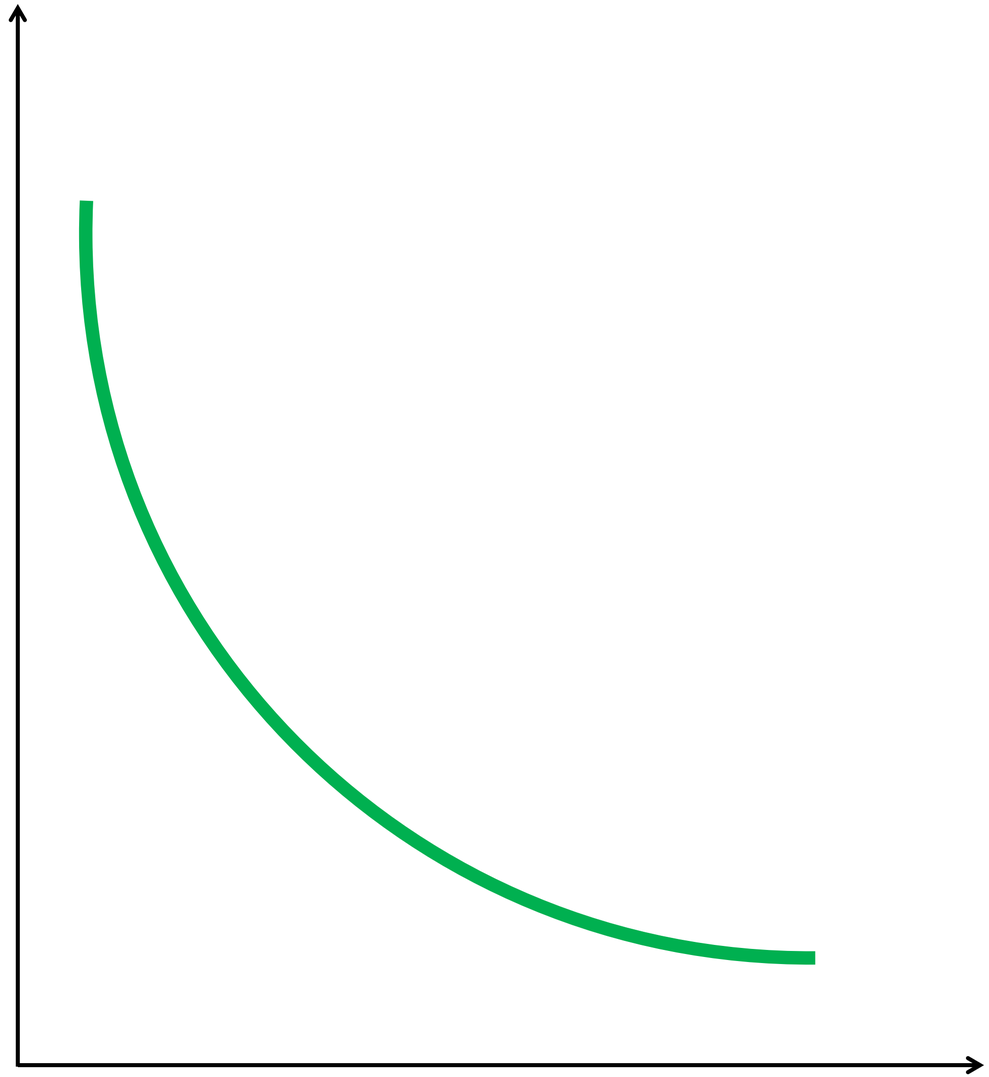
IT'S **SCOPE**

FUNCTION NAME LENGTH



IT'S **SCOPE**

CLASS NAME
LENGTH



IT'S **SCOPE**

V 1.0

- Tomcat.StoredProcedure
  - Properties
  - References
  - App.config
  - Form1.cs
  - Program.cs
  - Utility.cs

V 2.0

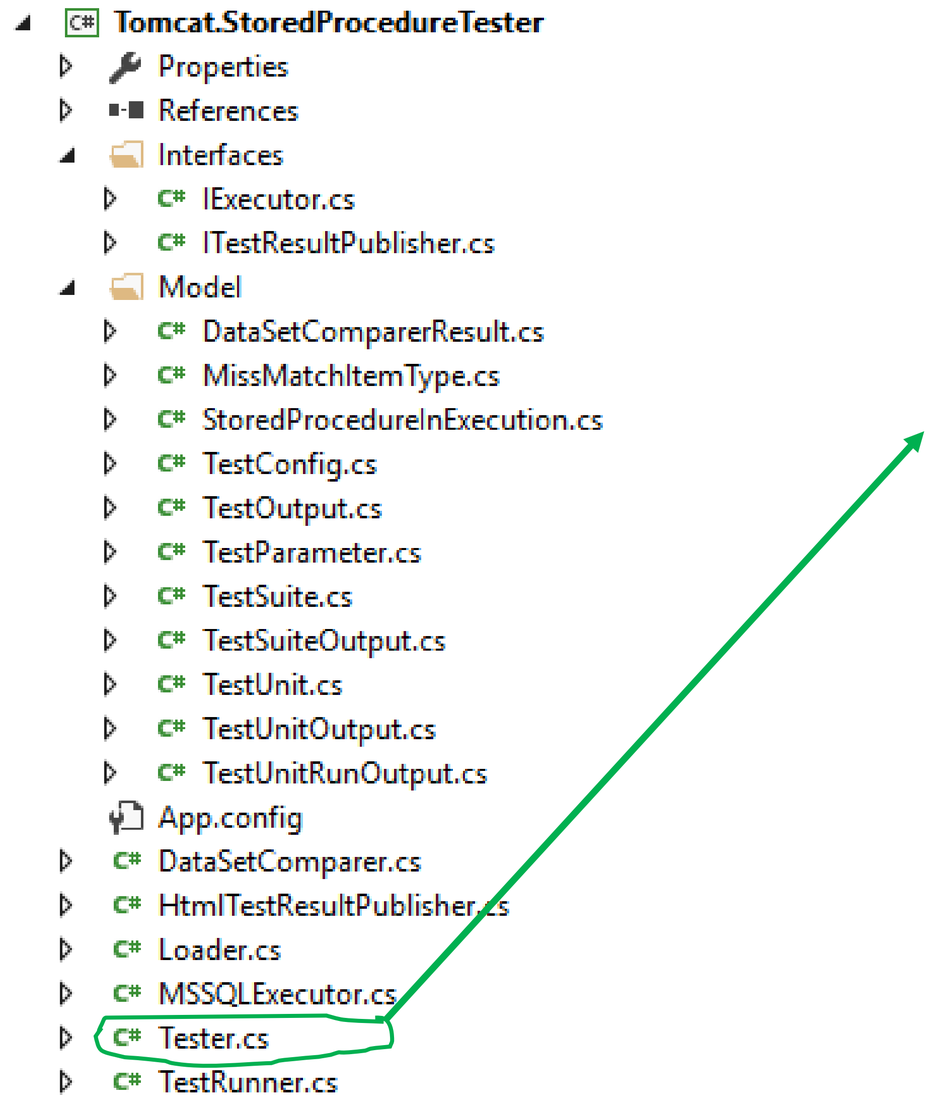
- Tomcat.SPComparer
  - Properties
  - References
  - Config
    - SPs
    - serverconfig.json
  - App.config
  - DataSetComparer.cs
  - DBItem.cs
  - Logger.cs
  - packages.config
  - Program.cs
  - SPComparer.cs
  - SPComparerOutput.cs
  - StoredProcedureComparer.cs
  - Utility.cs

V 3.0

- Tomcat.SPComparer
- Tomcat.SPComparerProvider
  - Properties
  - References
  - CommandOutput.cs
  - ComparerConfiguration.cs
  - ComparerProvider.cs
  - ComparingItem.cs
  - ComparingItemCollection.cs
  - DataSetComparer.cs
  - DataSetComparerResult.cs
  - DBItem.cs
  - ExecutionOutput.cs
  - ExecutionTime.cs
  - IStoredProcedureExecutor.cs
  - Logger.cs
  - SPComparerOutput.cs
  - StoredProcedureComparer.cs
  - StoredProcedureExecutor.cs
  - StoredProcedureOutputComparer.cs
  - Utility.cs

V 4.0

???



THE MOMENT NAME WAS **GIVEN** TO THIS CLASS THE WHOLE THING BECAME **CRYSTAL CLEAR**

GOOD NAMES TELLS THE **CONTEXT** OF ENTIRE SYSTEM

WE **COMMUNICATE**

AVOID **DISINFORMATION**

```
10 var XYZFooBarClassForBlabla
11 var XYZFooBarClassForBlablable
```

```
public class Document
public class DocumentInfo
public class DocumentDetail
public class Collateral
public class Documents
public class DocumentList
```

```
4 var a = l;
5 if ( O == l )
6 a = O1;
7 else
8 l = 01;
9
```

Using lower case **l** (looks like **number**-**1**) and uppercase **O** (looks like number-**0**) are also unhelpful.

"A software author must avoid leaving **false clues** which obscure the meaning of code."
- Ottinger

WE **COMMUNICATE**

USE **PRONOUNCEABLE** NAMES

```
// generation date, year, months, day, hour, minute, and second
class DtaRcrd102 {
        private Date genymdhms;
        /* ... */
}
```

```
// better:
class Customer {
        private Date generationTimestamp;
        /* ... */
}
```

```
@amount float,
@ca3isodev char(3),
@valueDate datetime,
@expiryDate datetime ,
```

CURRENCY

```
public void CalculateSCPUAvailableQuantity(DocumentTransfer transfer)
```

“Programming is a social activity” - Bob Martin

WE **COMMUNICATE**

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 j=0; j < NUMBER_OF_TASKS; j++) {
        int realTaskDays = taskEstimate[j] * realDaysPerIdealDay;
        int realTaskWeeks = (realdays / WORK_DAYS_PER_WEEK);
        sum += realTaskWeeks;
}
```

```
let a = {'apple'  : 2, 'mango' : 1,
      'banana': 0, 'orange': 2,
      'water-melon': 2}

for (b in a) {
   let c = 'no'
   if (a[b] == 2) c = 'many'
   else if (a[b] == 1) c = 'one'
   else if (a[b] == 0) c = 'no'

   console.log(a, 'has', c, c == 'many'?  'seeds' : 'seed')
}
```

```
const NO_SEED = 0
const ONE_SEED = 1;
const MANY_SEEDS = 2;

let fruits = {'apple'  : MANY_SEEDS, 'mango' : ONE_SEED,
      'banana': NO_SEED, 'orange': MANY_SEEDS,
      'water-melon': MANY_SEEDS}

// print the seeds count
for (fruit in fruits) {
   let count = 'no';

   if (fruits[fruit] == MANY_SEEDS) count = 'many'
   else if (fruits[fruit] == ONE_SEED) count = 'one'

   console.log(fruit, 'has', count, count == 'many'? 'seeds' : 'seed')
}
```

Replace literals with constants

CLASS **NAMES**

A CLASS NAME SHOULD
BE A **NOUN**,
NOT A **VERB**.

AVOID **WORDS** like
Manager, Processor, Data, or Info.

GOOD **NAMES** could be:
Tester, TestSuite, TestUnit, HtmlTestResultPublisher.

METHOD NAMES

METHODS SHOULD HAVE **VERB**

```
internal static void Consolidation(Process process,

internal static void UnitDdaNetting(

internal static void UnitAnalyticalOutstandingGeneration(Process process,

public double QualityUnitConversion(int quality, int idUnit, double quantity,
                                    int sourceDocumentIdQuality,
                                    int sourceDocumentIdUnit,
                                    DateTime valueDate)...
```

GOOD NAMES could be:

Save(), Run(), Publish(), Valuate(),
Consolidate()
RunNettingValuation(),
GenerateAnalyticalOutstanding()
ConvertQualityUnit()

WE **COMMUNICATE**

HOWEVER

THERE ARE **NO** **GOOD NAMES**

IT'S CONTINEIOUS PROCESS
EVERY TIME YOU HAVE BETTER
UNDERSTANDING OF CODE,
YOU FIND A BETTER NAME

ENGLAND'S ETON COLLEGE

“THAT’S EASY”

JUST BRUSH
OFF THE DEW
EVERY MORNING

MOW THEM
EVERY
OTHER DAY

AND ROLL
THEM
ONCE A WEEK


DO THAT FOR 500 YEARS &
YOU'LL HAVE A NICE LAWN, TOO


CARE

tl;**dr**

NEGLECTION **ACCELERATES** ROT FASTER THAN ANY OTHER FACTOR

TEST AUTOMATION GIVES YOU
POWER OF
FEARLESS REFACTORING

FIND **OPPORTUNITIES** OF
AUTOMATION IN ABSENCE OF
TESTS FOR OLD CODE

USE THE **GIFTED** POWER OF
NAMING THINGS WISELY

USE **INTENTION-REVEALING** NAMES

AVOID **DISINFORMATION**

USE **PRONOUNCEABLE** NAMES

USE **SEARCHABLE** NAMES

CLASS NAMES MUST BE **NOUNS**

METHOD NAMES MUST BE **VERBS**

REMEMBER WE **DON'T CODE**,
WE **COMMUNICATE**

REMEMBER WE **DON'T CODE**,
WE **COMMUNICATE**

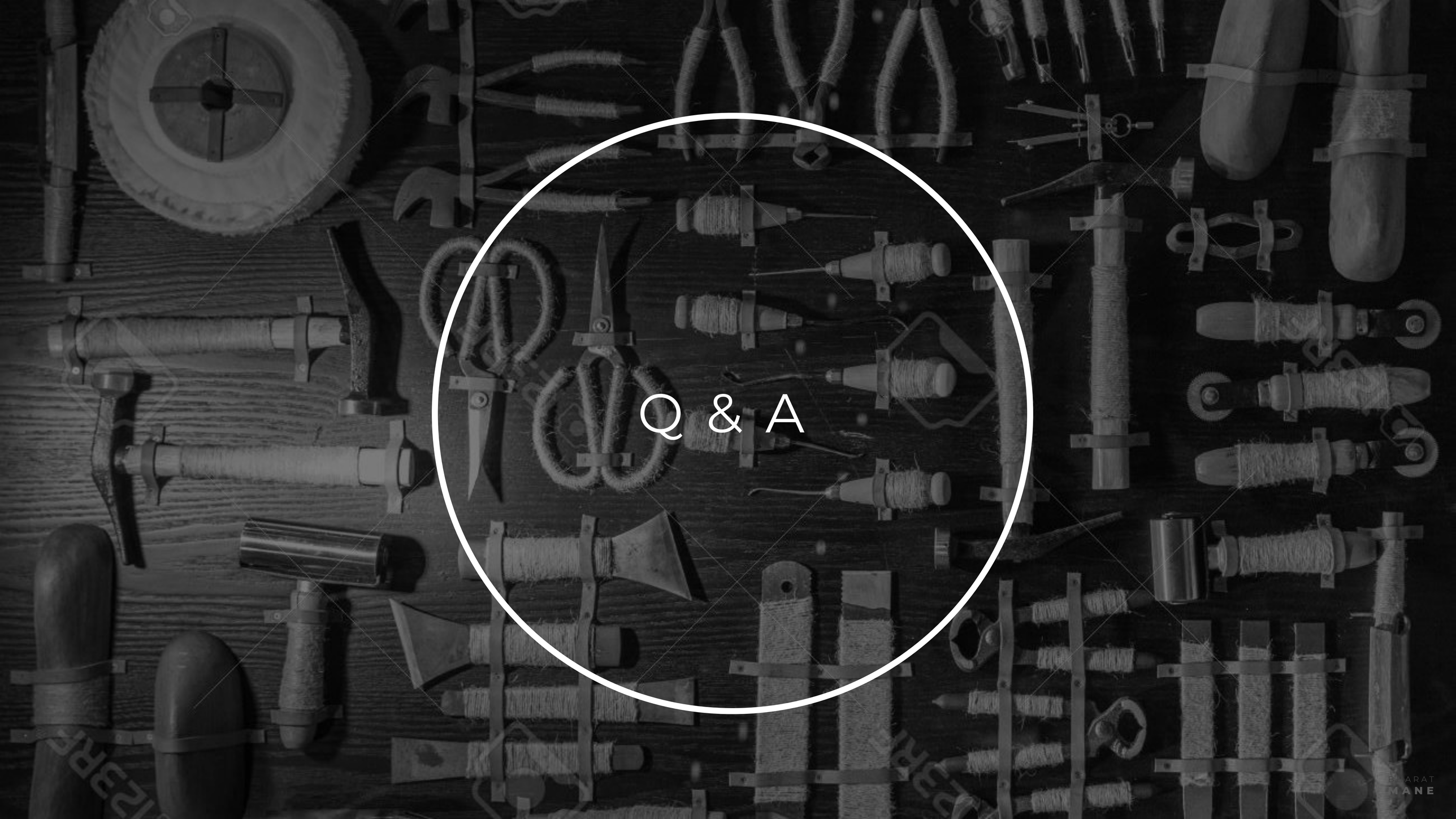
DON’T FEAR

&

TAKE CARE

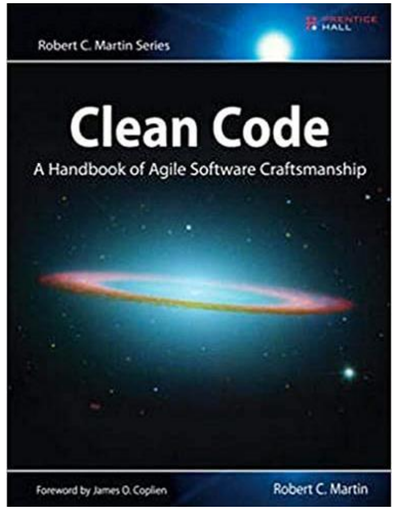
MINDSET

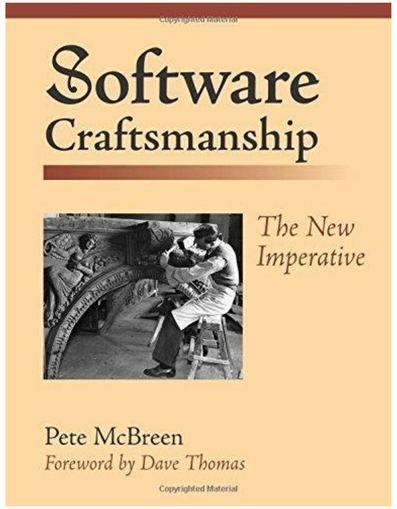
WANT TO DO IT

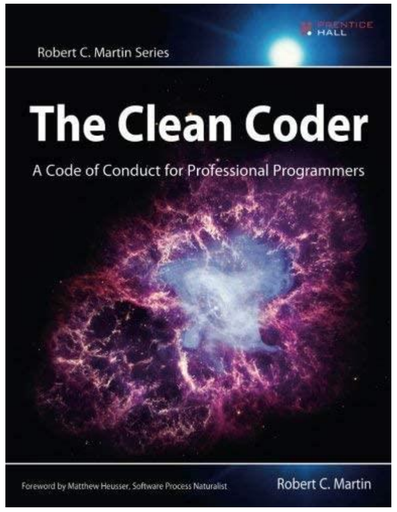

Q & A

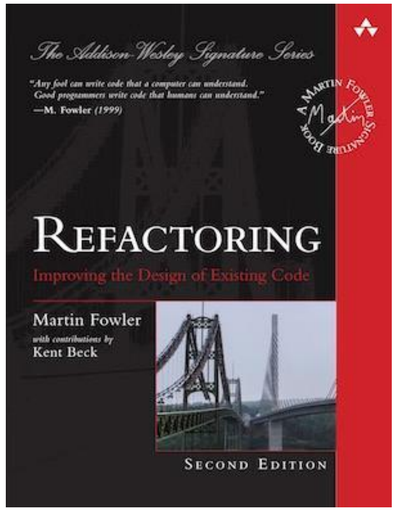
ANNEXTURE

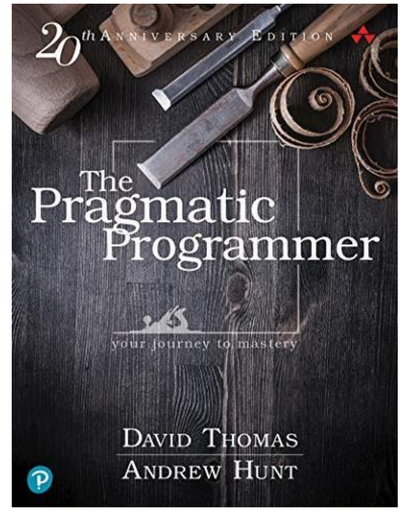
GOOD READS

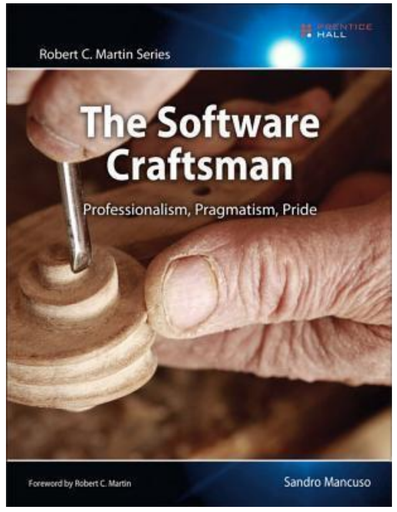


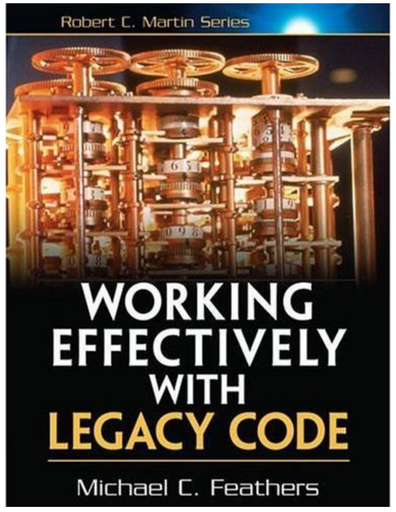


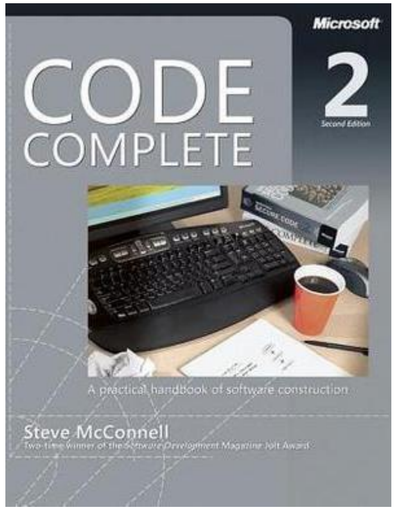


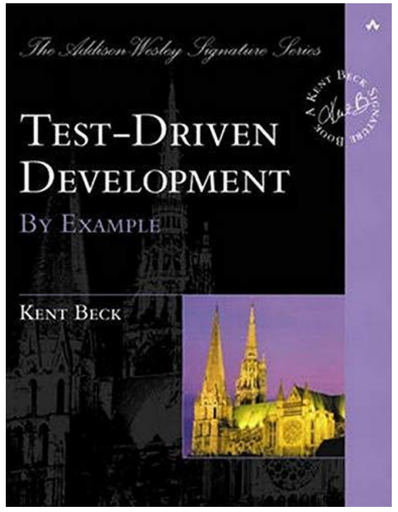


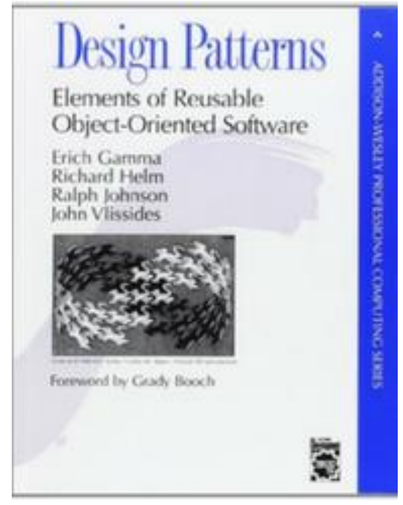




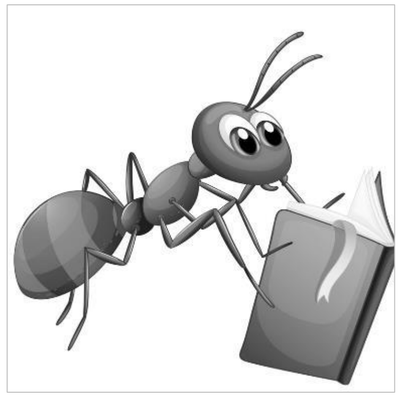








principles
patterns
practices
heuristics



KNOWLEDGE

+

Grind that
knowledge
into muscle
memory

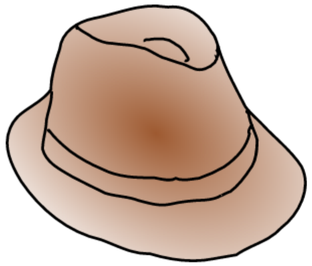
WORK

=

CRAFTSMANSHIP

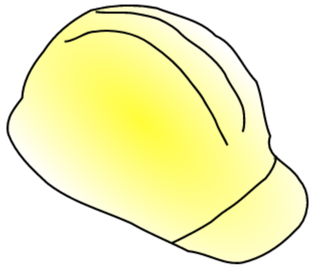
“HONESTY IN SMALL THINGS
IS NOT A SMALL THING.”

-Danish Saw



Refactoring

When refactoring every change you make is a small behavior-preserving change. You only refactor with green tests, and any test failing indicates a mistake. By stringing together a series of small changes like this you can move more quickly and with less risk because you shouldn't get trapped in debugging.



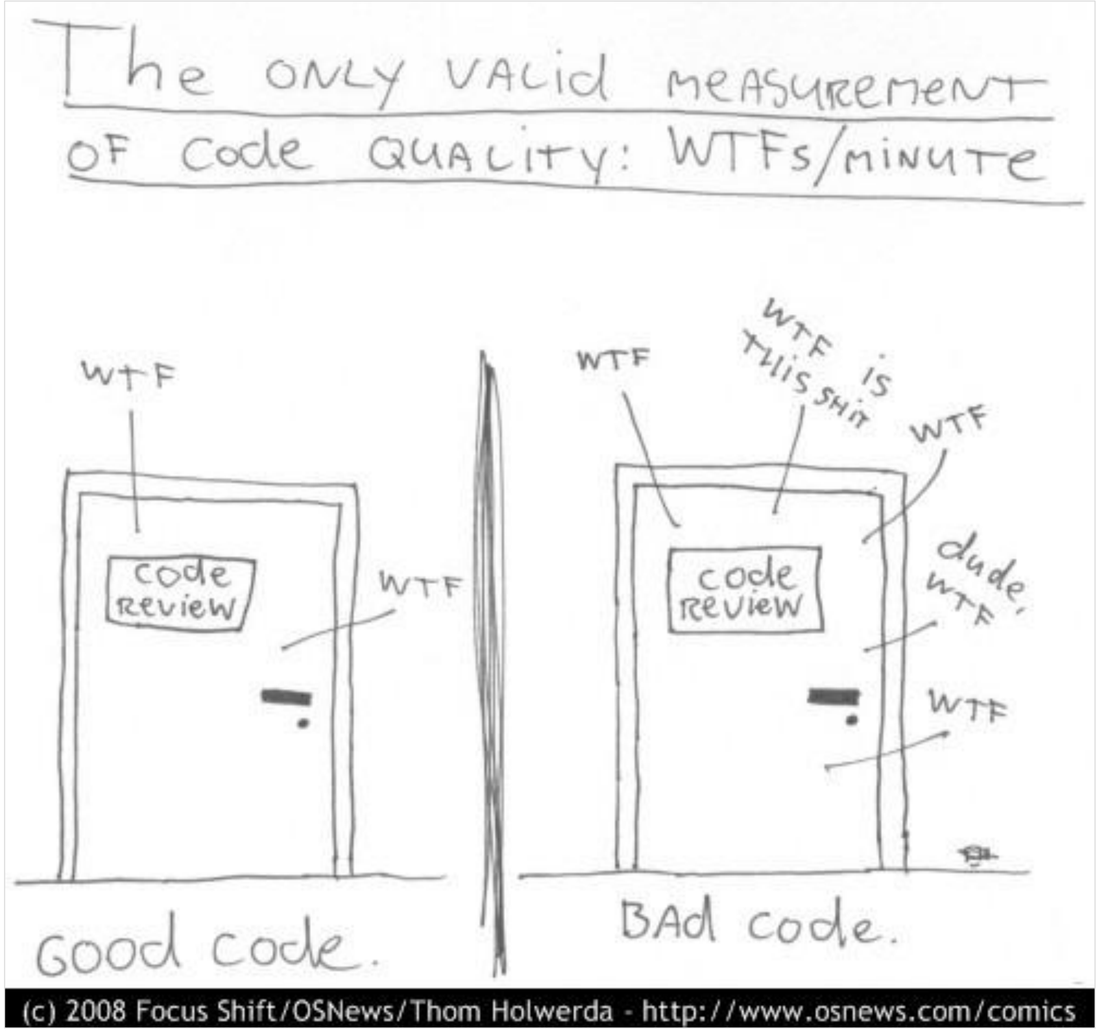
Adding Function

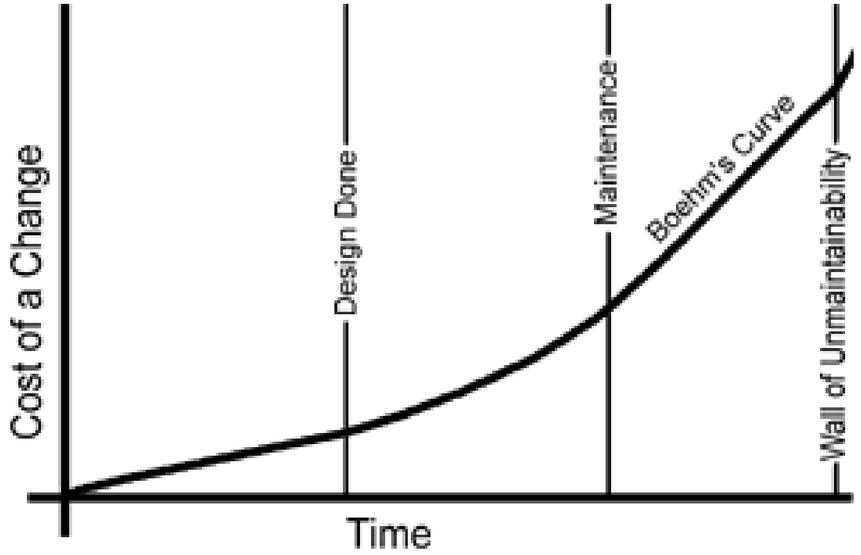
Any other change to the code is adding function. You will add new tests and break existing tests. You aren't confined to behavior-preserving changes (but it's wise to keep changes small and return to green tests swiftly).

During programming you may swap frequently between hats, perhaps every couple of minutes. But...

You can only wear one hat at a time

UNIT OF
CODE QUALITY



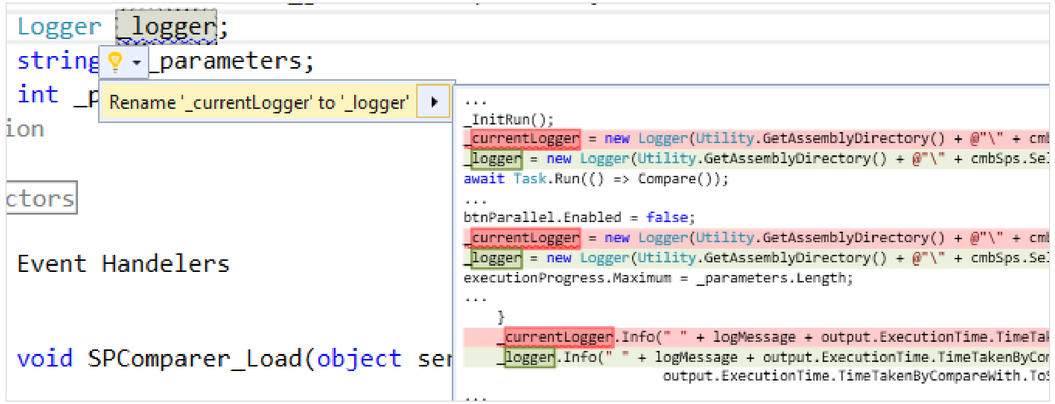


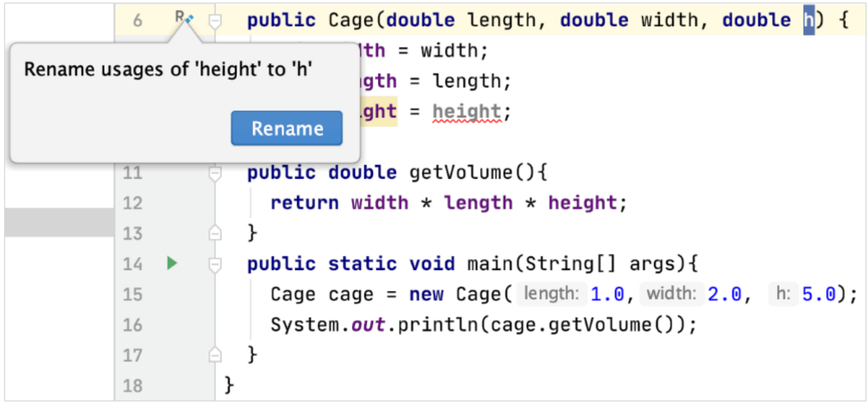
When we proclaim the design is done and accept no more changes

When we move the system into maintenance and change the team's process

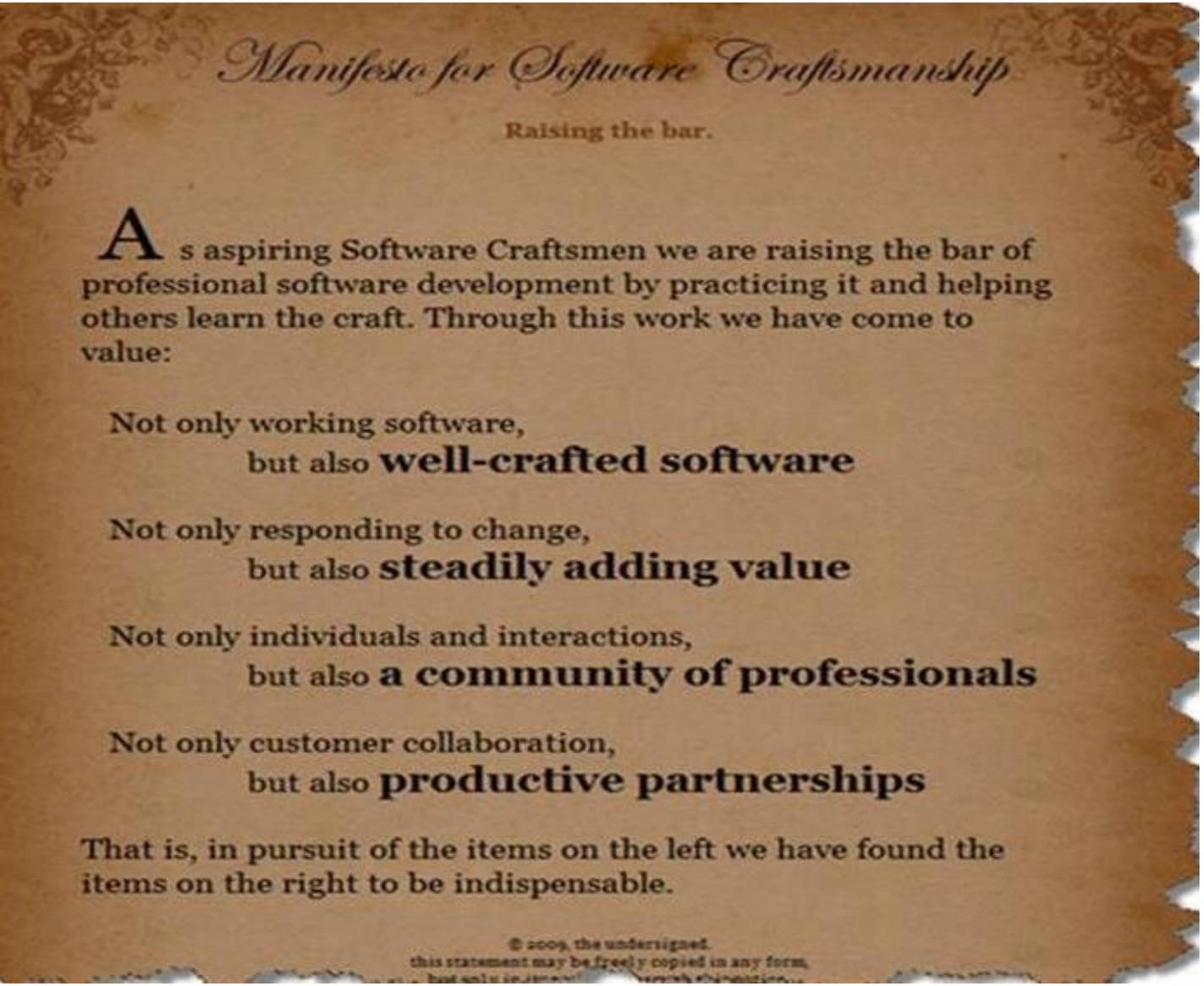
THERE ARE ONLY TWO HARD THINGS IN COMPUTER SCIENCE:
CACHE INVALIDATION AND **NAMING THINGS**

MOST OF THE IDES GIVES YOU VERY
SAFE WAY TO RENAME
VARIABLES, CLASSES, FUNCTIONS
ESPECIALLY PRIVATE STUFF IS
COMPLETELY SAGE





COMPREHENSION REFACTORING



MANIFESTO FOR SOFTWARE CRAFTSMANSHIP

EXTREME PROGRAMMING PRACTICES

Group	Practices
Feedback	✓ Test-Driven Development ✓ The Planning Game ✓ On-site Customer ✓ Pair Programming
Continual Process	✓ Continuous Integration ✓ Code Refactoring ✓ Small Releases
Code understanding	✓ Simple Design ✓ Collective Code Ownership ✓ System Metaphor ✓ Coding Standards
Work conditions	✓ 40-Hour Week

XP has simple rules that are based on **5 values**

COMMUNICATION
Everyone on a team works jointly at every stage of the project

SIMPLICITY
Developers strive to write simple code bringing more value to a product, as it saves time and efforts.

FEEDBACK
Team members deliver software frequently, get feedback about it, and improve a product according to the new requirements.

RESPECT
Every person assigned to a project contributes to a common goal.

COURAGE
Programmers objectively evaluate their own results without making excuses and are always ready to respond to changes.