



agilepartner

Clean code

Agile Mötter - 24 Feb 2011

▴ User story

- ▴ **As a** craftsman
- ▴ **I want to** write **better** code
- ▴ **So that** my fellow developers understand what I meant when I wrote it
- ▴ **And** **we** enjoy reading it

- ▶ Who are we ?
- ▶ What is our role in our organization ?
- ▶ What are our expectations from this seminar ?

- ▲ Agenda
 - ▲ **What** is code? Is it important?
 - ▲ **Why** clean code?
 - ▲ **How** to achieve clean code?

- ▶ We are uncovering better ways of developing software by doing it and helping others do it.
- ▶ Through this work we have come to value:
 - ▶ **Individuals and interactions** over **processes and tools**
 - ▶ **Working software** over **comprehensive documentation**
 - ▶ **Customer collaboration** over **contract negotiation**
 - ▶ **Responding to change** over **following a plan**
- ▶ That is, while there is value in the items on the right, we value the items on the left more.

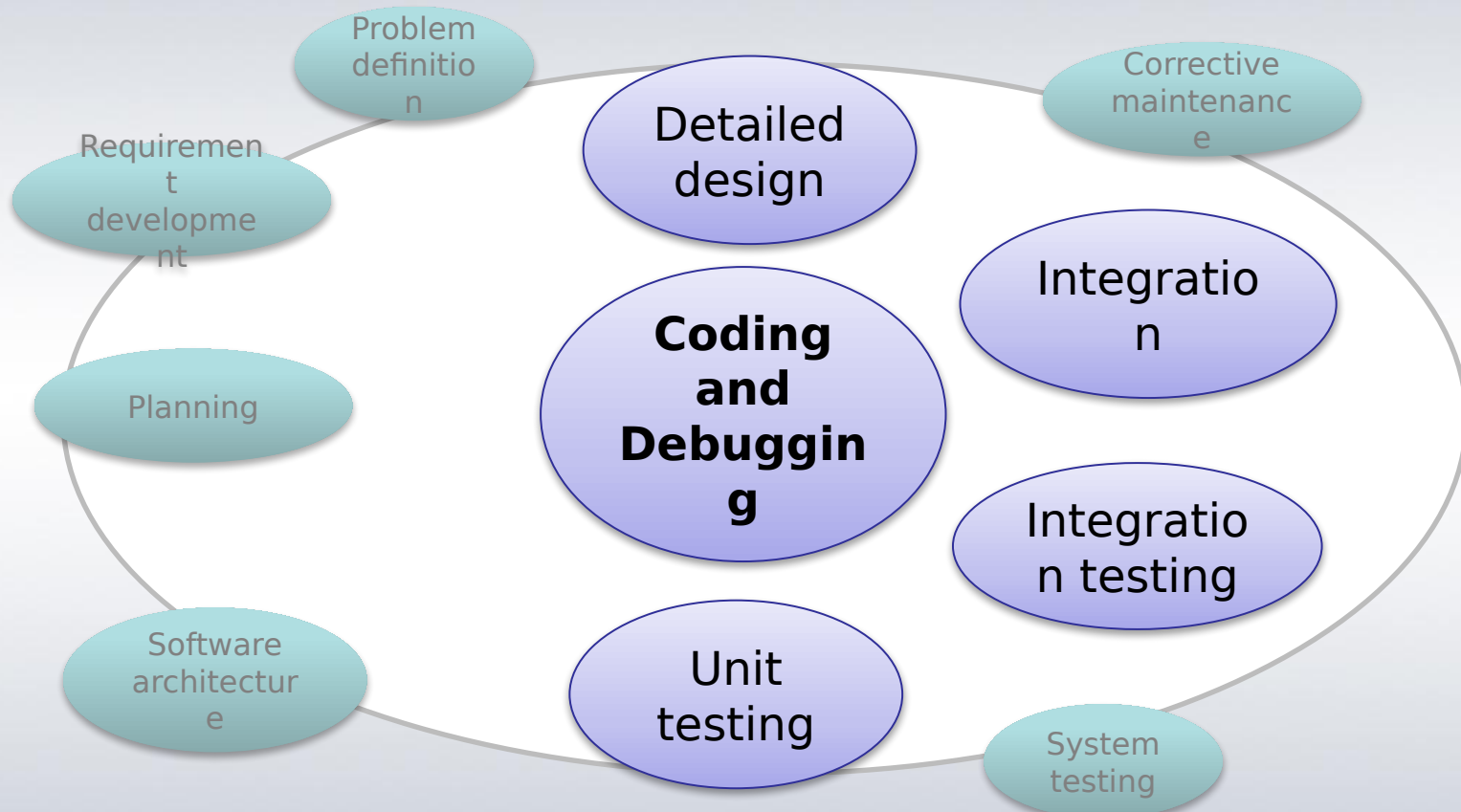
- ▶ As aspiring Software Craftsmen we are raising the bar of professional software development by practicing it and helping others learn the craft.
- ▶ Through this work we have come to value:
 - ▶ Not only **working software**, but also **well-crafted software**
 - ▶ Not only **responding to change**, but also **steadily adding value**
 - ▶ Not only **individuals and interactions**, but also a **community of professionals**
 - ▶ Not only **customer collaboration**, but also **productive partnerships**
- ▶ That is, in pursuit of the items on the left we have found the items on the right to be indispensable.

WHAT IS CODE? IS IT IMPORTANT?

```
10 PRINT "HELLO WORLD"  
20 END
```

- ▶ Source code
- ▶ Coding... and debugging activities!





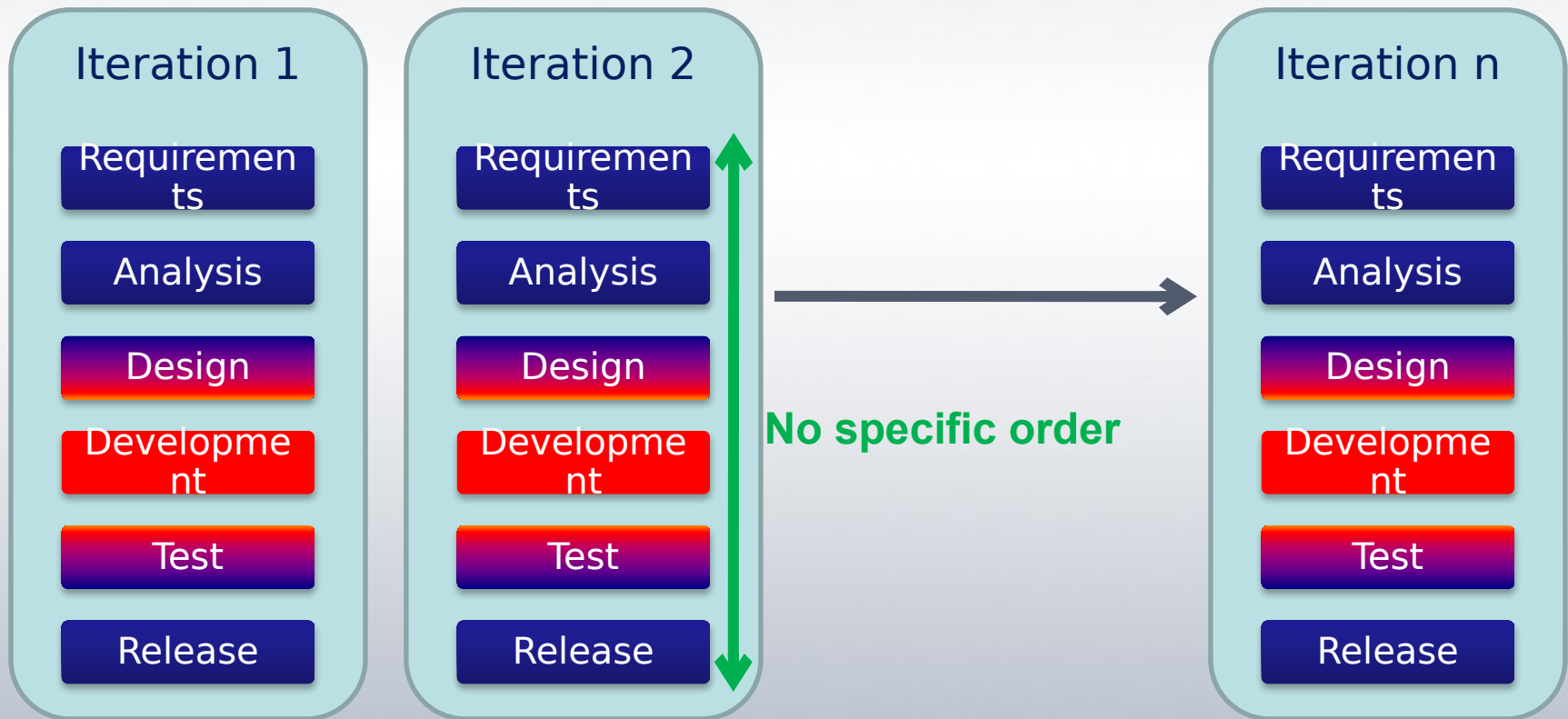
SOFTWARE DEVELOPMENT PROCESS

- Traditional / waterfall approach
 - Coding is part of Design, Development & Test



▀ Agile approach

- ▀ Coding is part of Design, Development & Test **for each iteration**



- ▶ Is code important?
 - ▶ The only part that cannot be skipped
- ▶ A mechanical process? (Code = Encode)
 - ▶ No! The same thing can be coded in a lot of different ways.
 - ▶ Some ways may be better than others
- ▶ Who cares as long as it runs!
 - ▶ But then don't ask for maintenance or additional features...

Who cares about code?!...

- ▶ Programmer: what was hard to write should be hard to read
- ▶ Client: As long as it runs...
- ▶ Studies: not the noble part
- ▶ Manager: Have you seen the deadline?
- ▶ Architect, designer: Code = Encode = Mechanical process

- ▶ Code **is** the working software
- ▶ 50-65% of the effort
- ▶ 50-75% of the errors: Easier to fix but cost a lot not to fix
- ▶ Must be done

- ▶ Software development 1st metaphor: Letter-writing
 - ▶ « Plan to throw one away, you will anyhow » (Fred Brooks - The Mythical Man-Month - 1995)
 - ▶ ... Unacceptable !

- ▶ Cleaning ?!
 - ▶ Cleaning is an iterative process (Boy camp rule)
 - ▶ Code smell
 - ▶

- ▶ Craftsmanship : idea of quality in the work

- ▶ A better code $\Leftrightarrow$ A better programmer
- ▶ We are professional !

- ▶ Construction metaphor

WHY CLEAN CODE ?

```

    }
    if () {
        for () {
        }
    }
    if () {
        for () {
        }
    }
    if () {
        for () {
        }
    }
} catch () {
}
if () {
    for () {
        try {
            if ()
        } catch () {
            try {
            } catch () {
            }
        }
    }
}
}
}
try {
} catch () {
}
if () {
    if () {
        for () {
            if () {
            } else {
                if ()
            } else
        }
    }
    if () {
        try {
        } catch () {
        }
    }
}
}
}
}
}
if ()
try {
} catch () {
}
if () {
    if () {
        for () {
            if () {
                try {
                } catch () {
                }
            }
        }
    }
}
}
}
if ()
return ();
}
} finally {
}
}

```

```

<input onclick="var flag = true;
flag = validateRecord(form.res_Addr_1,form.res_City_1,form.res_County_1, form.res_State, form.res_Zip);

if (flag == true)
{
    // validate start Date and End Date
    var dateflag = true;
    var day = "";
    dateflag = validate(form.res_MM, day,form.res_YYYY);
    if (dateflag == true)
    {
        dateflag = validate(form.res_MM_end, day, form.res_YYYY_end);
        if (dateflag == true)
        {
            dateflag = ComparedDate(form.res_MM,"01",form.res_YYYY, form.res_MM_end,"01", form.res_YYYY_end);
            if (dateflag == true)
            {
                // check to see if it is a new record or existing record
                if (editResRecord == -1)
                { // new record

                    // add the year values in the array, so we can use the value later on to see if resident information contains records up to 7 years
                    stYear.push(parseInt(form.res_YYYY.value)); // start year
                    // add record in the text area
                    pushOn(RecordArray, MsgArray,form.DisplayRecord);
                    ClearResidentField();
                    form.totalResRecord.value = RecordArray.length;
                    form.res_Addr_1.focus();
                }
            }
            else
            { // existing record that are being edited
                stYear[editResRecord] = form.res_YYYY.value

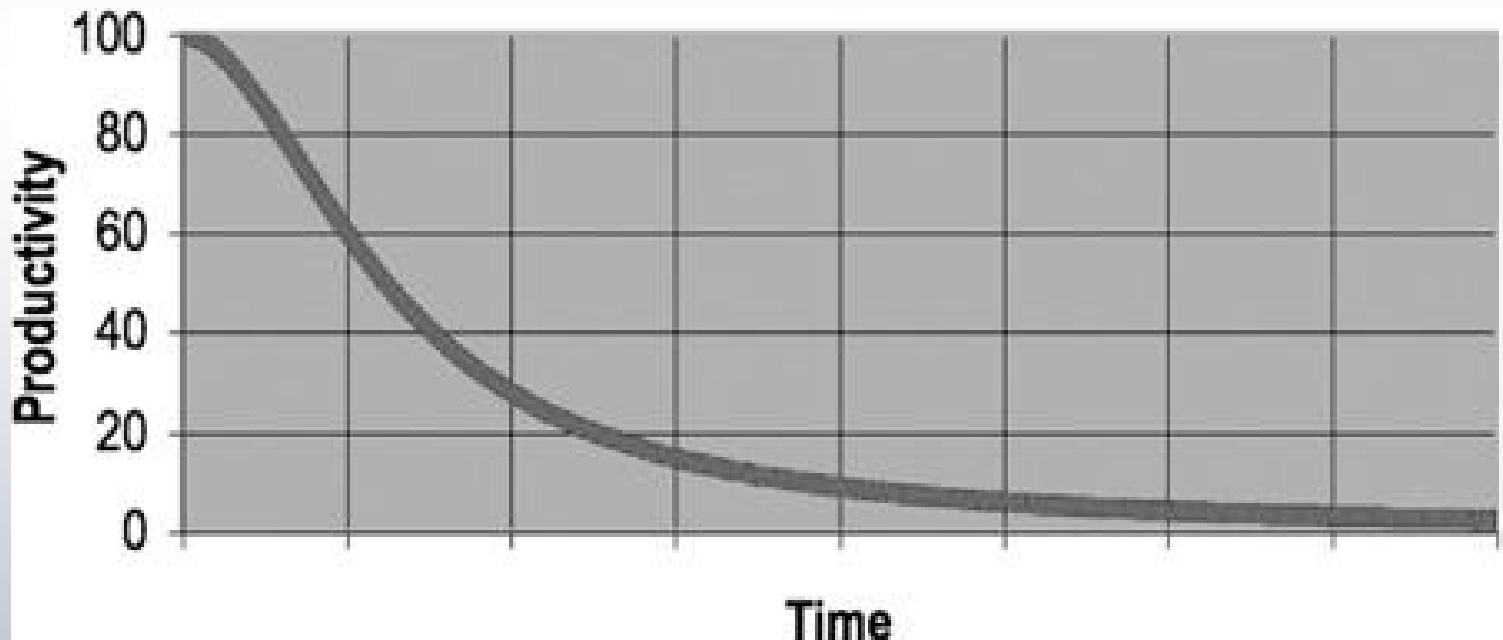
                //1) Loop through each array value and add all it all together as a record
                var FinalMessage = "";
                FieldInfo("ADDRESS:" , form.res_Addr_1.value, 0, "1");
                FieldInfo("CITY:" , form.res_City_1.value, 1, "1");
                FieldInfo("COUNTY:" , form.res_County_1.value, 4, "1");
                var i = form.res_State.selectedIndex;
                FieldInfo("STATE:" , form.res_State.options[i].value, 2, "1");
                FieldInfo("ZIP:" , form.res_Zip.value, 3, "1");
                FieldInfo("FROM:" , form.res_MM.value, 5, "1"); //start month
                FieldInfo("Y" , form.res_YYYY.value, 6, "1"); //start year
                FieldInfo("TO:" , form.res_MM_end.value, 7, "1"); // end month
                FieldInfo("Y" , form.res_YYYY_end.value, 8, "1"); // end year

                for (var i= 0; i < MsgArray.length; i++)
                {
                    if (MsgArray[i] != null)
                    {
                        FinalMessage = FinalMessage + MsgArray[i];
                        MsgArray[i] = "";
                    } // end if
                } // end for

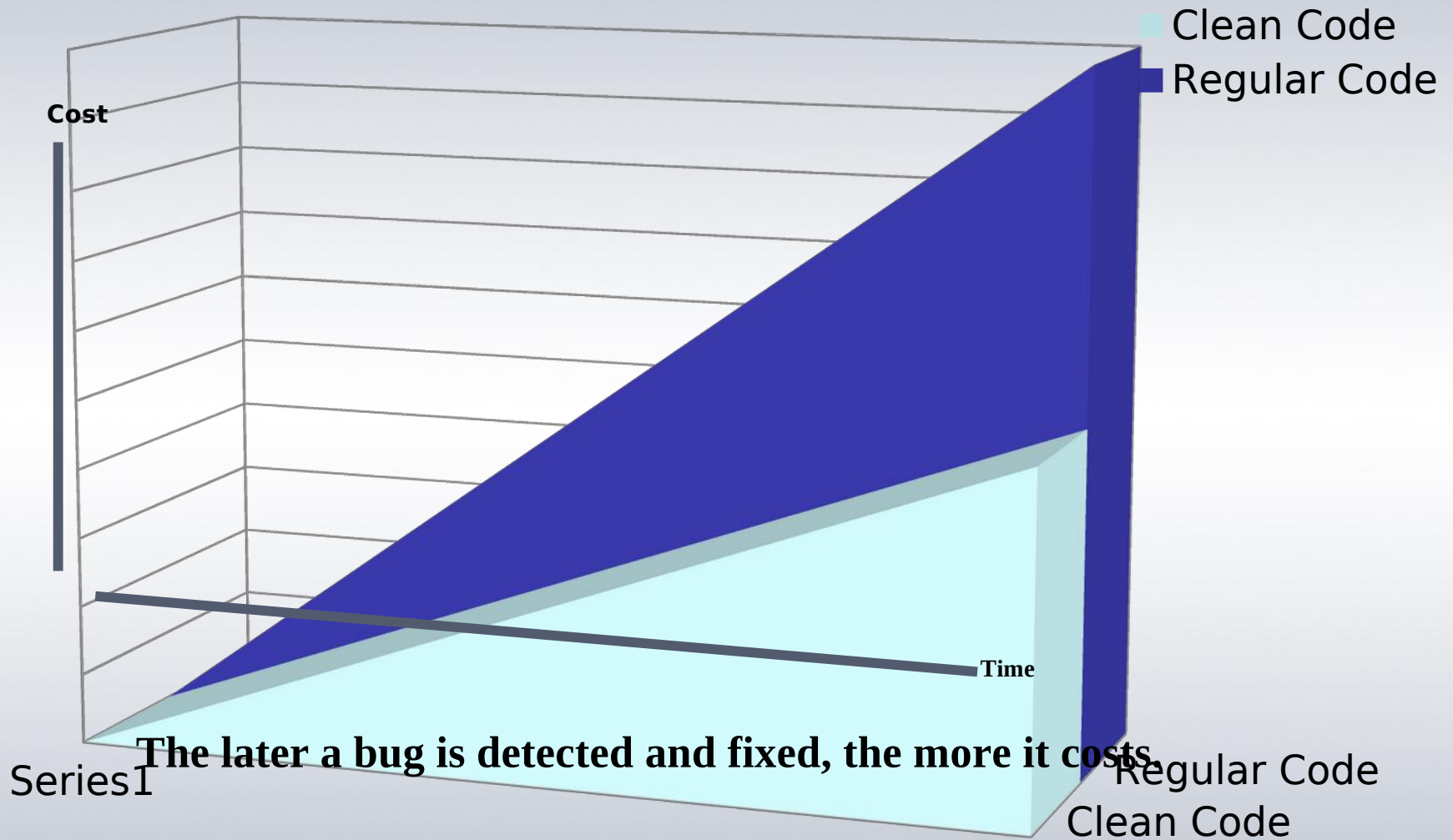
                RecordArray[editResRecord] = FinalMessage;
                DispArray (RecordArray, DisplayRecord)
                ClearResidentField();
                form.res_Addr_1.focus();
                editResRecord = -1 // change the edit record flag back to false
                form.totalResRecord.value = RecordArray.length;
            }
        }
    } // end if
} // end if
} // end if
" type="button" value="Add / Update Record" />

```

- As the mess builds, the productivity of the team continues to decrease, asymptotically approaching zero. As productivity decreases, management does the only thing they can; they add more staff to the project in hopes of increasing productivity. [...] they, and everyone else on the team, are under horrific pressure to increase productivity. So they all make more and more messes, driving the productivity ever further toward zero.
Clean Code, Robert C. Martin



- ▶ Defects can be introduced during
 - ▶ Requirements
 - ▶ Architecture
 - ▶ Design & coding
- ▶ Clean code
 - ▶ Less bug introduced during design & coding
 - ▶ All kind of bugs cost less to fix



- ▶ Unreadable
- ▶ Not easy to understand
- ▶ Classes are tightly coupled
- ▶ Code smells
 - ▶ Duplicated code
 - ▶ Long methods
 - ▶ Too many arguments
 - ▶ Switch statements
 - ▶ God class

- ▶ Our code base **is** the working software
 - ▶ It is often the only faithful description/documentation of the software
- ▶ Modeling and designing means nothing without implementing code
 - ▶ A beautiful model on paper is no use to the end user
- ▶ The quality of the code impacts the overall quality of the software

- ▴ Readability
- ▴ Understandability
- ▴ Testability
- ▴ Flexibility
- ▴ Modularity
- ▴ Evolutivity
- ▴ Reusability
- ▴ Maintainability

BUZZ

HOW TO ACHIEVE CLEAN CODE ?

- ▶ SRP : Single responsibility principle
- ▶ KISS : Keep it simple, Stupid!
- ▶ YAGNI : You ain't gonna need it
- ▶ DRY / DIE : Don't repeat yourself / Duplication is evil
- ▶ LoD : Law of Demeter (Don't talk to strangers)
- ▶ LC : Loose coupling (Don't wanna know too much about others)
- ▶ POLS : Principle of Least Surprise

- ▴ Meaningful names
- ▴ Functions
- ▴ Comments
- ▴ Formatting
- ▴ Objects and data structures
- ▴ Classes
- ▴ Error handling

- ▶ Intention revealing names
- ▶ Avoid disinformation
- ▶ Make meaningful distinctions
- ▶ Use pronouncable names
- ▶ Use searchable names
- ▶ Avoid encodings
 - ▶ Hungarian notation
 - ▶ Member prefix
- ▶ Avoid mental mappings

- ▶ Class names
- ▶ Method names
- ▶ Don't be cute
- ▶ Pick one word per concept
- ▶ Don't pun
- ▶ Use solution domain names
- ▶ Use problem domain names
- ▶ Add meaningful context
- ▶ Don't add gratuitous context

- ▶ Small
- ▶ Do one thing
- ▶ One level of abstraction per function
- ▶ Switch statement
- ▶ Use descriptive names
- ▶ Function arguments
- ▶ Have no side effects
- ▶ Command query separation

- ▶ Prefer exceptions to returning error codes
- ▶ Don't repeat yourself
- ▶ Structured programming

- Comments do not make up for bad code
- Good comments
- Bad comments

```
//This and that  
This()  
That()
```

```
300 LDA #0      A9 00  
302 STA $334    8D 34 03  
305 STA $335    8D 35 03  
308 STA $337    8D 37 03  
30B LDY $338    AC 38 03 multiplier  
30E LDX #8      A2 08 counter  
310 TYA         98  
311 LSR         4A  
313 TAY         48  
314 BCC $328    90 13  
315 CLC         18  
316 LDA $334    AD 34 03  
319 ADC $336    6D 36 03 add  
31C STA $334    8D 34 03 multiplicand  
31F LDA $335    AD 35 03 to product  
322 ADC $337    6D 37 03  
325 STA $335    8D 35 03  
328 CLC         18  
329 ROL $336    2E 36 03 shift  
32C ROL $337    2E 37 03 multiplicand  
32F DEX         CA  
330 BNE $310    D0 E0  
332 RTS         60  
      $33 bytes  
product  $334, $335  
multiplicand $336, $337  
multiplier $338  
  
8 x 8 Multiplication Routine
```

- ▴ Vertical formatting
 - ▴ Density
 - ▴ Vertical distance
- ▴ Horizontal formatting
 - ▴ Keep your lines short
 - ▴ Indentation
- ▴ One convention for the team

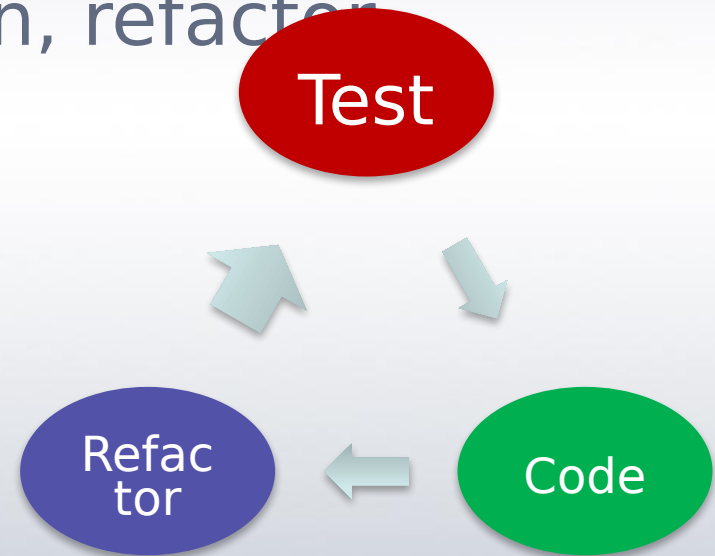
- ▴ High level of abstraction
 - ▴ Expose behavior
 - ▴ Hide data
- ▴ Law of Demeter
- ▴ Data Transfer Objects

- ▶ Class organization
 - ▶ Members order
 - ▶ Encapsulation
- ▶ Should be small
 - ▶ Single Responsibility Principle
 - ▶ Consistent abstraction
 - ▶ Cohesion
 - ▶ Many small classes

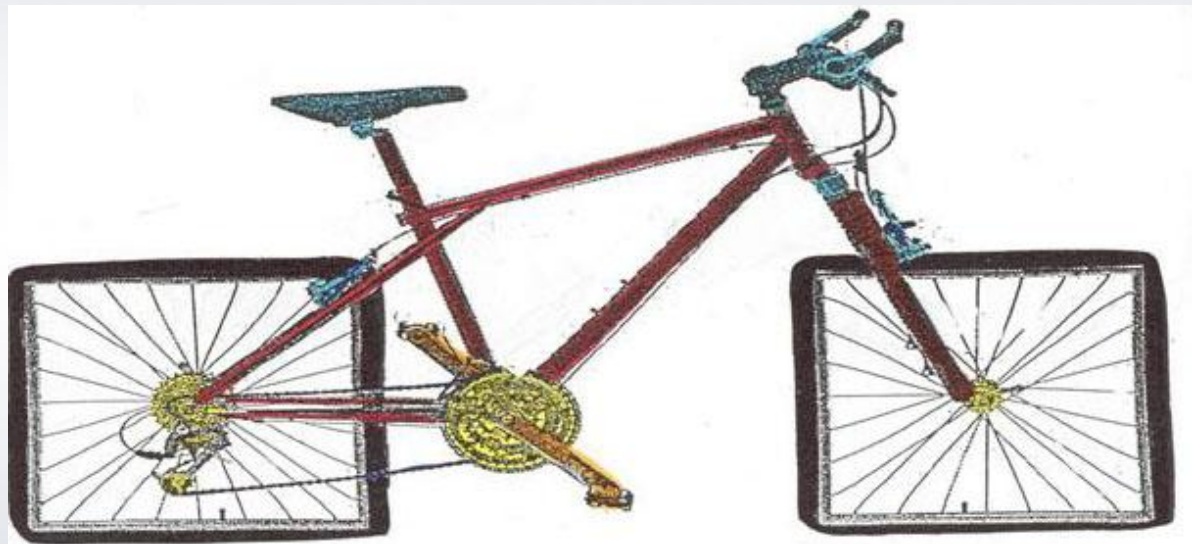
- ▶ Reduce coupling
 - ▶ Use an interface to decouple classes from each other
 - ▶ Avoid bidirectional dependencies
- ▶ Restrict visibility
 - ▶ Make things as « private » as possible
- ▶ Avoid semantic coupling (time dependent)
- ▶ Avoid deep inheritance hierarchies

- ▶ Use exception rather than return code
- ▶ Don't swallow exceptions
- ▶ Use finally block to clean up
- ▶ Provide context when throwing exception
- ▶ Exceptions should stay exceptionnal

- ▴ Successive refinement
- ▴ Refactoring
 - ▴ Make it work, then make it right
 - ▴ TDD mantra : red, green, refactor



- ▶ Do not reinvent the wheel
 - ▶ Use what's already out there



- ▶ Use design patterns, but don't force them

- ▶ Manager code complexity
- ▶ Refactor ruthlessly
- ▶ Tests are code and should be cleaned up like production code
- ▶ Your test harness is your safety net
- ▶ Direct benefit : design emergence
- ▶ Don't be limited by your language : think about the **What**, not the **How**

- ▶ <http://butunclebob.com/ArticleS.UncleBob.GreenWristBand>



- ▶ Clean code : A Handbook of Agile Software Cratfsmanship
by Robert C. Martin (Prentice Hall)
- ▶ Code complete : A practical handbook of software construction by Steve McConnell (Microsoft Press)
- ▶ Implementation Patterns by Kent Beck (Addison-Wesley – Kent Beck Signature)
- ▶ Refactoring: Improving the Design of Existing Code by Martin Fowler (Addison-Wesley)

