SiliconValley
codecamp_11
Oct 8th & 9th, 2011

Silicon Valley
Code Camp

Clean Code
Why Clean Code matters

Foothill College, October 9nd 2011

Theo Jungeblut

- Senior Software Developer at Omnicell Inc. in Mountain View
- Has been designing and implementing .NET based applications , components and frameworks for more than 8 years
- Previously worked in factory automation with focus on component based software and framework development for 3 ½ years
- Degree in Software Engineering and Network Communications

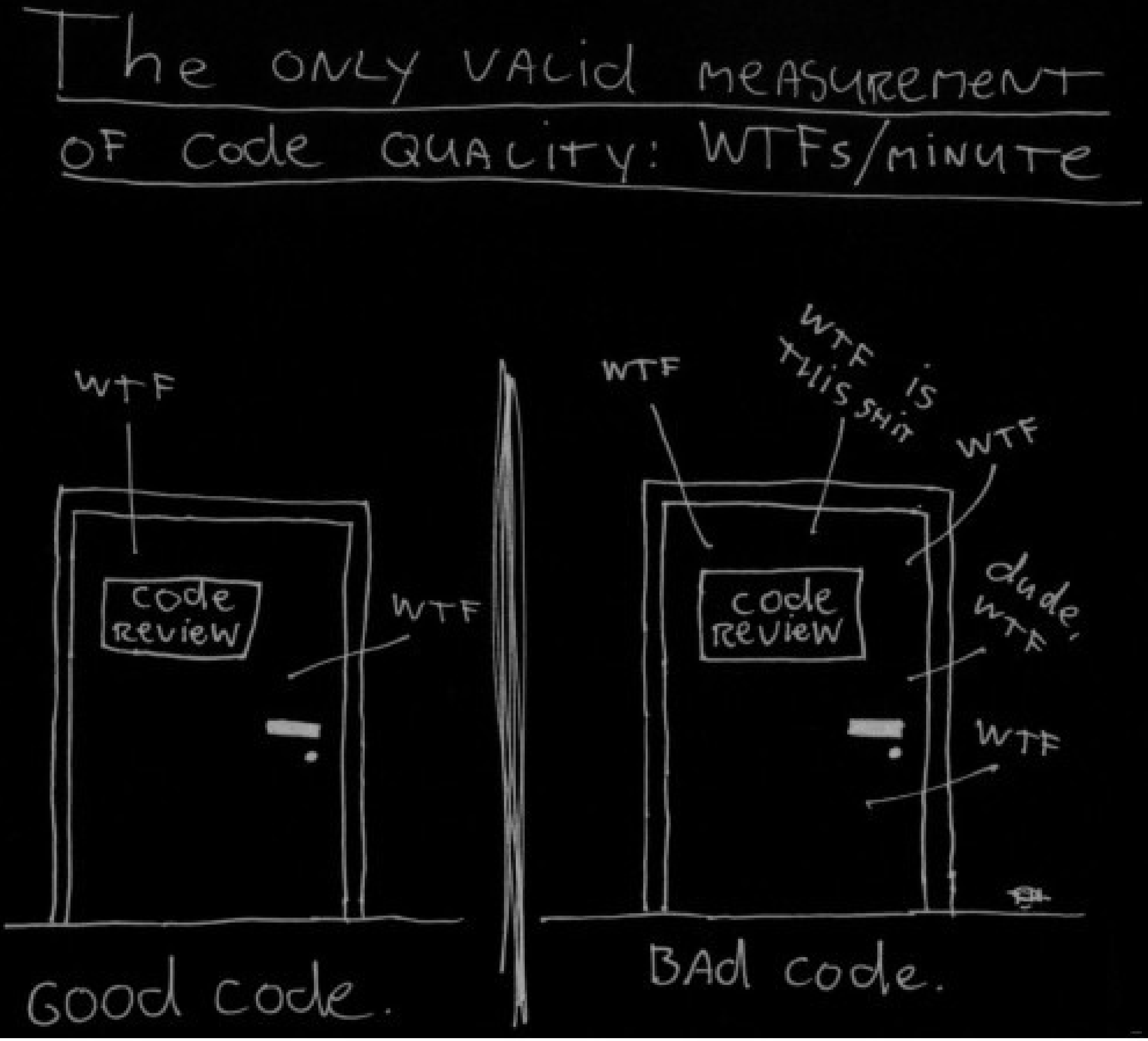
theo@csharp-lighthouse.com

www.csharp-lighthouse.com

Overview

- Why Clean Code
- Tools
    - Resharper
    - FxCop, StyleCop & StyleCop plugin for Resharper
    - GhostDoc & Spell Checker
    - Code Contracts, Pex & Moles
- Clean Code Developer Initiative
- Principles and Practices
- Code Comparison
- Q&A

Does writing Clean Code make us more efficient?



(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

What is Clean Code?

Clean Code is maintainable

Source code must be:

- readable & well structured
- extensible
- testable

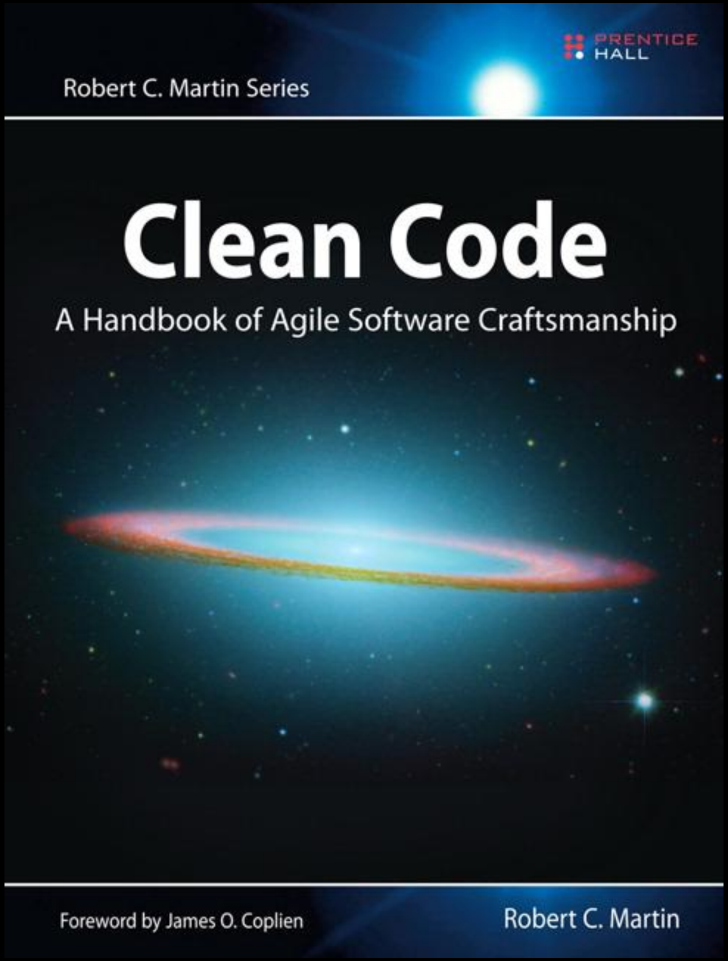
Software Engineering vs. Craftsmanship

The "Must Read"-Book(s)

by Robert C Martin

A Handbook of Agile Software Craftsmanship

"Even bad code can function. But if code isn't clean, it can bring a development organization to its knees."


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

Code Maintainability *

Principles	Patterns	Containers
Why?	How?	What?
Extensibility	Clean Code	Tool reuse

* from: Mark Seemann’s “Dependency Injection in .NET” presentation Bay.NET 05/2011

.NET Tools and their Impact

Tool name	Positive Impact	Negative Impact
Resharper	compiling ++++	VS responsiveness --
FxCop	code quality ++	compiling time -
StyleCop	code consistency +++	compiling time -
StyleCop plugin for Resharper	compiling time +++	VS responsiveness --
Ghost Doc	automated docs	potentially worse doc
Spell Checker	fewer spelling errors ++	performance --
Code Contracts	testability, quality ++	compiling time --
Pex & Moles	automated test ++	compiling time --

Resharper

“The single most impacting development addition to Visual Studio”

Features:

- Code Analysis
- Quick Fixes
- Code Templates
- Code Generation
- Code Cleanup
- Many, many more…

http://www.jetbrains.com/resharper/

FxCop / Static Code Analysis

Code Analysis:

- Correctness
- Library design
- Internationalization and localization
- Naming conventions
- Performance
- Security

http://msdn.microsoft.com/en-us/library/3z0aeatx.aspx

Style Cop with R# Integration

Code Consistency & Readability:

- Automated check of C# coding standard
- Enforceable at check-in with TFS check-in Policy

- Full Integration in Resharper with Style Cop plugin:
- Code Analysis
- Quick Fixes
- Code Cleanup

Ghost Doc

- Save keystrokes and time
- Simplify documenting your code
- Benefit of the base class documentation

http://submain.com/products/ghostdoc.aspx

Spell Checker

- Spelll chicking for literals and comments in VS

http://visualstudiogallery.msdn.microsoft.com/7c8341f1-ebac-40c8-92c2-476db8d523ce/

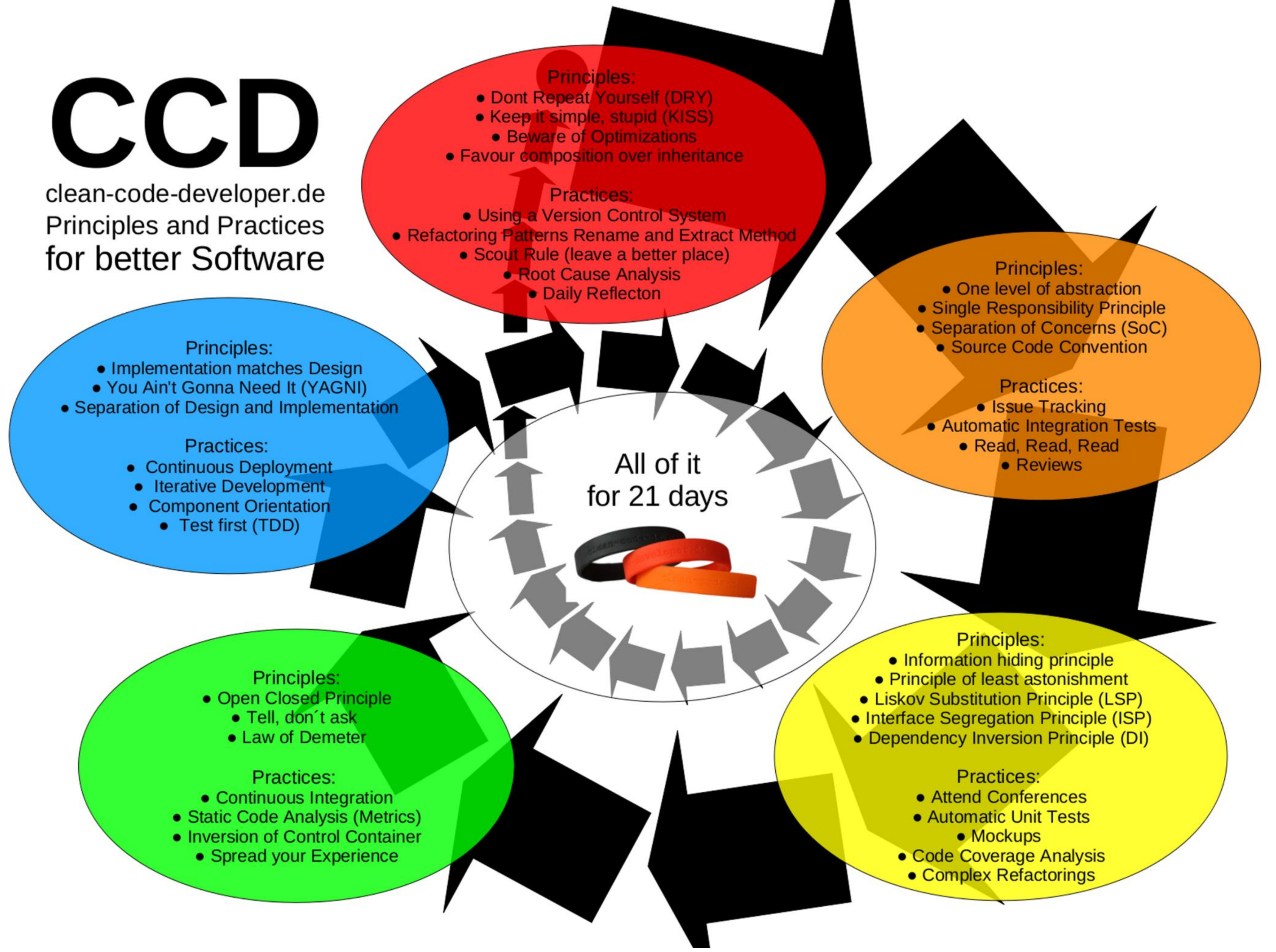
- Design-by-Contract programming
- Improved testability
- Static verification
- API documentation integration with Sandcastle

http://msdn.microsoft.com/en-us/devlabs/dd491992

Microsoft Pex & Moles

- Pex automatically generates test suites with high code coverage.
- Moles allows to replace any .NET method with a delegate.

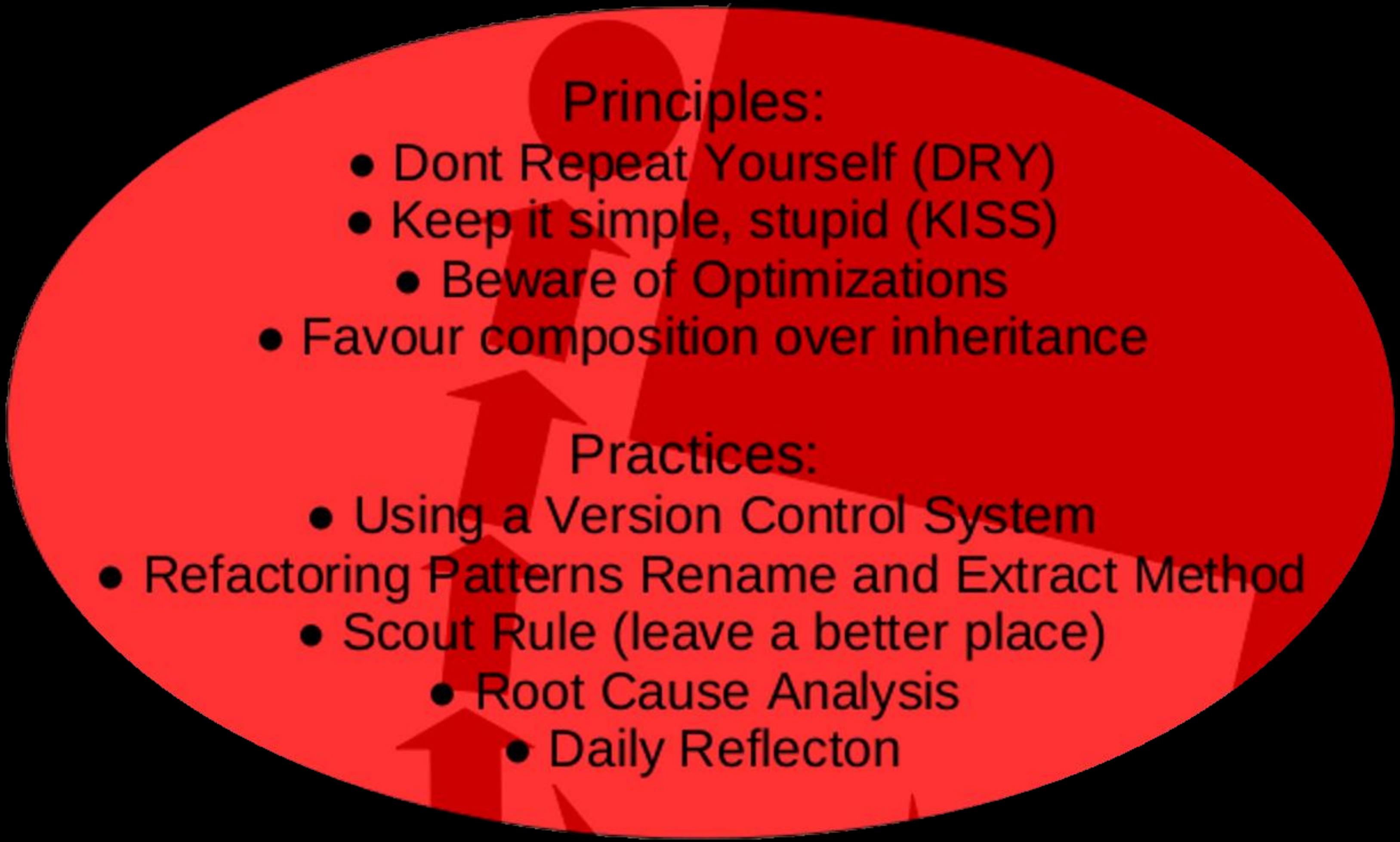
http://research.microsoft.com/en-us/projects/pex/


CCD
clean-code-developer.de
Principles and Practices
for better Software
Principles:
• Dont Repeat Yourself (DRY)
• Keep it simple, stupid (KISS)
• Beware of Optimizations
• Favour composition over inheritance
Practices:
• Using a Version Control System
• Refactoring Patterns Rename and Extract Method
• Scout Rule (leave a better place)
• Root Cause Analysis
• Daily Reflecton
Principles:
• One level of abstraction
• Single Responsibility Principle
• Separation of Concerns (SoC)
• Source Code Convention
Practices:
• Issue Tracking
• Automatic Integration Tests
• Read, Read, Read
• Reviews
Principles:
• Information hiding principle
• Principle of least astonishment
• Liskov Substitution Principle (LSP)
• Interface Segregation Principle (ISP)
• Dependency Inversion Principle (DI)
Practices:
• Attend Conferences
• Automatic Unit Tests
• Mockups
• Code Coverage Analysis
• Complex Refactorings
Principles:
• Open Closed Principle
• Tell, don´t ask
• Law of Demeter
Practices:
• Continuous Integration
• Static Code Analysis (Metrics)
• Inversion of Control Container
• Spread your Experience
Principles:
• Implementation matches Design
• You Ain't Gonna Need It (YAGNI)
• Separation of Design and Implementation
Practices:
• Continuous Deployment
• Iterative Development
• Component Orientation
• Test first (TDD)
All of it
for 21 days

Graphic by Michael Hönnig http://michael.hoennig.de/2009/08/08/clean-code-developer-ccd/

Clean Code Developer – 1st Iteration

by Ralf Westphal & Stefan Lieser – http://www.clean-code-developer.de



Principles:

- Dont Repeat Yourself (DRY)
- Keep it simple, stupid (KISS)
- Beware of Optimizations
- Favour composition over inheritance

Practices:

- Using a Version Control System
- Refactoring Patterns Rename and Extract Method
- Scout Rule (leave a better place)
- Root Cause Analysis
- Daily Reflecton

Graphic by Michael Hönnig http://michael.hoennig.de/2009/08/08/clean-code-developer-ccd/

Keep it simple, stupid (KISS)

KISS-Principle – “Keep It Simple Stupid”

by Kelly Johnson

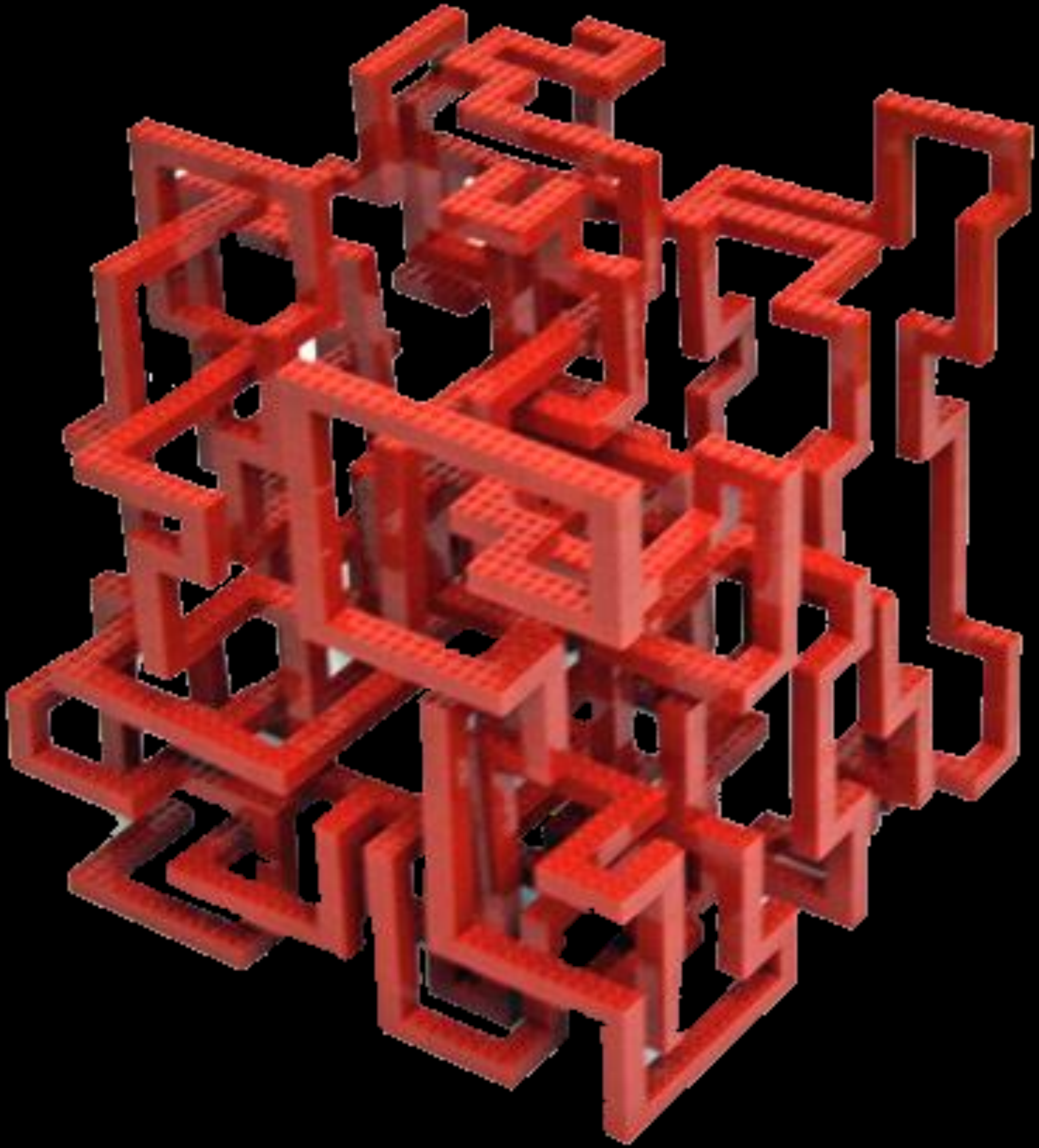
http://blogs.smarter.com/blogs/Lego%20Brick.jpg

The Power of Simplicity

Graphic by Nathan Sawaya courtesy of brickartist.com

Graphic by Nathan Sawaya courtesy of brickartist.com

http://www.geekalerts.com/lego-iphone/



Graphic by Nathan Sawaya courtesy of brickartist.com

Don’t repeat yourself (DRY)

Don't repeat yourself (DRY)

by Andy Hunt and Dave Thomas in their book "The Pragmatic Programmer"

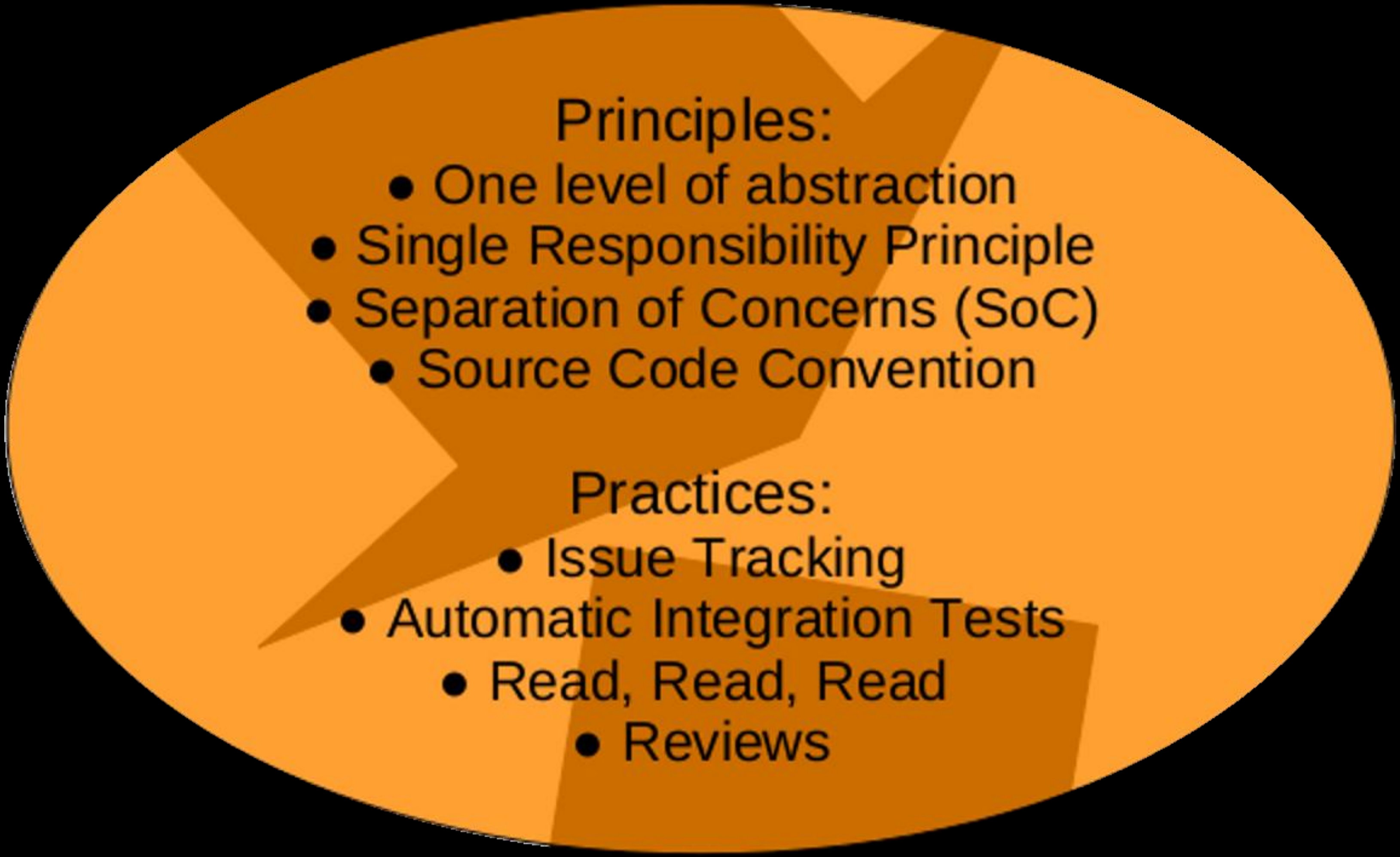
```
// Code Copy and Paste Method
public Class Person
 {
   public string FirstName { get;  set;}
   public string LastName { get; set;}

   public Person(Person person)
   {
      this.FirstName = string.IsNullOrEmpty(person.FirstName)
                   ? string.Empty : (string) person.FirstName.Clone();

      this.LastName = string.IsNullOrEmpty(person.LastName)
                   ? string.Empty : (string) person.LastName.Clone();
   }

   public object Clone()
   {
      return new Person(this);
   }
}
```

```
// DRY Method
public Class Person
 {
   public string FirstName { get;  set;}
   public string LastName { get; set;}

   public Person(Person person)
   {
      this.FirstName = person.FirstName.CloneSecured();
      this.LastName = person.LastName.CloneSecured();
   }

   public object Clone()
   {
      return new Person(this);
   }
}
```

```
public static class StringExtension
{
   public static string CloneSecured(this string original)
   {
     return string.IsNullOrEmpty(original) ? string.Empty : (string)original.Clone();
   }
}
```

Clean Code Developer – 2nd Iteration

by Ralf Westphal & Stefan Lieser – http://www.clean-code-developer.de



Principles:

- One level of abstraction
- Single Responsibility Principle
- Separation of Concerns (SoC)
- Source Code Convention

Practices:

- Issue Tracking
- Automatic Integration Tests
- Read, Read, Read
- Reviews

Graphic by Michael Hönnig http://michael.hoennig.de/2009/08/08/clean-code-developer-ccd/

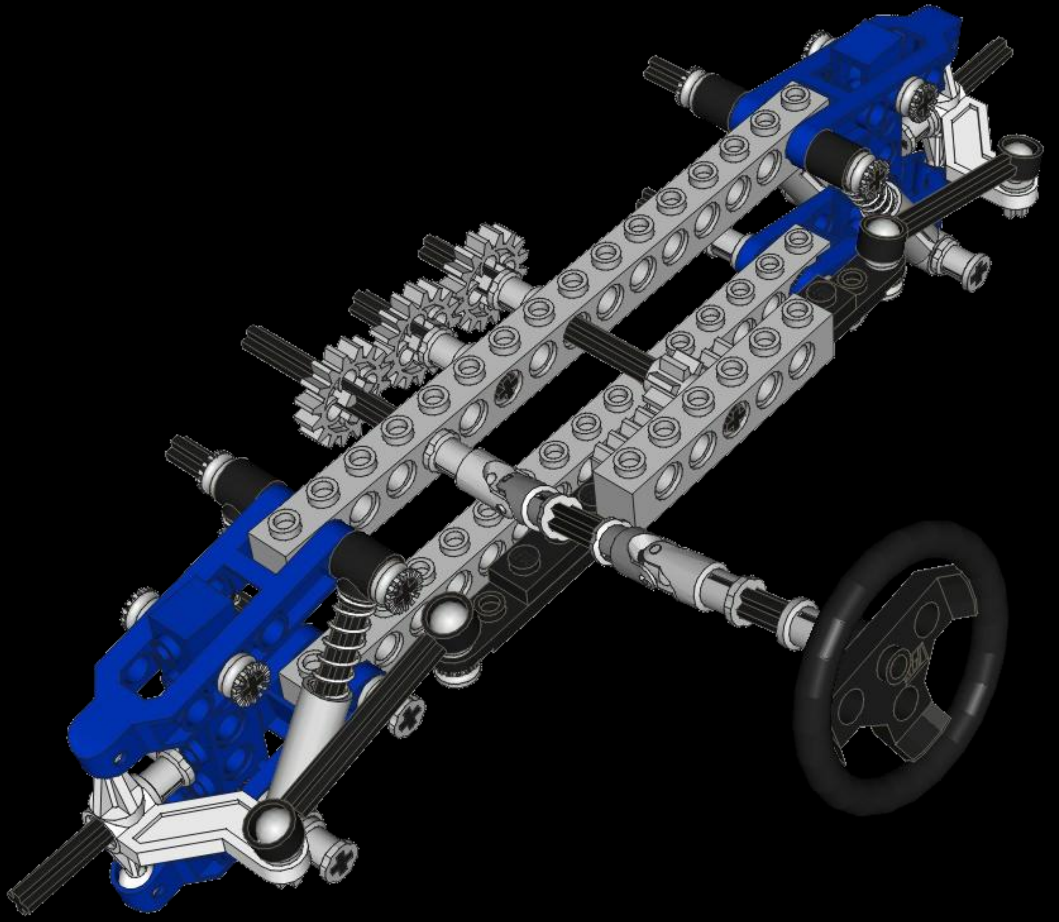
Separation of Concerns (SoC)

Single Responsibility Principle (SRP)

The Product

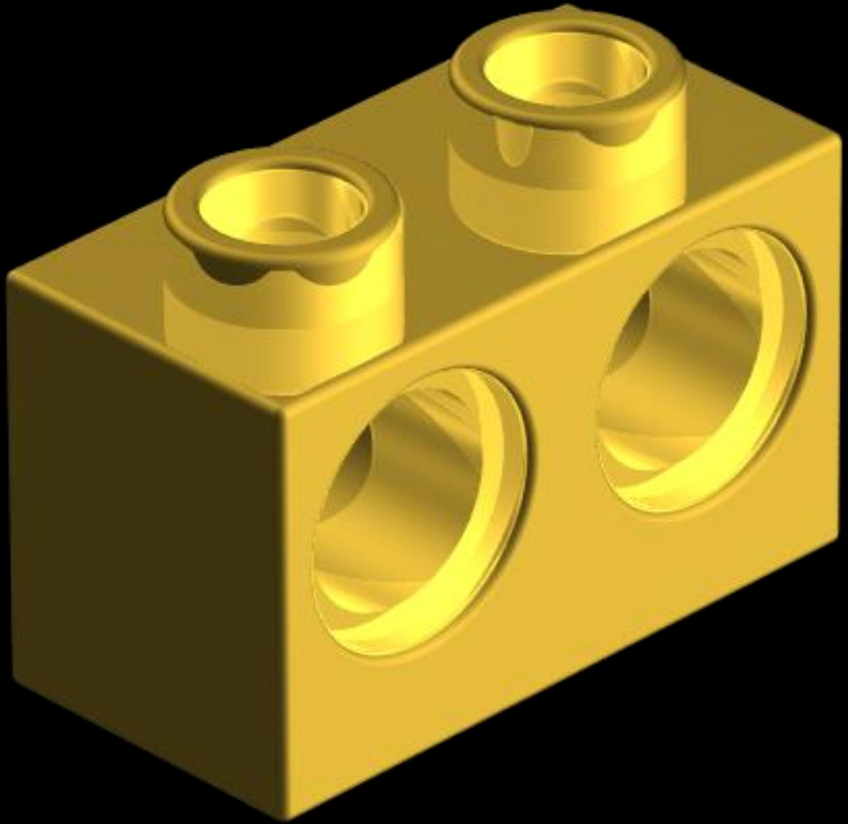
http://www.technicopedia.com/8865.html

Component / Service



http://www.technicopedia.com/8865.html

Class, Struct, Enum etc.



http://technicbricks.blogspot.com/2009/06/tbs-techpoll-12-results-2009-1st.html

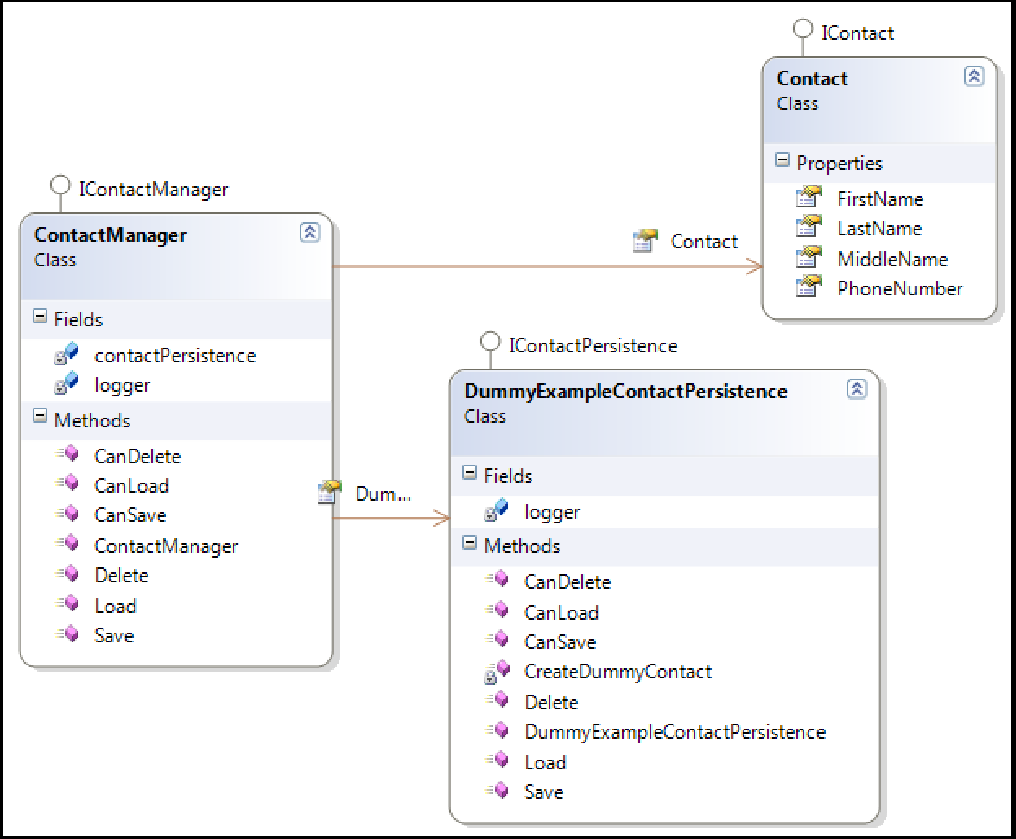
Separation of Concerns (SoC)

probably by Edsger W. Dijkstra in 1974

- "In computer science, separation of concerns (SoC) is the process of separating a computer program into distinct features that overlap in functionality as little as possible.

- A concern is any piece of interest or focus in a program. Typically, concerns are synonymous with features or behaviors. "

http://en.wikipedia.org/wiki/Separation_of_Concerns



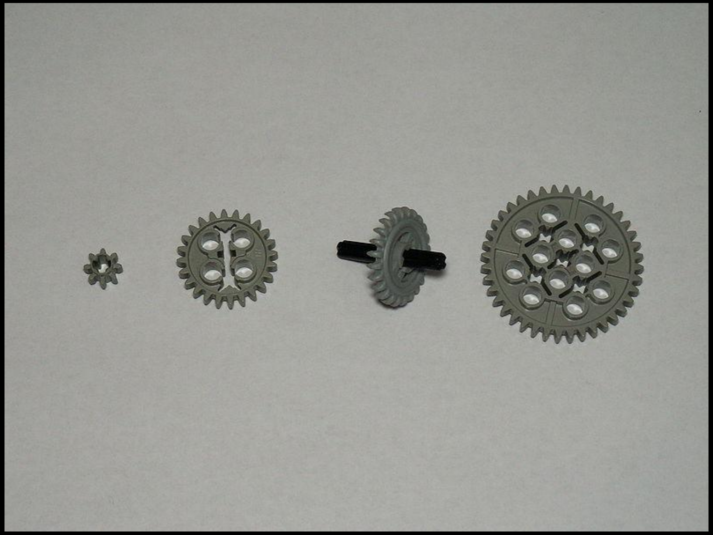
Single Responsibility Principle (SRP)

by Robert C Martin

“Every object should have a single responsibility, and that responsibility should be entirely encapsulated by the class.”

http://en.wikipedia.org/wiki/Single_responsibility_principle

```
public class Logger : ILogger
{
   public Logger(ILoggingSink loggingSink)
   {}

   public void Log(string message)
   {}
}
```



http://www.ericalbrecht.com

Source Code Conventions

“Clean Code” –Guidelines *

by Robert C. Martin

- use meaningful, pronounceable, searchable Names
- write code readable top to bottom (Journal Style)
- prefer Exceptions over returning Error Codes
- explain yourself in Code
- avoid redundant, misleading and noise Comments
- don’t use a Comment when you can use a Method or Variable
- Avoid commented-out code and Javadocs in NonPublic Code
- Don’t Return or Pass Null
- Keep Tests Clean and have only One Assert per Test
- Classes and Methods should be small
- Limit the scope of Data and use Copies of Data
- Builds and Tests should only require 1 Step

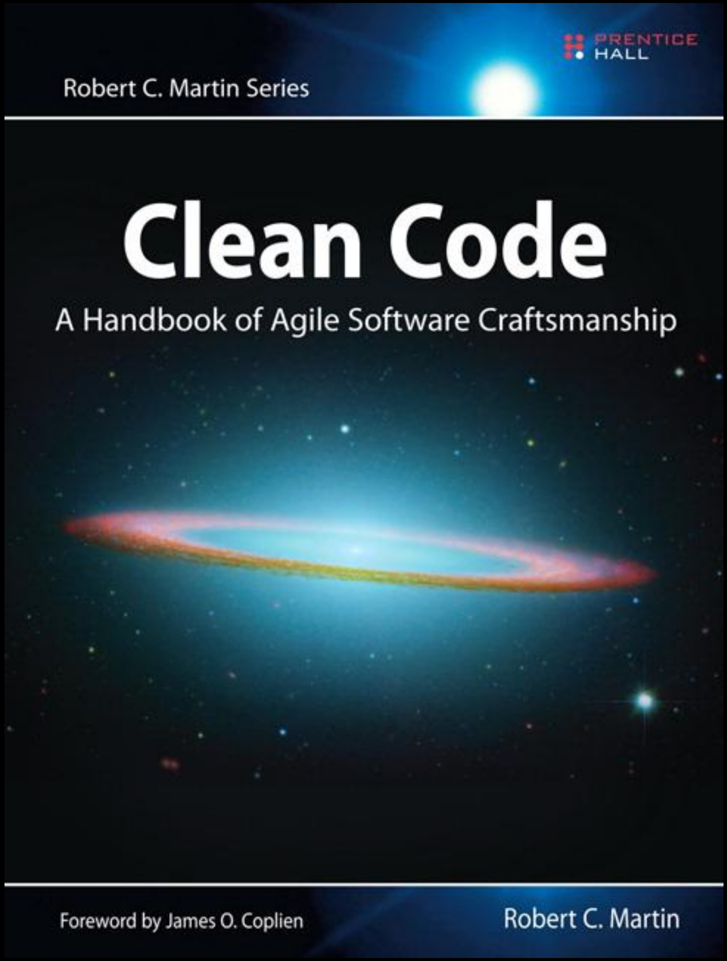
* Chapter extract: Robert C. Martin –” Clean Code”, Parson Education, Inc. 2008

The “Must Read”-Book(s)

by Robert C Martin

A Handbook of Agile Software Craftsmanship

“Even bad code can function. But if code isn’t clean, it can bring a development organization to its knees.”

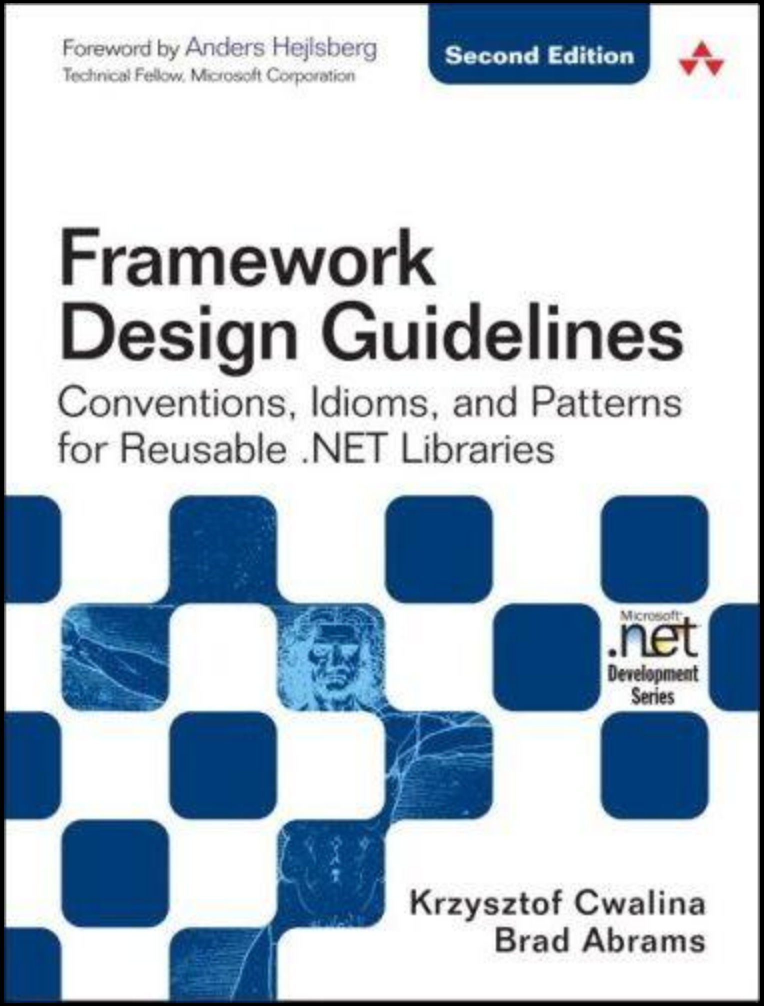

PRENTICE HALL
Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin

The "Must Read"-Book(s)

by Krzysztof Cwalina, Brad Abrams

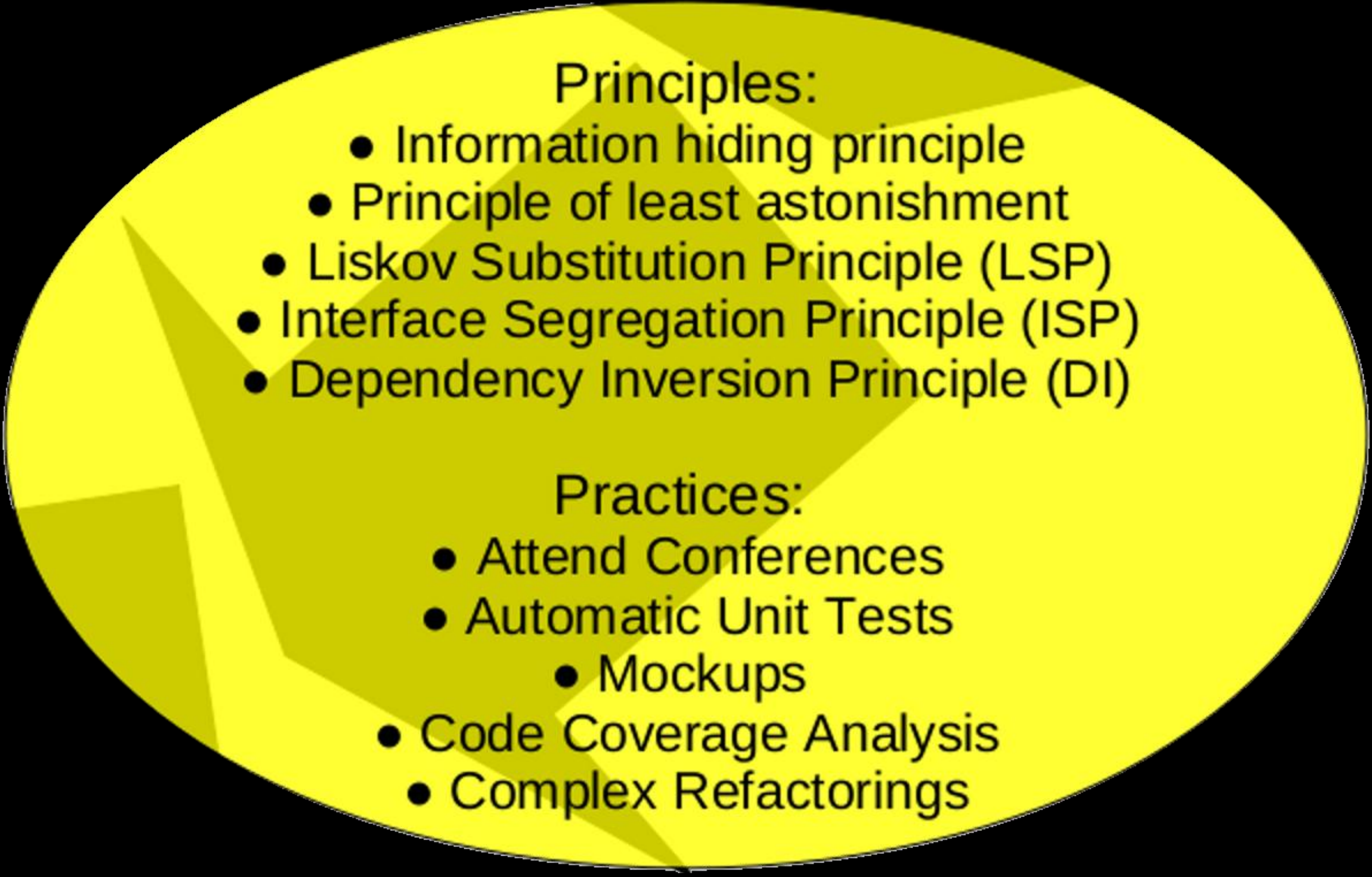
Framework Design Guidelines

"teaches developers the best practices for designing reusable libraries for the Microsoft .NET Framework."


Foreword by Anders Hejlsberg
Technical Fellow, Microsoft Corporation
Second Edition
Framework Design Guidelines
Conventions, Idioms, and Patterns for Reusable .NET Libraries
Microsoft .net Development Series
Krzysztof Cwalina
Brad Abrams

Clean Code Developer – 3rd Iteration

by Ralf Westphal & Stefan Lieser – http://www.clean-code-developer.de



Principles:
- Information hiding principle
- Principle of least astonishment
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DI)

Practices:
- Attend Conferences
- Automatic Unit Tests
- Mockups
- Code Coverage Analysis
- Complex Refactorings

Graphic by Michael Hönnig http://michael.hoennig.de/2009/08/08/clean-code-developer-ccd/

Information Hiding Principle (IHP)

Information Hiding Principle (IHP)

by David Parnas (1972)

“.. information hiding is the principle of segregation of the design decisions on a computer program that are most likely to change, ..”

http://en.wikipedia.org/wiki/Information_hiding

Liskov Substitution Principle (LSP)

Liskov Substitution Principle (LSP)

by Barbara Liskov, Jannette Wing (1994)

“Liskov’s notion of a behavioral subtype defines a notion of substitutability for mutable objects”

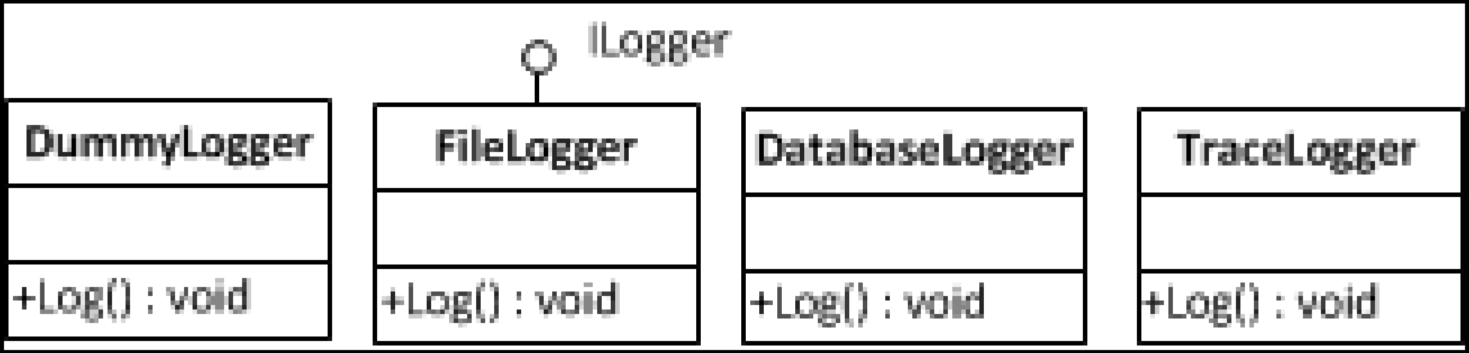
http://en.wikipedia.org/wiki/Liskov_substitution_principle

Interfaces / Contracts

- Decouple Usage and Implementation through introduction of a contract
- Allows to replace implementation without changing the consumer

```
public interface ILogger
{
  void Log(string message);
}
```

```
public class Logger : ILogger
{
    public Logger(ILoggingSink loggingSink)
    {}

    public void Log(string message)
    {}
}
```



Dependency Inversion Principle (DIP)

Dependency Inversion Principle (DIP)

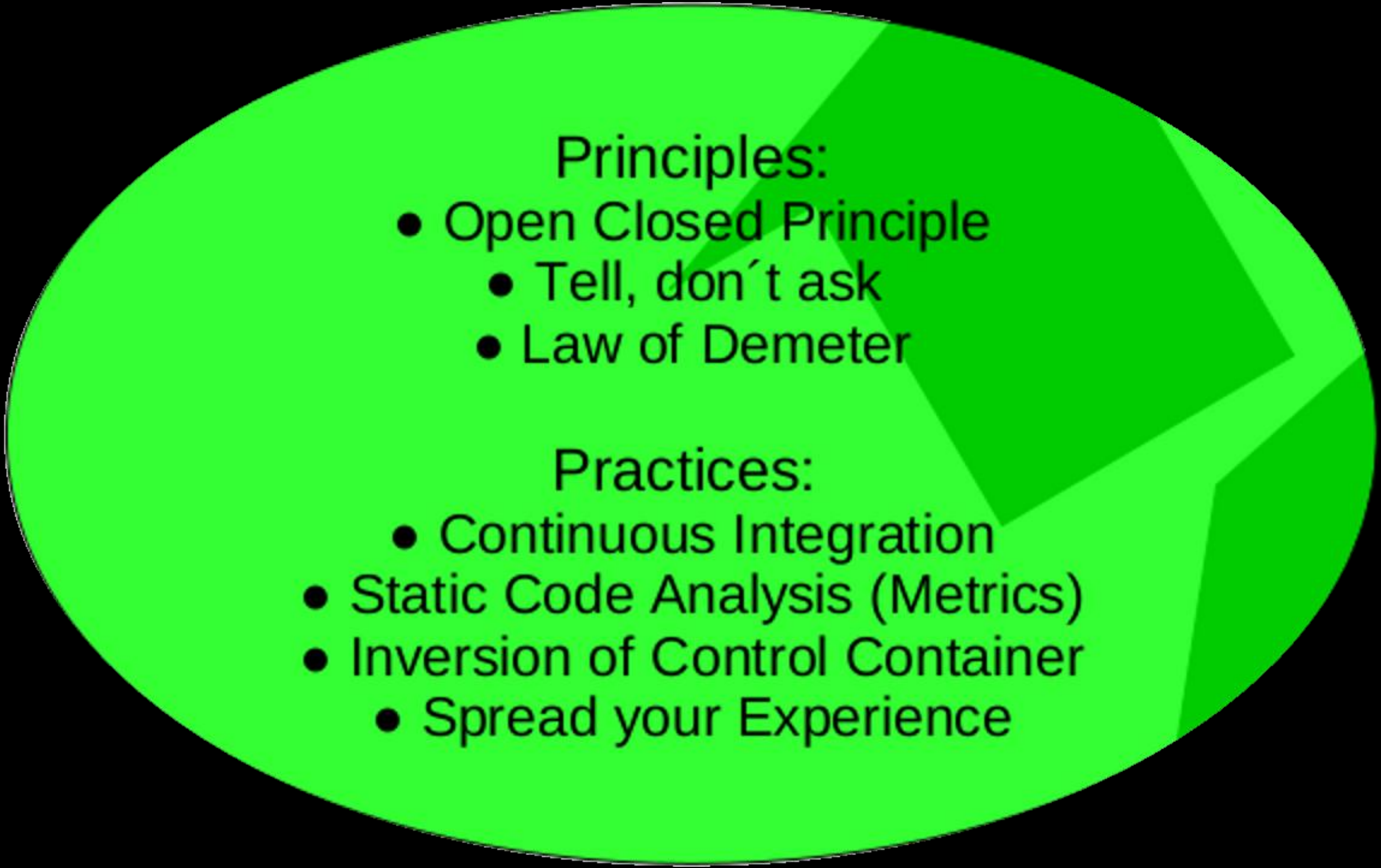
by Robert C. Martin

- “High-level modules should not depend on low-level modules. Both should depend on abstractions.
- Abstractions should not depend upon details. Details should depend upon abstractions.”

http://en.wikipedia.org/wiki/Dependency_inversion_principle

Clean Code Developer – 4th Iteration

by Ralf Westphal & Stefan Lieser – http://www.clean-code-developer.de



Principles:

- Open Closed Principle
- Tell, don´t ask
- Law of Demeter

Practices:

- Continuous Integration
- Static Code Analysis (Metrics)
- Inversion of Control Container
- Spread your Experience

Graphic by Michael Hönnig http://michael.hoennig.de/2009/08/08/clean-code-developer-ccd/

Open Closed Principle (OCP)

Open/Closed Principle (OCP)

by Bertrand Meyer (1988)

An implementation is open for extension but closed for modification

http://en.wikipedia.org/wiki/Open/closed_principle

Law of Demeter (LoD)

Law of Demeter (LoD)

Northeastern University (1987)

“

- Each unit should have only limited knowledge about other units: only units “closely” related to the current unit.
- Each unit should only talk to its friends; don’t talk to strangers
- Only talk to your immediate friends.”

http://en.wikipedia.org/wiki/Law_Of_Demeter

S Single Responsibility Principle

O Open/Closed Principle

L Liskov Substitution Principle

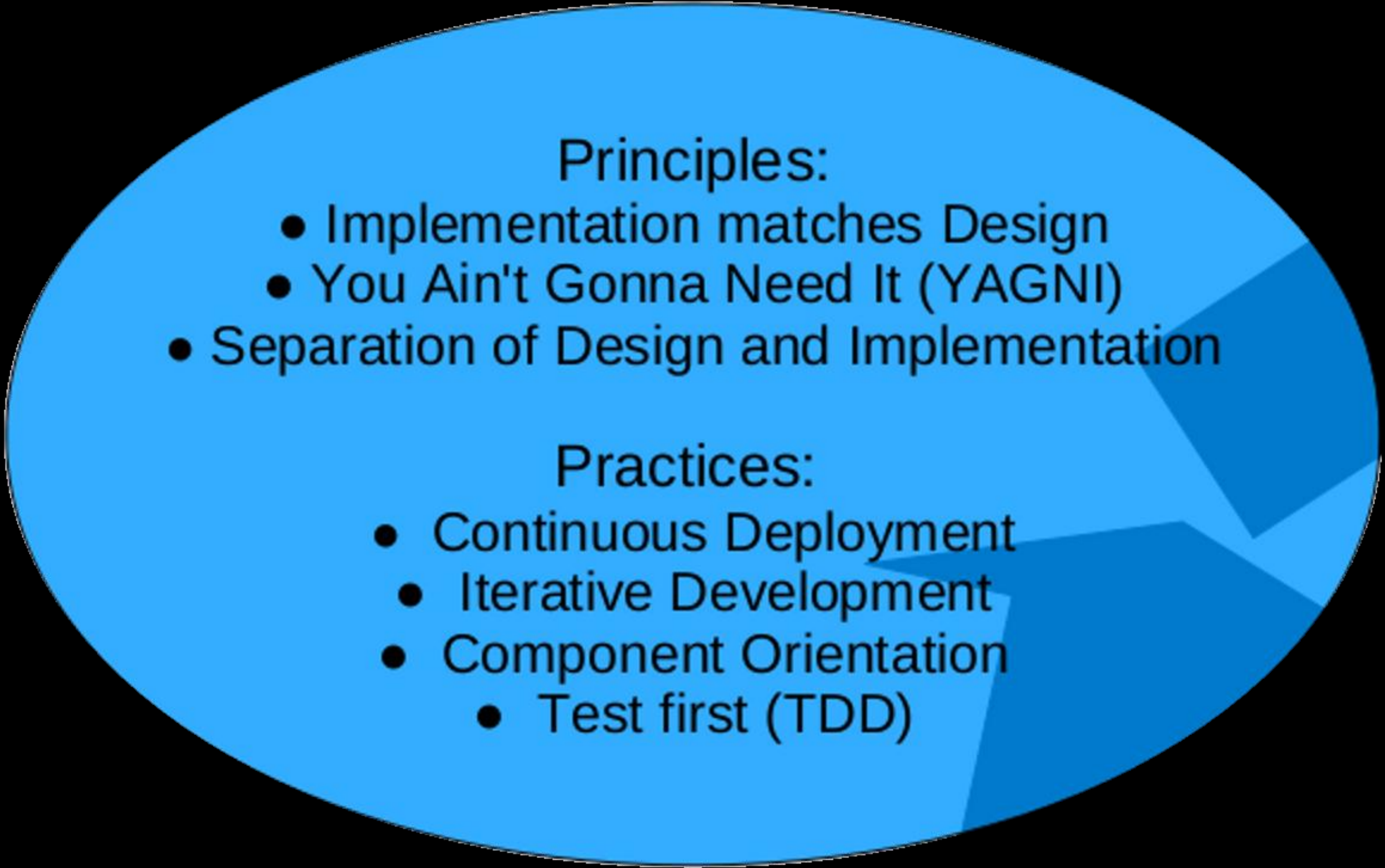
I Interface Segregation Principle

D Dependency Inversion Principle

Robert C Martin: http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

Clean Code Developer – 5th Iteration

by Ralf Westphal & Stefan Lieser – http://www.clean-code-developer.de



Principles:

- Implementation matches Design
- You Ain't Gonna Need It (YAGNI)
- Separation of Design and Implementation

Practices:

- Continuous Deployment
- Iterative Development
- Component Orientation
- Test first (TDD)

Graphic by Michael Hönnig http://michael.hoennig.de/2009/08/08/clean-code-developer-ccd/

Component-Oriented Programming (CoP)

Different Ways of doing something similar

http://www.ericalbrecht.com

http://www.ericalbrecht.com

http://www.julianaheng.com/transformers-rotf-bumblebee-and-sam-action-figures/

Why Reusable Components rock

http://www.ericalbrecht.com/technic/8020/8020all.jpg

Inversion of Control (IoC)

Inversion of Control (IoC)

by Martin Fowler 1994

```
Logger logger = new Logger();
```

Inversion of Control (IoC)

by Martin Fowler 1994

Logger Avoid *ger();*

Inversion of Control –

Constructor Injection

http://www.martinfowler.com/articles/injection.html

```
public class ContactManager : IContactManager
{
    public ContactManager(ILogger logger,
                                            IContactPersistence contactPersistence)
    {
        this.logger = logger;
        if (logger == null)
        {
            throw new ArgumentNullException("logger");
        }


      …
  }
}
```

Dependency Injection Container & more

- **Typically support all types of Inversion of Control mechanisms**
  - Constructor Injection
  - Property (Setter) Injection
  - Interface Injection
  - Service Locator

- **.NET based DI-Container**
  - Unity
  - Castle Windsor
  - StructureMap
  - Spring.NET
  - Autofac
  - Puzzle.Nfactory
  - Ninject
  - PicoContainer.NET
  - and more

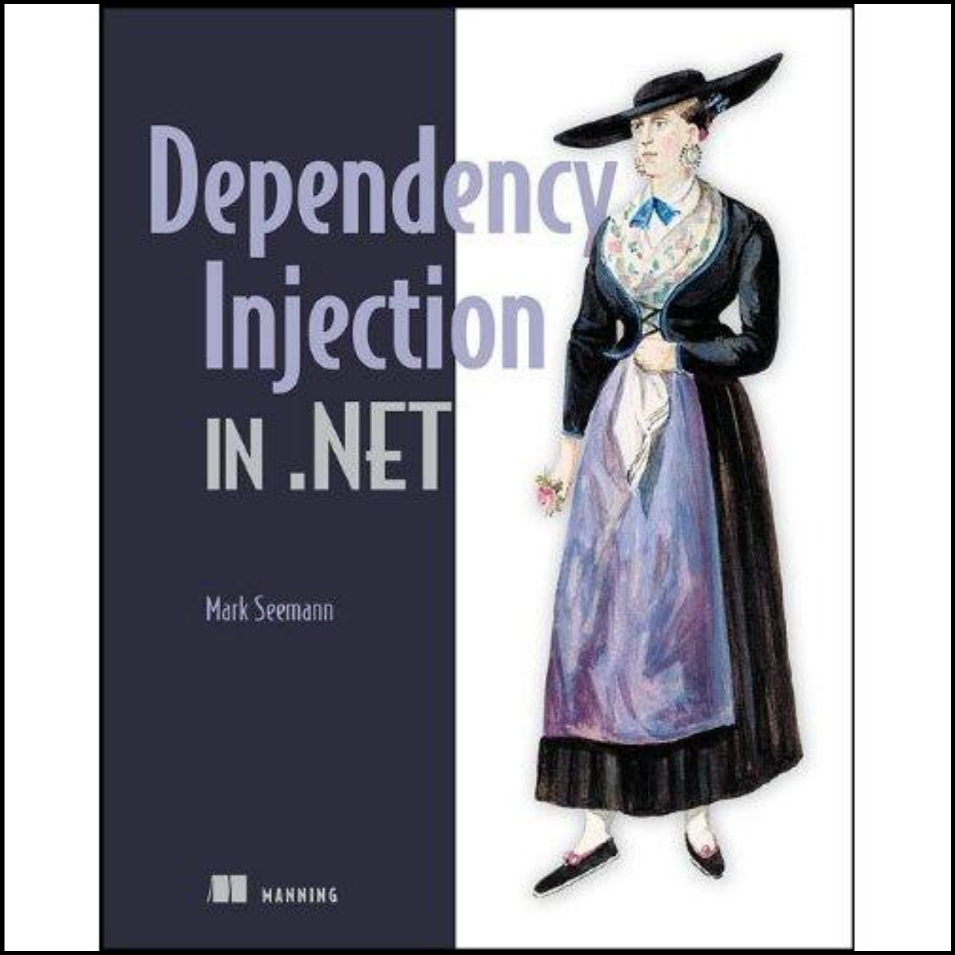
Related Technology:

- Managed Extensibility Framework (MEF)
- Windows Communication Foundation (WCF)

The “Must Read”-Book(s)

by Mark Seemann

Dependency Injection is a set of software design principles and patterns that enable us to develop loosely coupled code.

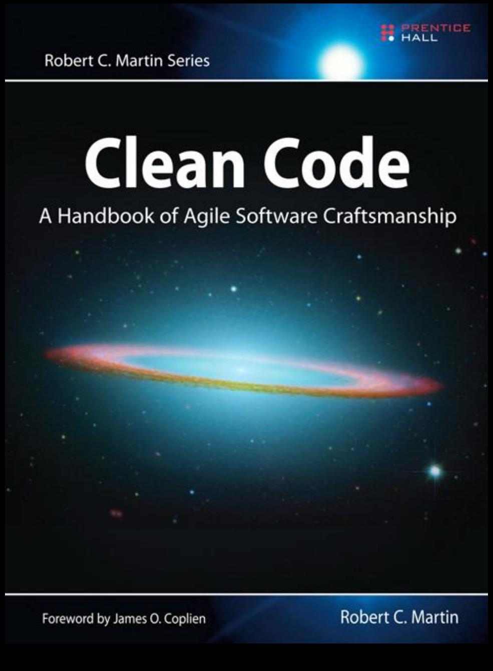


http://www.manning.com/seemann/

Summary Clean Code

Maintainability is achieved through:

- Readability (Coding Guidelines)
- Simplification and Specialization *(KISS, SoC, SRP, OCP,)*
- Decoupling *(LSP, DIP, IHP, Contracts, LoD, CoP, IoC or SOA)*
- Avoiding Code Bloat (DRY, YAGNI)
- Quality through Testability (all of them!)


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

Q & A

Downloads, Feedback & Comments:

theo@csharp-lighthouse.com
www.csharp-lightouse.com
www.speakerrate.com/theoj

Graphic by Nathan Sawaya courtesy of brickartist.com

References…

http://clean-code-developer.com
http://michael.hoennig.de/2009/08/08/clean-code-developer-ccd/
http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod
http://www.manning.com/seemann/
http://en.wikipedia.org/wiki/Keep_it_simple_stupid
http://picocontainer.org/patterns.html
http://en.wikipedia.org/wiki/Separation_of_concerns
http://en.wikipedia.org/wiki/Single_responsibility_principle
http://en.wikipedia.org/wiki/Information_hiding
http://en.wikipedia.org/wiki/Liskov_substitution_principle
http://en.wikipedia.org/wiki/Dependency_inversion_principle
http://en.wikipedia.org/wiki/Open/closed_principle
http://en.wikipedia.org/wiki/Law_Of_Demeter
http://en.wikipedia.org/wiki/Don't_repeat_yourself
http://en.wikipedia.org/wiki/You_ain't_gonna_need_it
http://en.wikipedia.org/wiki/Component-oriented_programming
http://en.wikipedia.org/wiki/Service-oriented_architecture
http://www.martinfowler.com/articles/injection.html
http://www.codeproject.com/KB/aspnet/IOCDI.aspx
http://msdn.microsoft.com/en-us/magazine/cc163739.aspx
http://msdn.microsoft.com/en-us/library/ff650320.aspx
http://msdn.microsoft.com/en-us/library/aa973811.aspx
http://msdn.microsoft.com/en-us/library/ff647976.aspx
http://msdn.microsoft.com/en-us/library/cc707845.aspx
http://msdn.microsoft.com/en-us/library/bb833022.aspx
http://unity.codeplex.com/
http://www.idesign.net/idesign/DesktopDefault.aspx?tabindex=5&tabid=11

… more References

Resharper
http://www.jetbrains.com/resharper/

FxCop / Code Analysis
http://msdn.microsoft.com/en-us/library/bb429476(VS.80).aspx
http://blogs.msdn.com/b/codeanalysis/
http://www.binarycoder.net/fxcop/index.html

Code Contracts
http://msdn.microsoft.com/en-us/devlabs/dd491992
http://research.microsoft.com/en-us/projects/contracts/

Pex & Mole
http://research.microsoft.com/en-us/projects/pex/

StyleCop
http://stylecop.codeplex.com/

Ghostdoc
http://submain.com/products/ghostdoc.aspx

Spellchecker
http://visualstudiogallery.msdn.microsoft.com/
7c8341f1-ebac-40c8-92c2-476db8d523ce//

Lego (trademarked in capitals as LEGO)

Please fill out the
online feedback, and…

… thanks for you attention!

And visit and support the
Bay.net User Group
http://baynetug.org