

Sponsors and Partners

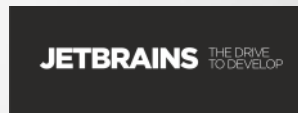
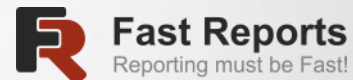
Strategic Sponsors



Gold Sponsors



Silver Sponsors





Deadly Sins of .NET Developers

Piotr Spikowski, Marcin Nowak

Agenda

Introduction

Performance

Security

Testing

Maintainability

Summary




Introduction

Disclaimer


Not real source code of our applications



Simple and basic examples to convey the message




Everyone knows about them, but they are still found in code



Common sins, not necessarily the most harmful



Let's learn from those mistakes!



Introduction

Imagine that IT now builds trucks

What expectations do you and your customers have?



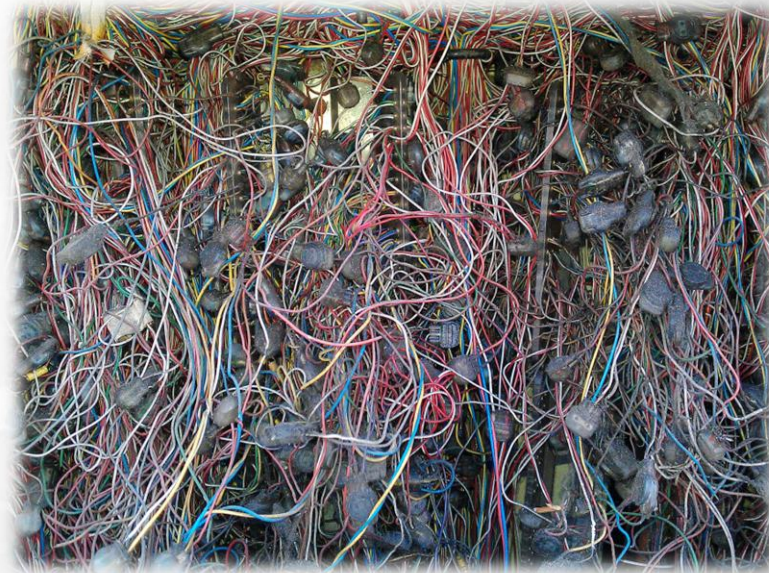
Introduction (cont.)

So everybody in the team starts working really hard...



Introduction (cont.)

However, the reality is closer to this...



Code gems

```
// Dear maintainer:  
//  
// Once you are done trying to 'optimize' this routine,  
// and have realized what a terrible mistake that was,  
// please increment the following counter as a warning  
// to the next guy:  
//  
// total_hours_wasted_here = 42
```

```
Exception up = new Exception("Something is really wrong.");  
throw up; //ha ha
```

```
const int TEN = 10; // As if the value of 10 will fluctuate...
```

Code gems (cont.)

```
//When I wrote this, only God and I understood what I was doing  
//Now, God only knows
```

```
// I dedicate all this code, all my work, to my wife, Darlene, who will  
// have to support me and our three children and the dog once it gets  
// released into the public.
```

```
// Magic. Do not touch.
```

```
// A Gorgon class - For the love of Zeus don't look directly at it!
```

```
try { ... }  
finally { // should never happen }
```

<http://stackoverflow.com/questions/184618/what-is-the-best-comment-in-source-code-you-have-ever-encountered>

Session goal





Performance

Measure, optimize & tune

Performance sins

No performance requirements

No 'real' data, volumes knowledge

No measurements

Optimizations with no perf figures

Optimizations in not needed areas

Measurement sins

Unit or integration tests used

Measuring the measurement code

Including JIT, cache warm-up

Excluding hardware config

Excluding runtime environment setup

No memory & GC related tuning

```
foreach (FlatMessage message in Messages)
{
    switch (message.Content.Substring(0, 3))
    {
        case "INS":
            InsertItem(message);
            break;
        case "LCK":
            LockItem(message);
            break;
    }
}
```



```
foreach (FlatMessage message in Messages)
{
    switch (message.FullType)
    {
        case FlatMessageType.INT:
            InsertItem(message);
            break;
        case FlatMessageType.LCK:
            LockItem(message);
            break;
    }
}
```

No behind-the-scene analysis

```
return Session
    .QueryOver(() => warehouseAlias).WithSubquery
        .WhereExists(QueryOver.Of(() => partInStockAlias)
            .Where(part => part.Warehouse.Id == warehouseAlias.Id && part.ReservedQuantity > 0)
            .And(Subqueries
                .WhereExists(QueryOver.Of<ReservedForOrder>()
                    .Where(r => r.PartsInStock.Id == partInStockAlias.Id && r.OrderId == scopeId)
                    .Select(r => r.Id)))
            .Select(c => c.Warehouse))
        .List<Warehouse>();
```

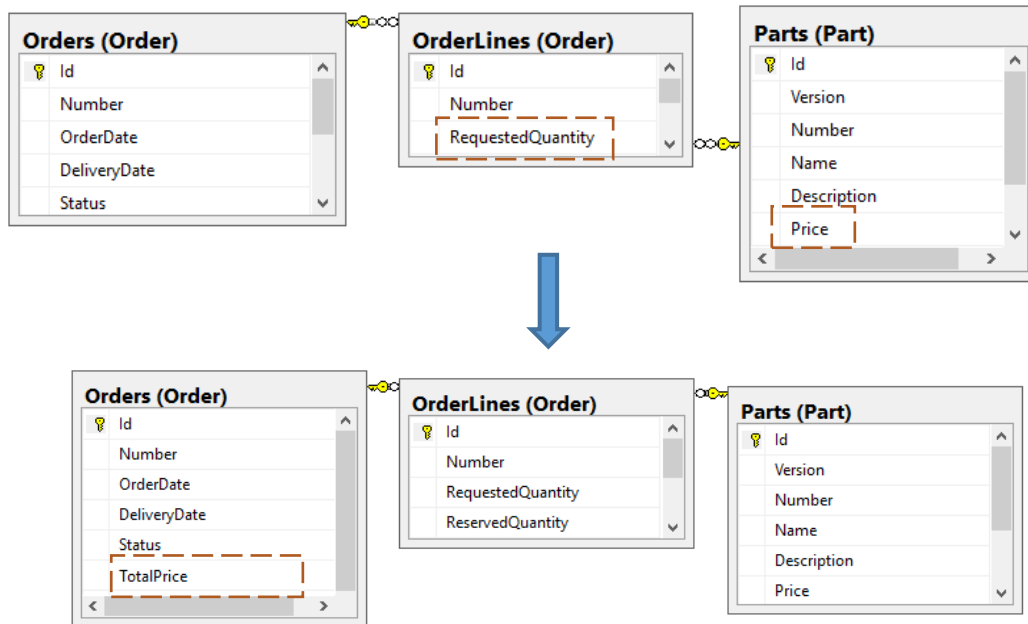

No deep-dive investigations

```
private static void OnRequestPlaced(int requestId)
{
    Logger.LogBusiness(string.Format("The request {0} has been placed", requestId));
}
```

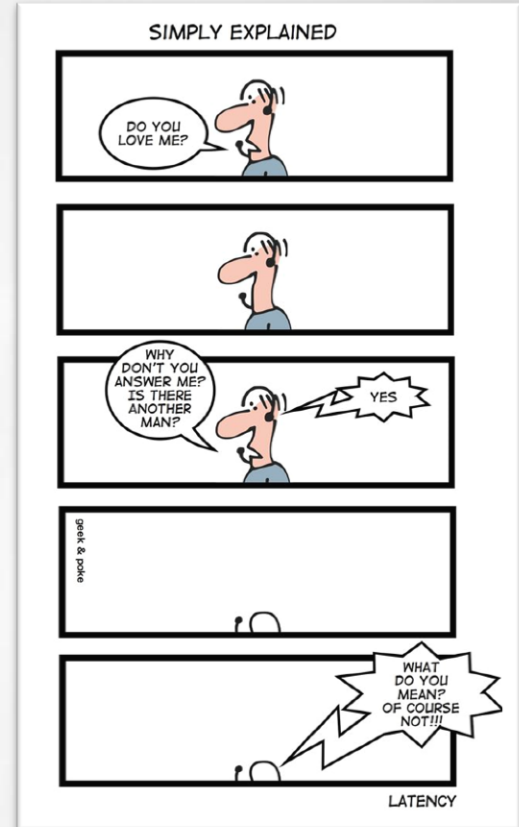
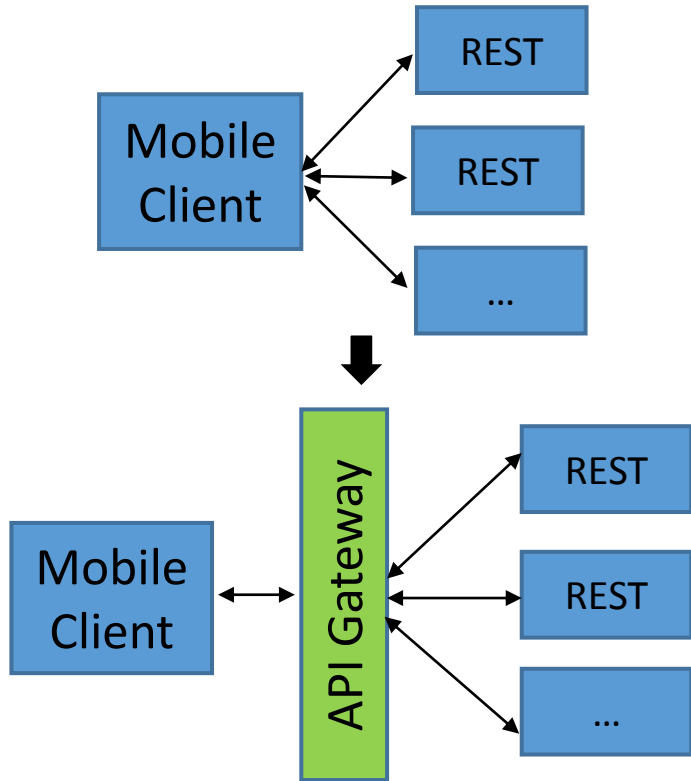


```
private static void OnRequestPlaced(int requestId)
{
    Logger.LogBusiness(string.Format("The request {0} has been placed", requestId.ToString()));
}
```

Design with no performance in mind



“Chatty” communication, latency



The cure

Include Performance in Requirements

Include Performance in Design

Constantly Measure

Ensure Measures are Comparable

Constantly Compare & Act

Toolset

GitHub

BenchmarkDotNet, NBench
Miniprofiler, Netling, GCVisualization

Visual Studio

Diagnostic Tools
Load Tests

Code
Reviews

Know what and when to use (e.g. simd System.Numerics.Vectors)
Know when to add threads, what should be async etc.

Ways of
Working

Architecture & Design Meetings
Utilize your test team



Security

Prevent, Detect & Respond

75% of attacks occur on **Web Applications**

More than **95%** of **Web Applications** have some sort of vulnerability

78% of easily exploitable weaknesses occur in **Web Applications**

More than **75%** of **Mobile Applications** fail basic security tests (2015)

Gartner.  **IMPERVA**®  Symantec.

Security sins

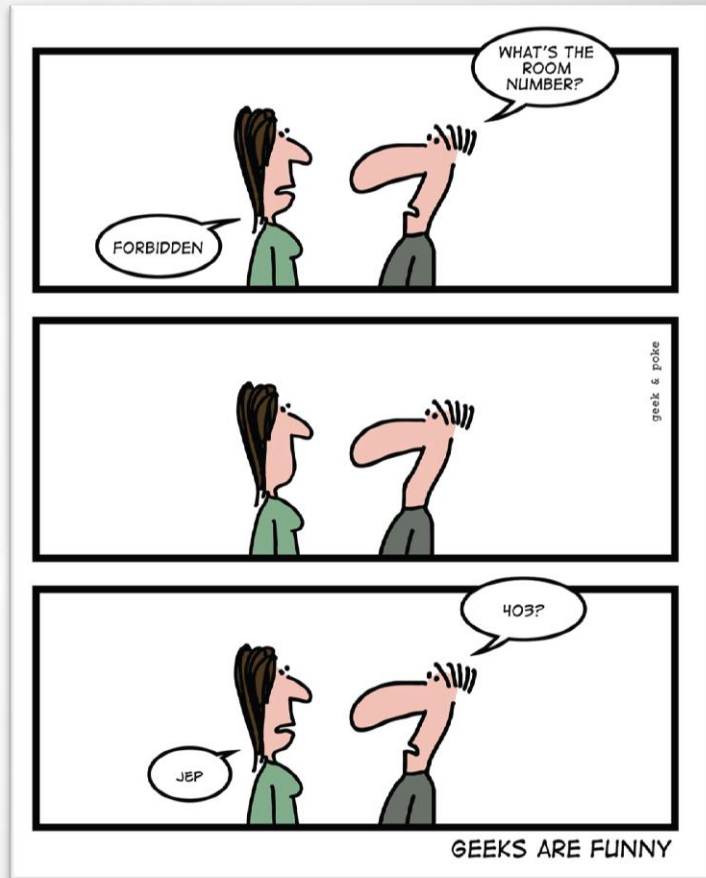
Ignored in intranet

Anonymous works rule

Incorrect authorization

Incorrect assumptions

Outdated tools, frameworks, algorithms



Security sins

Logging sensitive data

Incorrect client side protection

No security requirements

No security testing

Passwords in clear text

Dev to prod access

Classic weaknesses - SQL injection

```
public Party FindByName(string name)
{
    ...
    var q = Session.CreateSQLQuery("select * from Party where Name = '" + name + "'");
    ...
}
```

'Classic' may not be noticeable

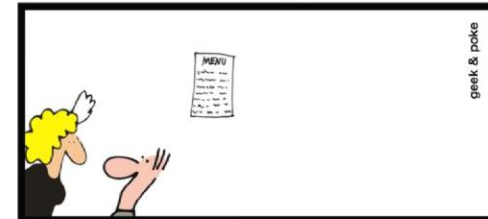
```
public void UpdateUserProcess(string value)
{
    Session.CreateSQLQuery("EXEC UpdateUserProcess :value")
        .SetString("value", value).ExecuteUpdate();
}

CREATE PROCEDURE UpdateUserProcess @value varchar(50)
AS
BEGIN
    DECLARE @sql nvarchar(MAX);
    SET @sql = 'UPDATE Process SET Value = ' + @value + ' WHERE ' + GetProcessCondition();
    EXEC(@sql);
END
```


Hash with no salt

```
private static string ComputePasswordHash(string password)
{
    using (SHA256 alg = SHA256.Create())
    {
        byte[] hash = alg.ComputeHash(Encoding.UTF8.GetBytes(password));
        return Convert.ToBase64String(hash);
    }
}
```

1000 WAYS TO COMPLAIN IN A RESTAURANT



PART 1: THE SECURITY GEEK WAY

Incorrect salt and...

```
private static string ComputePasswordHash(string password, out byte[] saltBytes)
{
    byte[] pwdBytes = Encoding.UTF8.GetBytes(password);

    saltBytes = new byte[10];
    Random random = new Random();
    random.NextBytes(saltBytes);
    // RNGCryptoServiceProvider random = new RNGCryptoServiceProvider();
    // random.GetNonZeroBytes(saltBytes);

    byte[] input = new byte[pwdBytes.Length + saltBytes.Length];
    saltBytes.CopyTo(input, 0);
    pwdBytes.CopyTo(input, saltBytes.Length);

    using (SHA256 alg = SHA256.Create())
    {
        byte[] digest = alg.ComputeHash(input);
        return Convert.ToBase64String(digest);
    }
}
```

The cure

Define Security Requirements

Classify Information

Run Vulnerability Scanning

Define Identity Management Rules

Perform Risk Analysis

The cure (cont.)

Use Existing Frameworks, Components & Standards

Do not Mix Authentication with Authorization

Write Tests

Treat Authorization as Business Logic

Prefer Claim Based Authorization

Use Operations Over Roles

The cure (cont.)

Use available articles,
methodologies,
documentation & tools

- OWASP Top 10 for Web Applications
- OWASP Top 10 Mobile Risks
- OWASP Dependency Checker tool

Introduction

Performance

Security

Testing

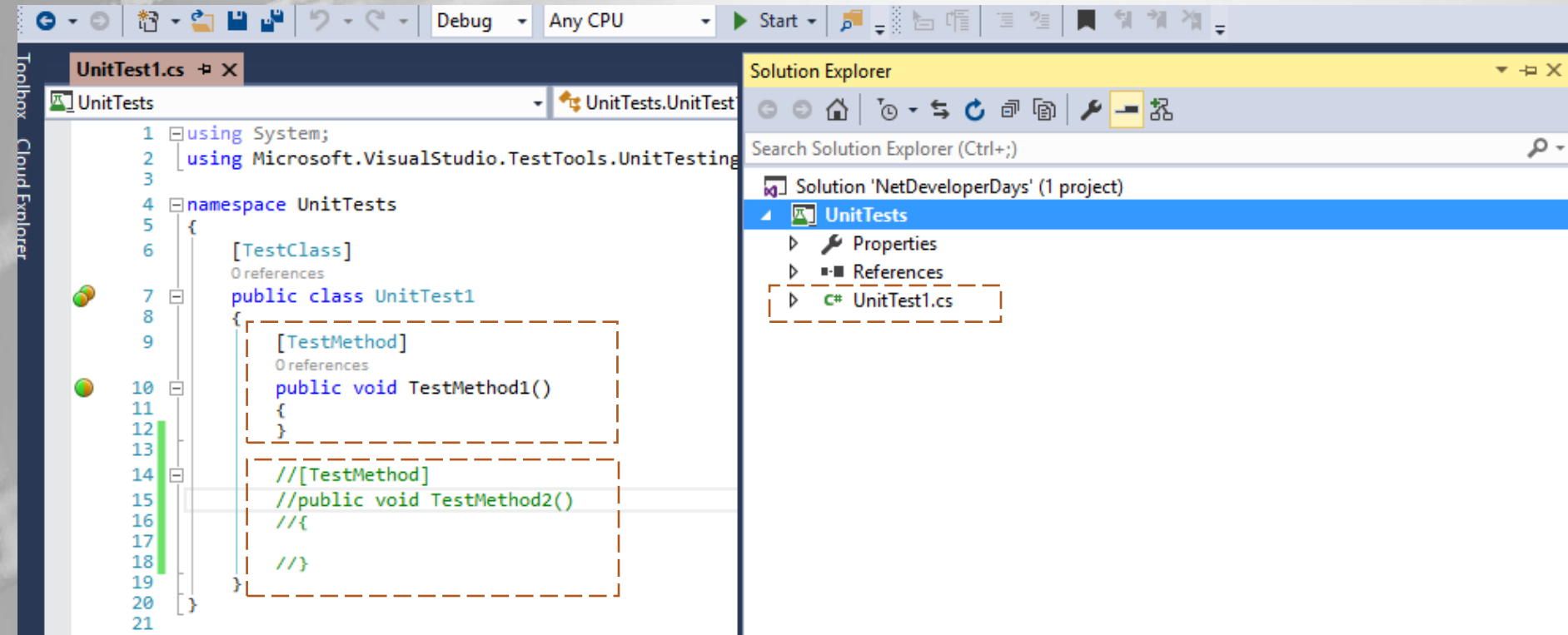
Maintainability

Summary

Testing

Find and fix problems
before they become expensive

No tests at all



The screenshot shows the Visual Studio IDE with the following components:

- Code Editor:** Displays `UnitTests.cs` with the following code:

```
1 using System;
2 using Microsoft.VisualStudio.TestTools.UnitTesting;
3
4 namespace UnitTests
5 {
6     [TestClass]
7     public class UnitTest1
8     {
9         [TestMethod]
10        public void TestMethod1()
11        {
12        }
13
14        // [TestMethod]
15        // public void TestMethod2()
16        // {
17        // }
18    }
19 }
20
21
```

The code is annotated with red dashed boxes: one around the `[TestMethod]` attribute and the `TestMethod1()` method, and another around the commented-out `[TestMethod]` attribute and `TestMethod2()` method.
- Solution Explorer:** Shows the project structure for 'NetDeveloperDays' (1 project). The `UnitTests` folder is expanded, showing `Properties`, `References`, and `C# UnitTest1.cs`.
- Toolbox:** On the left, the 'Cloud Explorer' tab is visible.
- Toolbar:** At the top, the 'Debug' dropdown is set to 'Any CPU' and the 'Start' button is visible.

Non testable code

```
public class DAL
{
    public List<OrderEntity> GetOrdersForGivenDate(DateTime date)
    {
        return new List<OrderEntity>()
        {
            new OrderEntity() {CreatedAt = date, Id = 1, Price = 25.7m, ShippingAddress = "Lyon"},
            new OrderEntity() {CreatedAt = date, Id = 2, Price = 25.3m, ShippingAddress = "Wroclaw"},
            new OrderEntity {CreatedAt = date, Id = 3, Price = 49, ShippingAddress = "Gent"}
        };
    }
}
```

```
public class BL
{
    private DAL dal = new DAL();

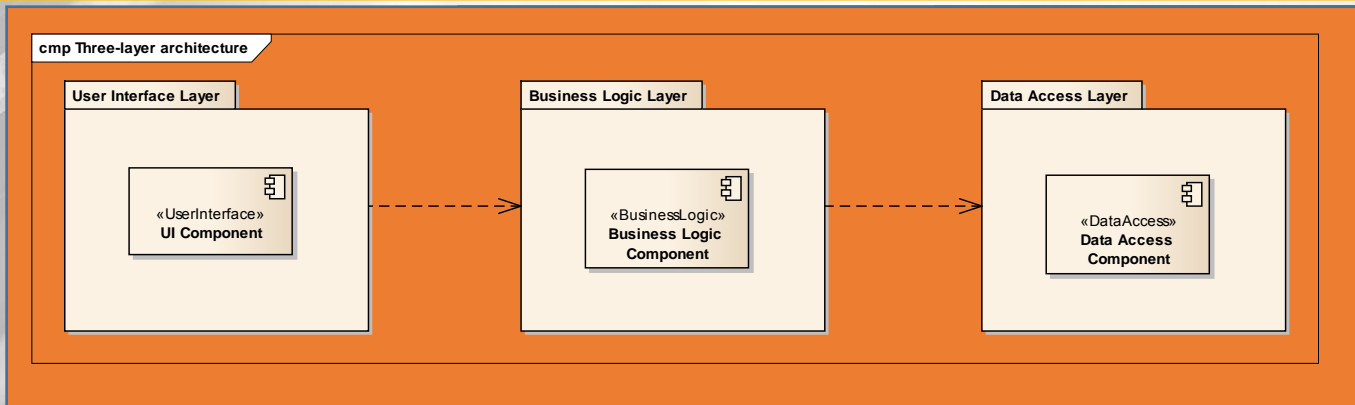
    public decimal CalculateTotalValueOfOrdersForAGivenDate(DateTime date)
    {
        List<OrderEntity> orders = dal.GetOrdersForGivenDate(date);
        return orders.Sum(x => x.Price);
    }
}
```

Lack of isolation

```

[TestClass]
public class BLTests
{
    [TestMethod]
    public void TestIfTotalIs100()
    {
        // Arrange
        DAL dal = new TestDAL();
        BL b1 = new BL(dal);
        // Act
        var result = b1.CalculateTotalValueOfOrdersForAGivenDate(DateTime.Parse("2014-04-01"));
        // Assert
        Assert.AreEqual(100, result);
    }
}

```



Not local test data



```
graph LR; Input --> UUT[Unit Under Test]; UUT --> Output;
```

Input

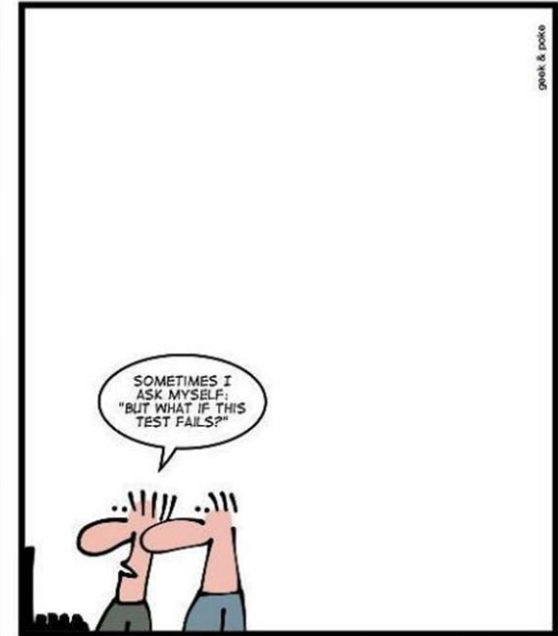
Unit
Under Test

Output

Testing the obvious/wrong thing

```
[TestMethod]
public void TestXor()
{
    Assert.IsTrue(true ^ false);
    Assert.IsFalse(true ^ true);
    Assert.IsFalse(false ^ false);
}
```

PHILOSOPHISING GEEKS



`assert(true);`

Missing assertions

```
[TestMethod]
[TestCategory("Unit Test")]
public void UpdateUserSuccess()
{
    // Arrange
    var user = fixture.Build<User>().Without(x => x.Id).Without(x => x.Version).Create();
    userRepository.Setup(x => x.Merge(user)).Returns(user);

    // Act
    userService.SaveUser(user);
}
```


Redundancy

```
[TestMethod]
[TestCategory("Unit Test")]
public void CreateUserSuccess()
{
    // Arrange
    fixture = new Fixture();
    var user = fixture.Build<User>().Without(x => x.Id).Without(x => x.Version).Create();
    userRepository.Setup(x => x.Save(user));
    // Act
    userService.CreateUser(user);
    // Assert
    userRepository.Verify(x => x.Save(user), Times.Once());
}

[TestMethod]
[TestCategory("Unit Test")]
public void UpdateUserSuccess()
{
    // Arrange
    fixture = new Fixture();
    var user = fixture.Build<User>().Without(x => x.Id).Without(x => x.Version).Create();
    userRepository.Setup(x => x.Merge(user)).Returns(user);

    // Act
    userService.SaveUser(user);

    // Assert
    userRepository.Verify(x => x.Merge(user), Times.Once());
}
```

State verification only

```
[TestMethod]
[TestCategory("Unit Test")]
public void CreateUserSuccess()
{
    // Arrange
    var user = fixture.Build<User>().Without(x => x.Id).Without(x => x.Version).Create();
    var wasCreated = userRepository.Setup(x => x.Save(user)).Returns(true);
    // Act
    userService.CreateUser(user);
    // Assert
    Assert.IsTrue(wasCreated);
}
```

The cure

No tests at all

Awareness

Non testable code

Test Driven Development

Dependency injection

Lack of isolation

Using abstractions

Mocking frameworks

- Moq
- Fakes

Not local test data

No global setup

The cure (cont.)

Testing the obvious/wrong thing

Balancing what to test

Missing assertions

Arrange, Act, Assert

Redundancy

Test initializers/cleanup

AutoFixture library or ObjectMother pattern

State verification only

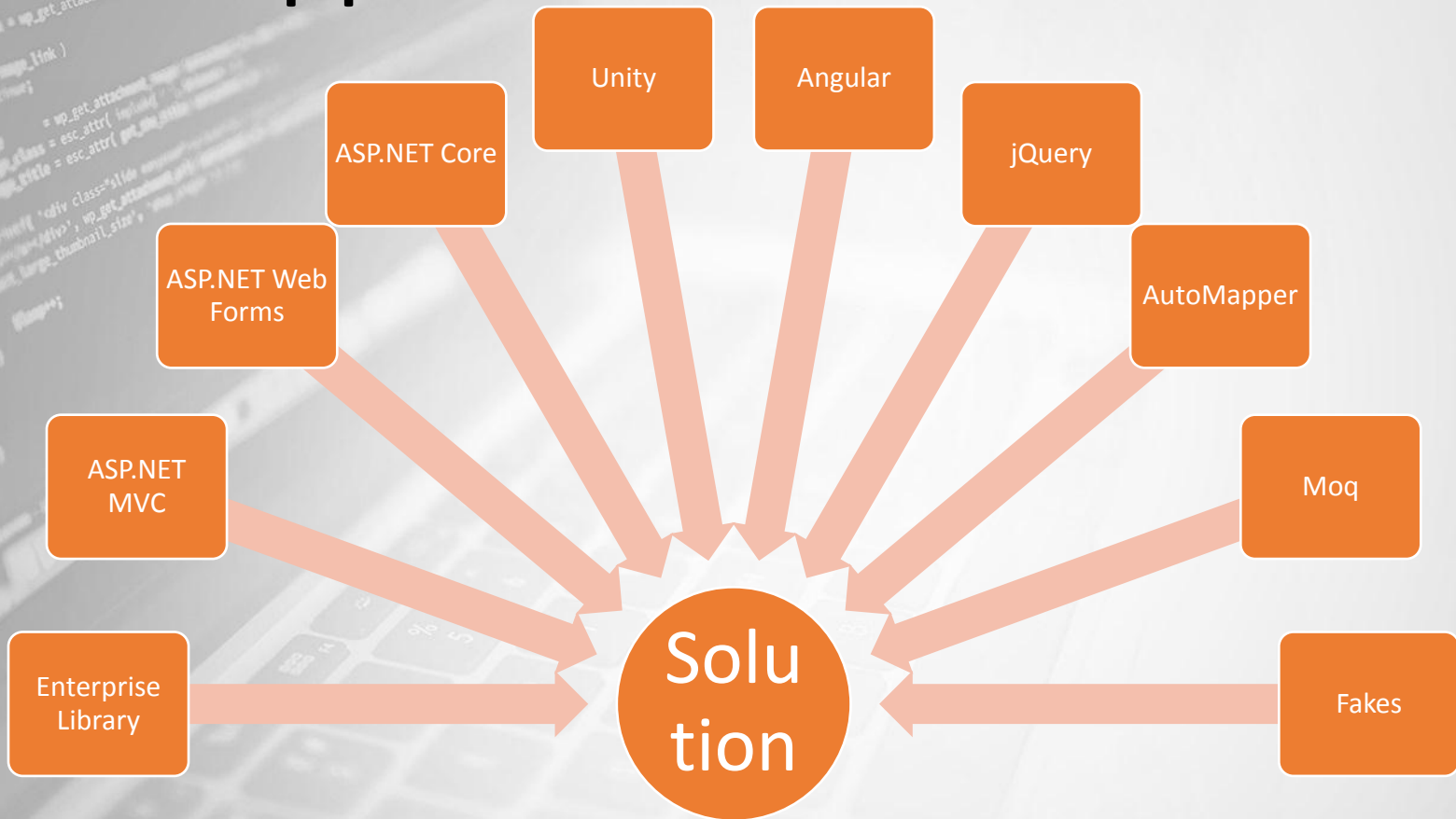
Behaviour verification



Maintainability

Reducing maintenance expenses

Lack of appointed toolset



Lack of code comments



Volvo Trucks. Driving Progress

TECHNICAL SPECIFICATIONS

I-Shift Dual Clutch



June 2014



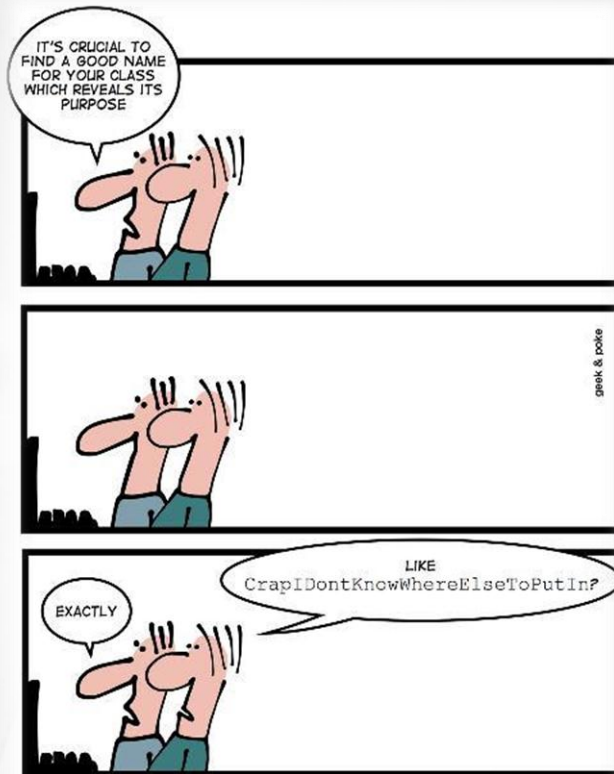
PROGRAMMING IS AN ART

Conventions

```
for (int i = 0; i < 10; i++)  
{  
    Console.WriteLine(i);  
}
```

```
private string m_sMyString;  
  
private int m_iOrderNo;
```

```
public void DoSomething(string param1, int param2)  
{  
    //...  
}
```

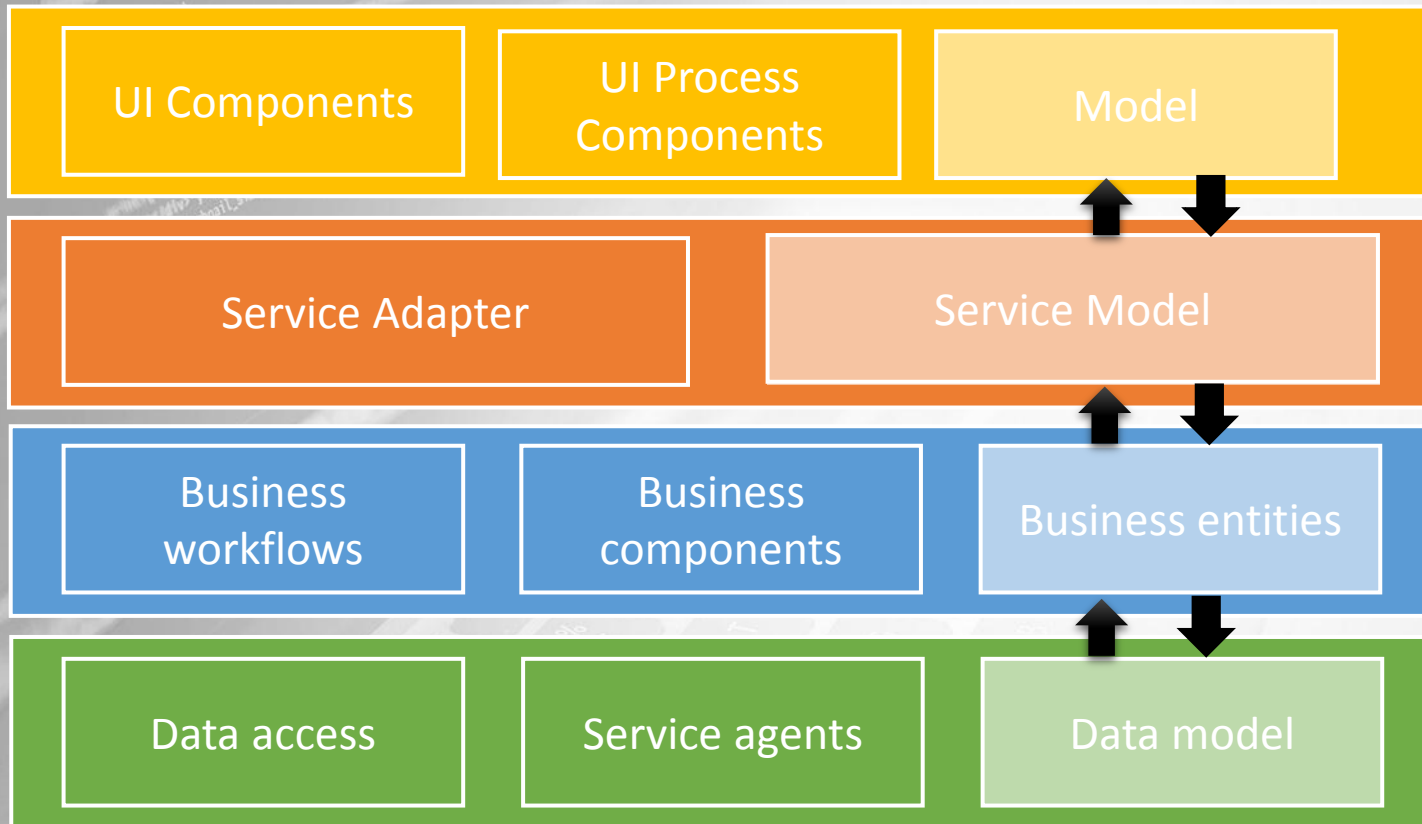


NAMING IS KEY

Lack of design patterns



Manual translations



SOLID violated



Poor reusability and code duplication

```
public class ConfigurationService
{
    public string ReadValueFromFile()
    {
        var fileStream = new FileStream("UserFile.txt", FileMode.Open, FileAccess.Read);
        using (var streamReader = new StreamReader(fileStream, Encoding.UTF8))
        {
            return streamReader.ReadToEnd();
        }
    }
}

public class FileVerification
{
    public bool VerifyFileVersion()
    {
        var fileStream = new FileStream("MainStore.bin", FileMode.Open, FileAccess.Read);
        using (var streamReader = new StreamReader(fileStream, Encoding.UTF8))
        {
            var content = streamReader.ReadToEnd();
            if (content.StartsWith("Ver1"))
                return true;

            return false;
        }
    }
}
```

The cure

How we code

- Code comments (at least for public API)
- Arrange, Act, Assert (for tests)
- Naming conventions
- Domain Driven Design
- SOLID
- Design patterns
- Minimal code
- Refactoring
- Boy scout rule

The cure (cont.)

Tools

- Resharper/JustCode
- Code Analysis
- Code Clones Analysis
- StyleCop
- SonarQube
- nDepend

Process

- Code and architecture reviews
- Libraries toolbox and lifecycle

Introduction

Performance

Security

Testing

Maintainability

Summary



Summary

Summary

Work on the awareness



Use proper tools and processes



Revisit frequently



Mature with the organization, like us :)

Sources

- <http://geek-and-poke.com/>
- <https://www.flickr.com/>
- Internal Volvo images

Sponsors and Partners

Strategic Sponsors



Gold Sponsors



Silver Sponsors

