See Code Differently

Kevin Trethewey

Who am I?

- Born in Zimbabwe in 1977
- Living in South Africa since 1988
- Started coding in 1996
- Started consulting, coaching and training full time in 2008
- Currently working for Driven Software

Environmental Constraints

Process

Code

Environmental Constraints

Process

Code

Environmental Constraints

Process

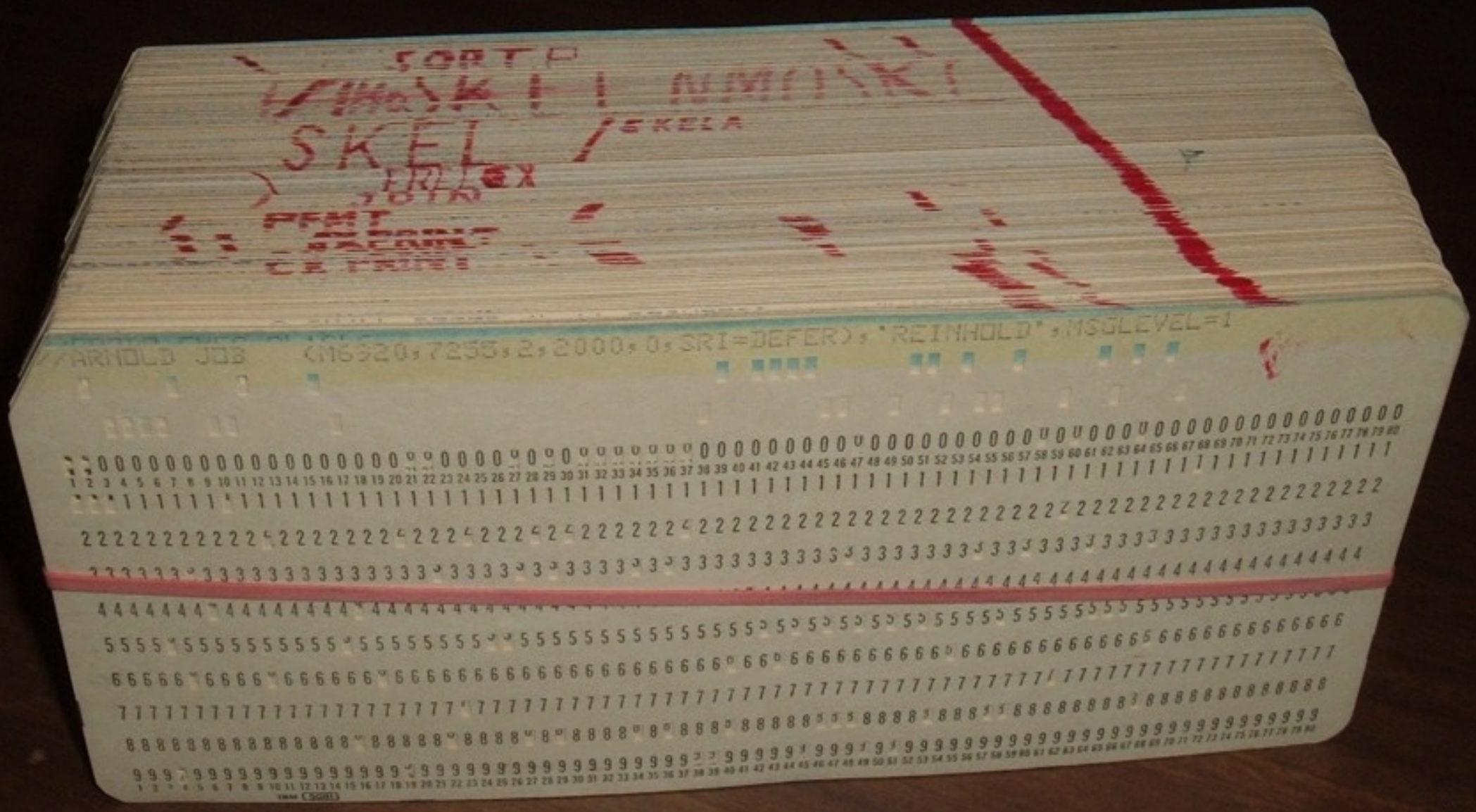
Code

Environmental Constraints

Process

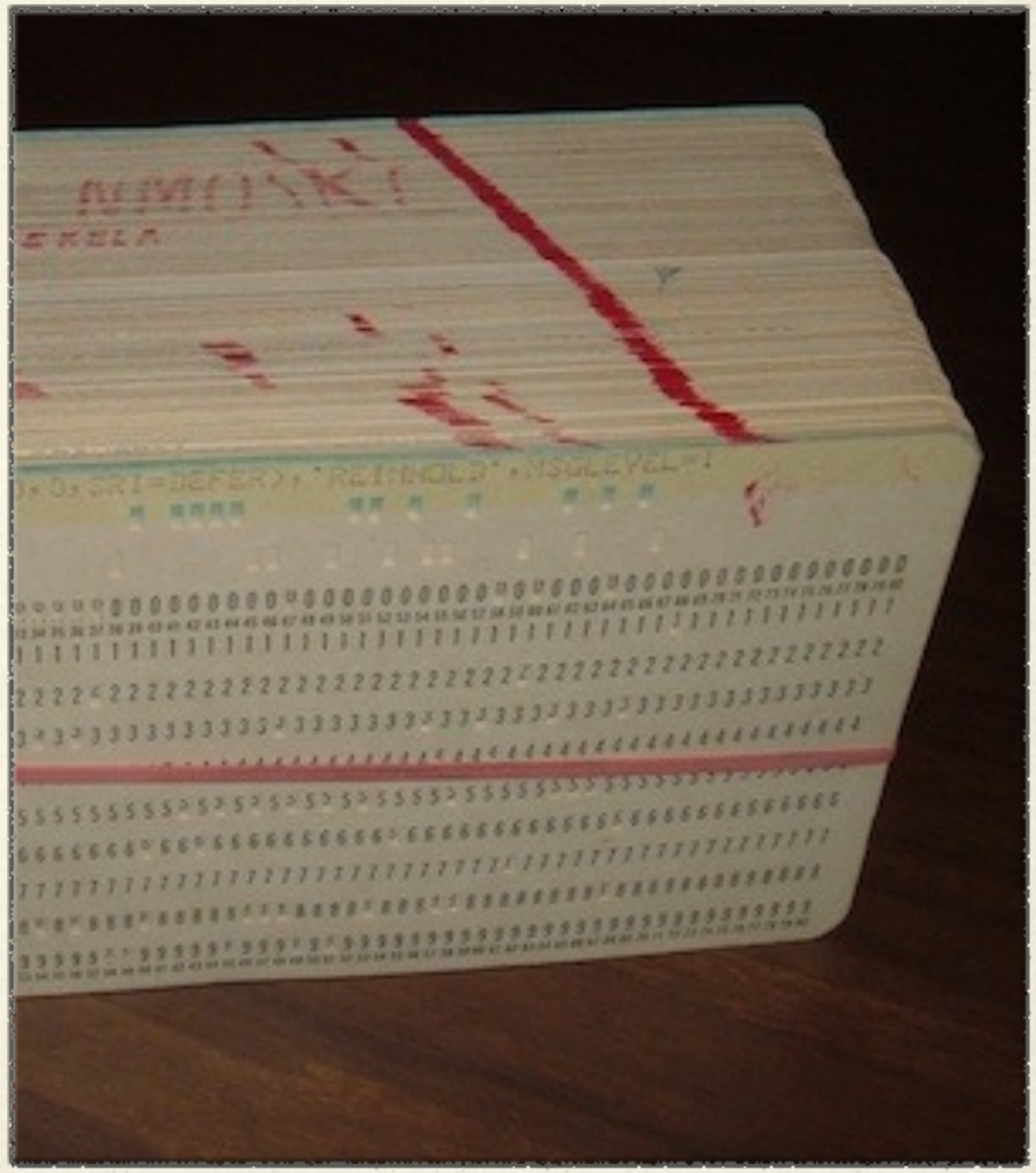
Code

Leslie Webb, 1922 to ___


SORTP
SKEL
PRINT
//ARNOLD JOB (M6920,7255,2,2000,0,SRI=DEFER),'REINHOLD',MSGLEVEL=1

Code was...

- Business requirements, written in the language of the computer
- Very hard and manual to change or debug
- Massive focus on optimisation and minimal use of computing resources
- Not that different to hardware



COBOL CODING SHEET

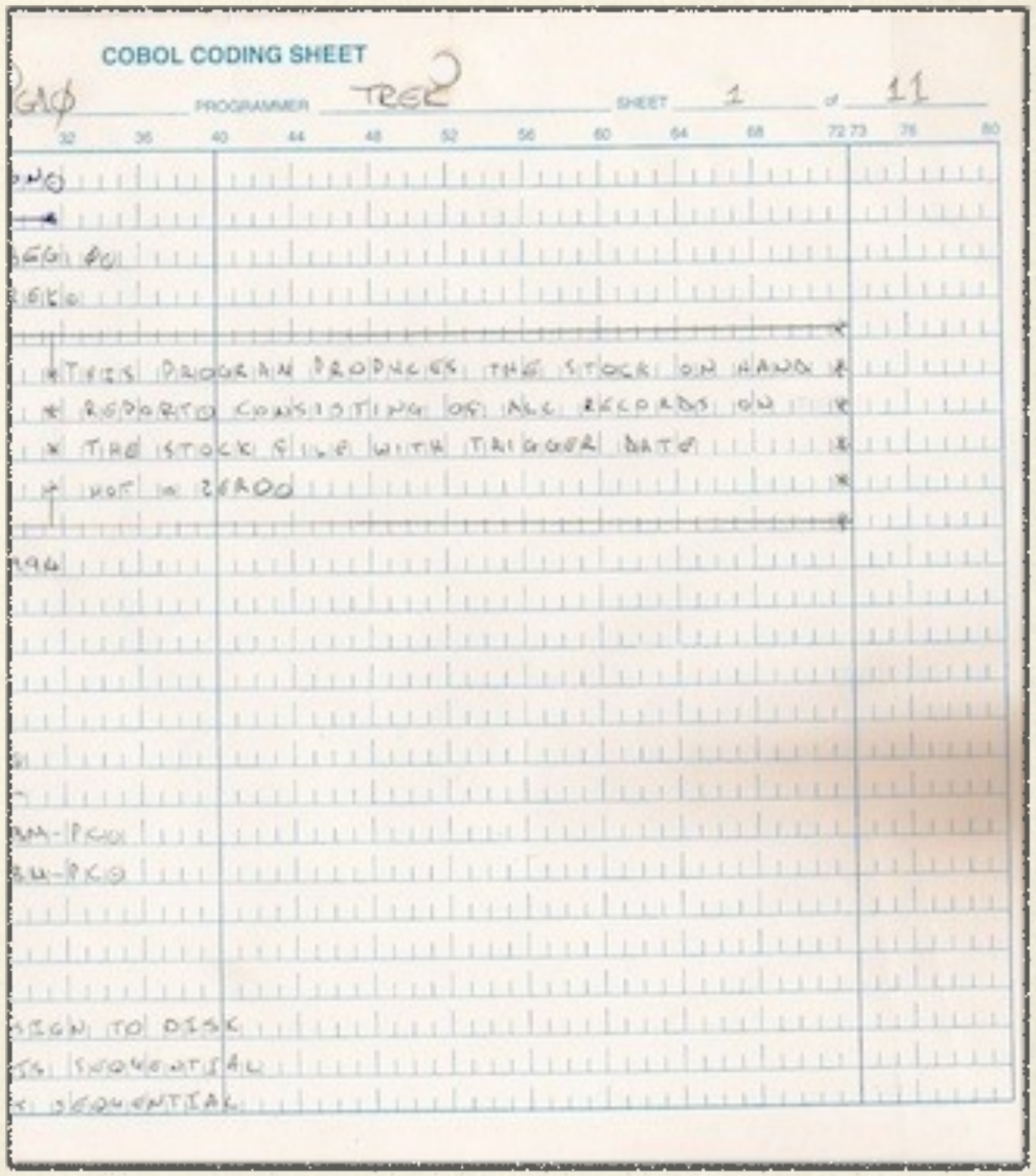
PROGRAM BBEG10 PROGRAMMER TREK SHEET 1 of 11

```
      IDENTISICATION DIVISIONO
     *-------------------------*
      PROGRAM IDO                BBEG10O
      AYTHORO                    TREKO
     *-------------------------------------------------------------*
     *                    *THIS PROGRAN PRODUCES THE STOCK ON HAND *
     *                    *REPORTO CONSISTING OF ALL RECORDS ON    *
     *                    *THE STOCK FILE WITH TRIGGER DATE        *
     *                    *NOT = ZEROO                             *
     *-------------------------------------------------------------*
      DATE WRITTENO
      DATE COMPILEDO
     *
      ENNIRONMENT DINISIONO
     *-------------------------*
      CONFIGURATION SECTIONO
     *-----------------------
      SONRCE COMPNTERO           IBM-PCO
      ORJECT COMPUTERO           IBM-PCO
      INPNT-OYTPNT SECTIONO
     *-----------------------
      FILE-CONTROLO
          SELECT STKFILE ASSIGN TO DISK
                 ORGANISATION IS SEQUENTIAL
                 ACCESS MODE IS SEQUENTIAL
```

N.J.H.

Code was...

- Business requirements, written in the language of the computer
- Very hard to change
- Syntax checkers, but still largely a manual exercise
- Massive focus on optimisation and minimal use of computing resources



VB6SimpleUpgrade - Microsoft Visual Basic [design] - [cSimpleUpgrade (Code)]

File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help

Ln 1, Col 1

(General) (Declarations)

```
Option Explicit

'All Variants are upgraded to the new Object Type.
Dim GlobalVariant As Variant
Dim mMyProperty As Integer

Private Sub BooleanUpgrade()

    'In your VB 6 projects you should use constant names instead of their
    'underlying values.  This protects you values of the constants from changing
    'in VB.Net.  One example is the constant value of True and False.  In VB6,
    'the underlying value of True has changed from -1 to 1.
    '
    'This statement would produce a "True" result in VB 6 and a "False" result
    'in VB.Net.  You should always use the constant names of True and False and
    'always use the Boolean data type to ensure a smooth upgrade.
    Dim i As Boolean
    i = True
    If i = True Then
        MsgBox ("True")
    Else
        MsgBox ("False")
    End If

End Sub

Private Sub DateUpgrade()

    'The .NET Framework provides the ToOADate and FromOADate functions to convert
    'between doubles and dates. However, when your project is upgraded to Visual
    'Basic.NET, it is difficult to determine the intention of code that uses doubles
    'to store dates. To avoid unnecessary modifications to your code in Visual Basic
    '.NET, always use the Date data type to store dates.

    Dim dblDate As Double
    Dim datDate As Date

    datDate = Now
```

Project - VB6SimpleUpgrade

VB6SimpleUpgrade (VB6Sir
Class Modules
cSimpleUpgrade (cSimpl

Properties - cSimpleUpgrade

cSimpleUpgrad ClassModule

Alphabetic | Categorized

(Name)	cSimpleUpgrade
DataBindingBehavic	0 - vbNone
DataSourceBehavic	0 - vbNone
Instancing	5 - MultiUse
MTSTransactionMoc	0 - NotAnMTSObj
Persistable	0 - NotPersistable

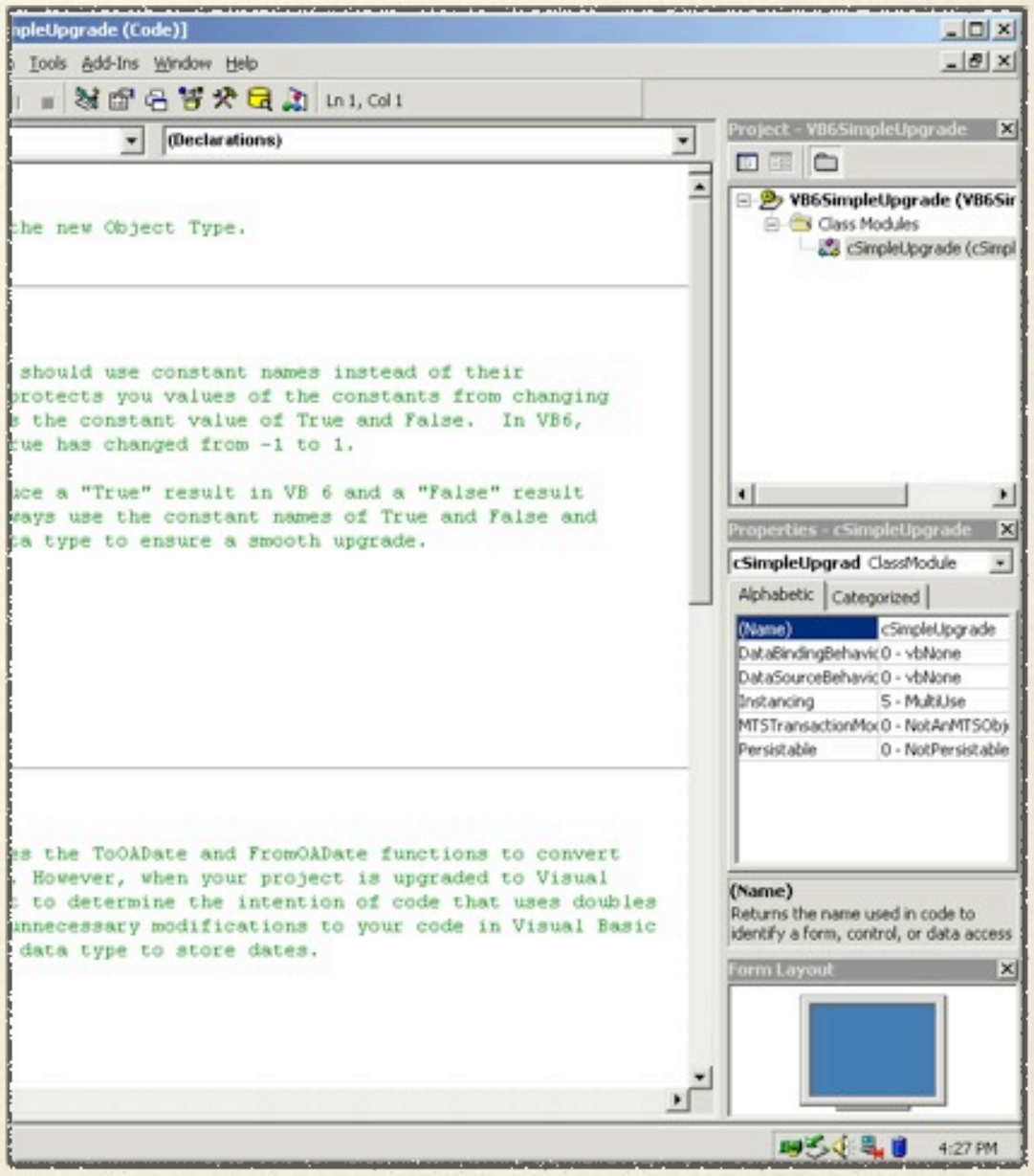
(Name)
Returns the name used in code to identify a form, control, or data access

Form Layout

Start VB6SimpleUpgrade - ... 4:27 PM

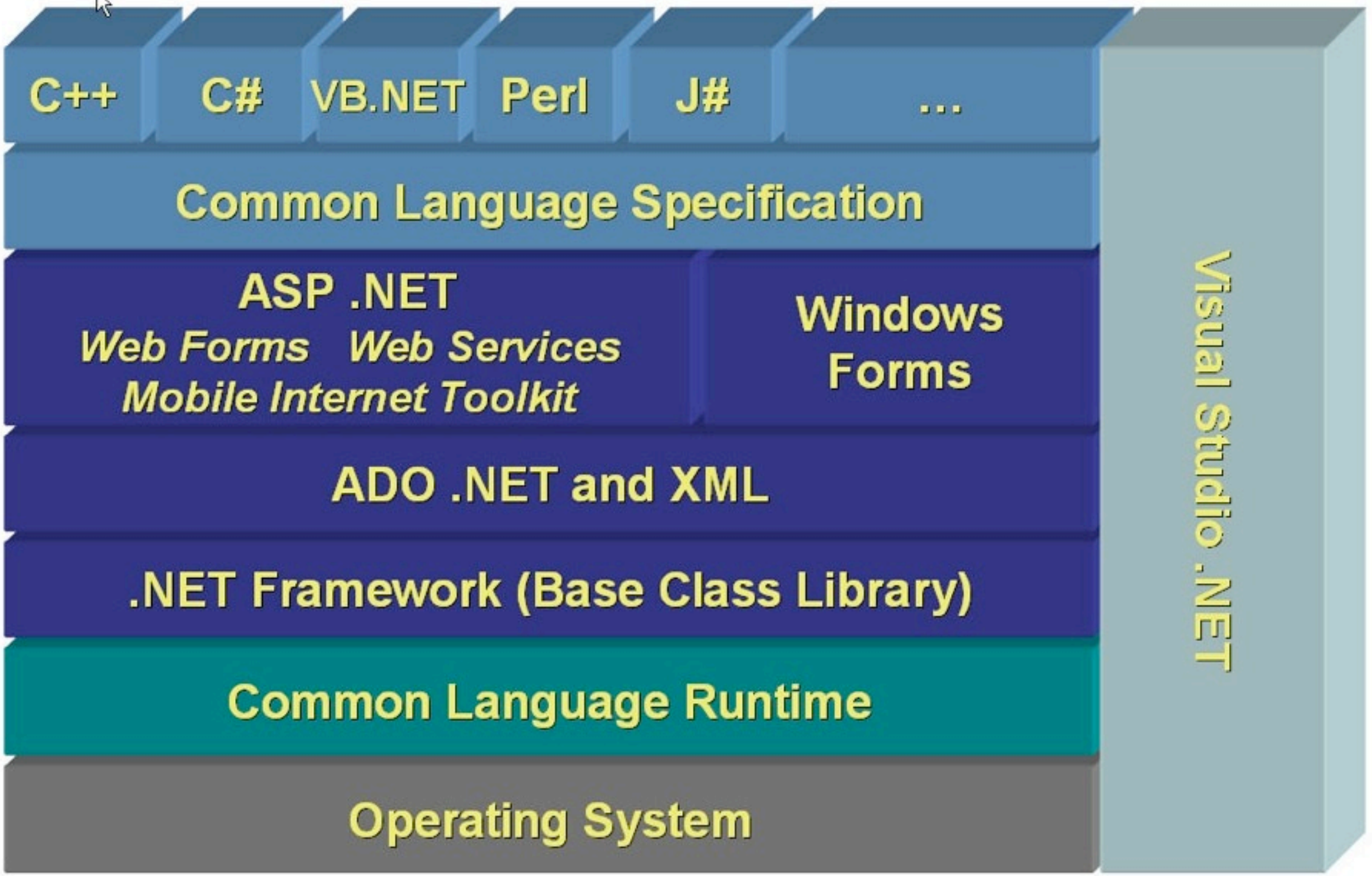
Code was...

- Business requirements, written (quickly) in the language of the computer
- Very easy to change
- Fairly easy to debug
- Still a focus on performance & optimisation, but storage now less of an issue



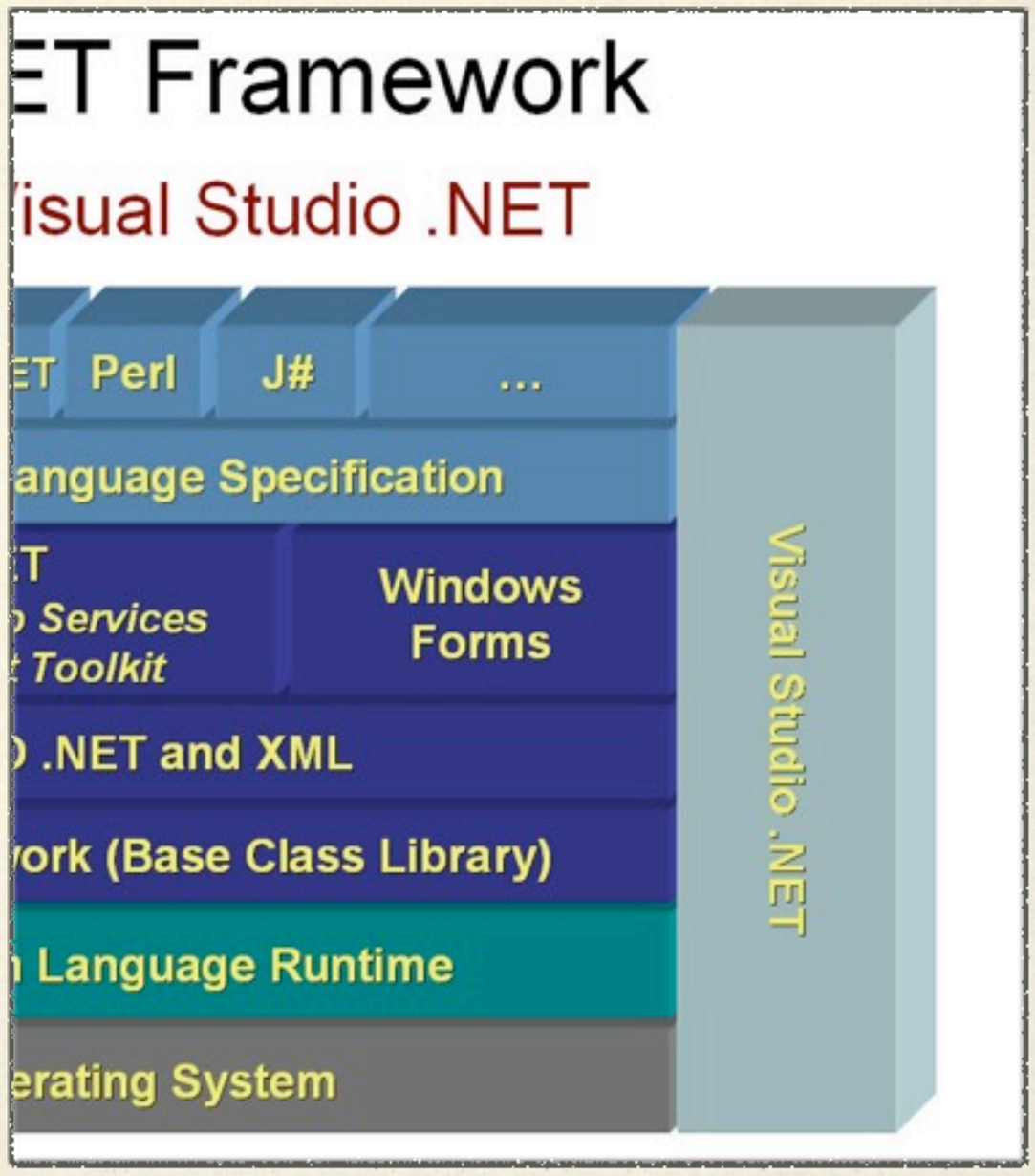
.NET Framework

Visual Studio .NET



Code was...

- Business requirements, written (slower) in the language of the computer
- Very easy to change
- Very easy to debug
- Performance delegated elsewhere



Manifesto for Agile Software Development

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.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas


BUFFALO
BUFFALO
micro
SD
HC
4

Fail

Fail

Baby Steps

Data Layer

Business Layer

Data Layer

Service Layer

Business Layer

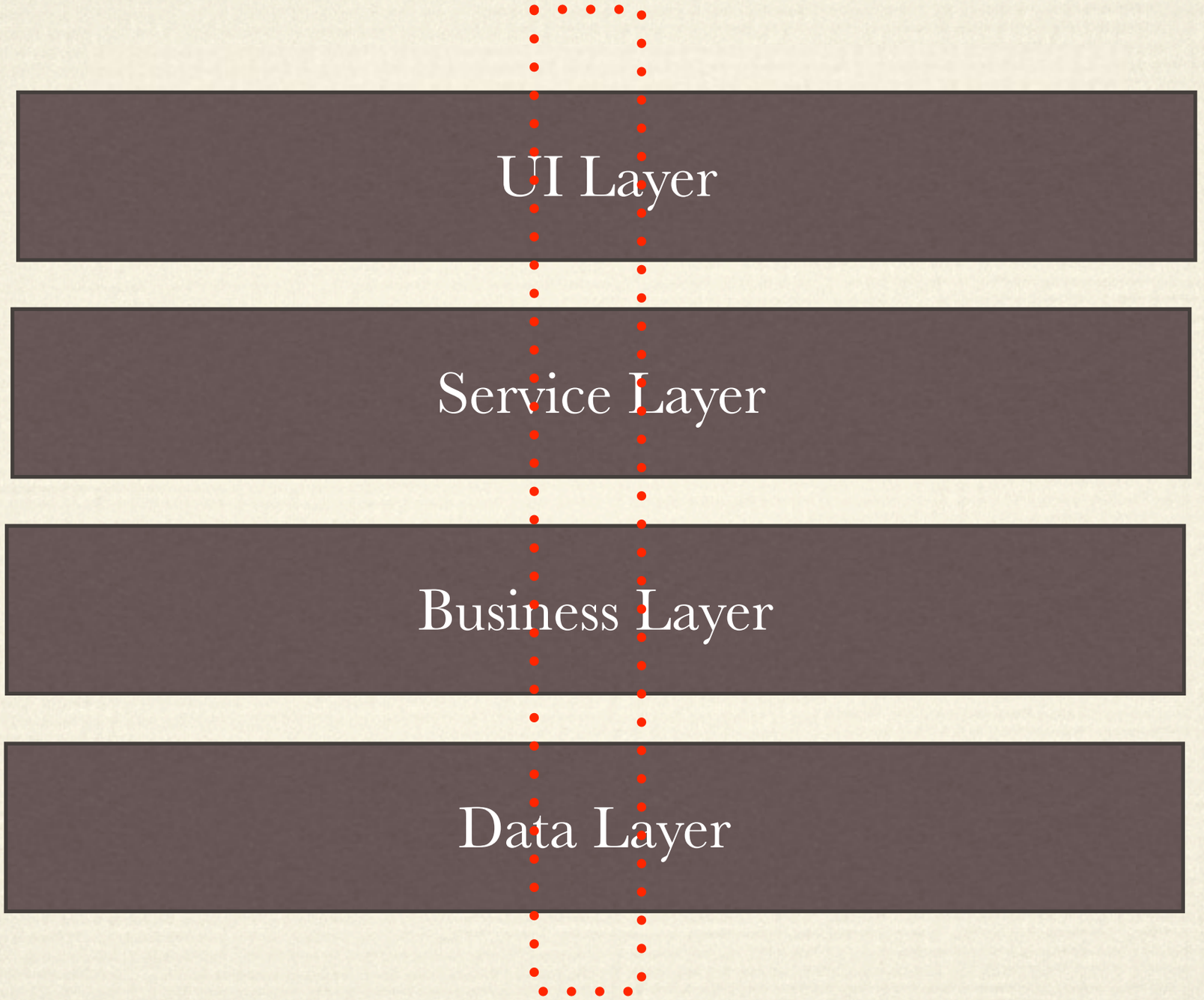
Data Layer

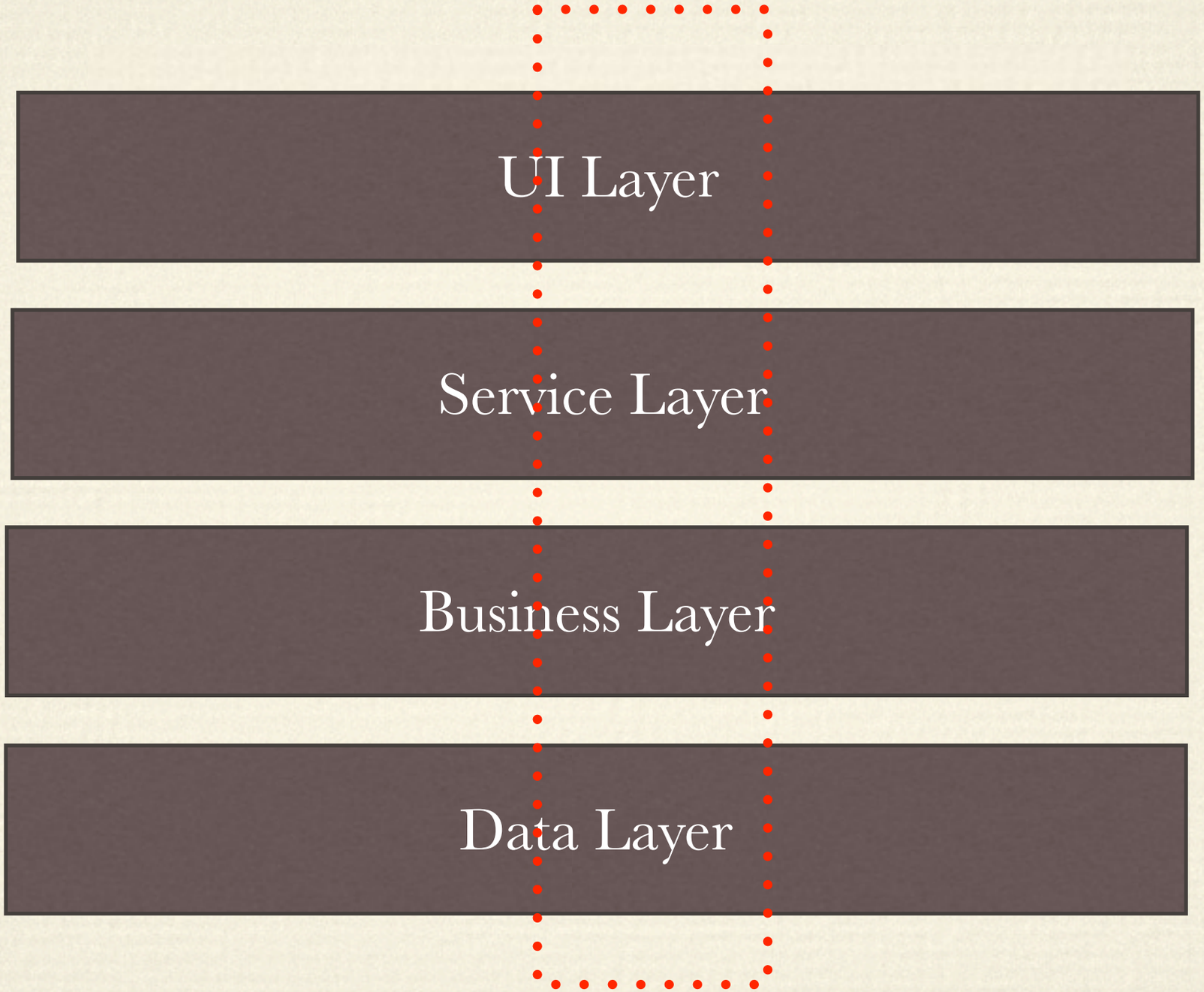
UI Layer

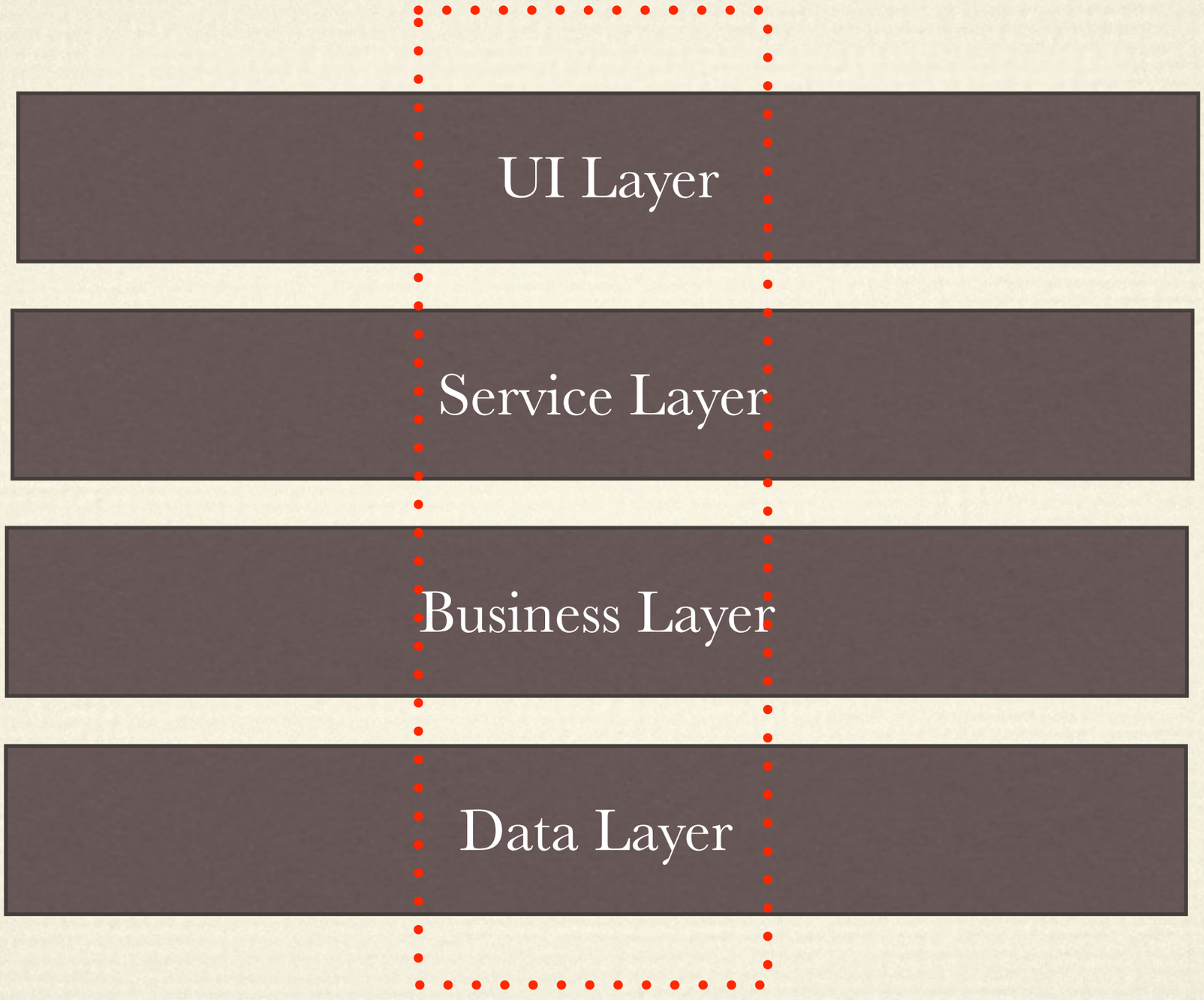
Service Layer

Business Layer

Data Layer


UI Layer
Service Layer
Business Layer
Data Layer


UI Layer
Service Layer
Business Layer
Data Layer


UI Layer
Service Layer
Business Layer
Data Layer


UI Layer
Service Layer
Business Layer
Data Layer

LONG RUNNING REPORT PROGRESS

8

G+H

AS A: ACCOUNTANT

I WANT: TO BE ABLE TO SEE THE PROGRESS OF MY LONG RUNNING REPORT

SO THAT: I KNOW HOW LONG I MUST WAIT FOR IT.

KB-589

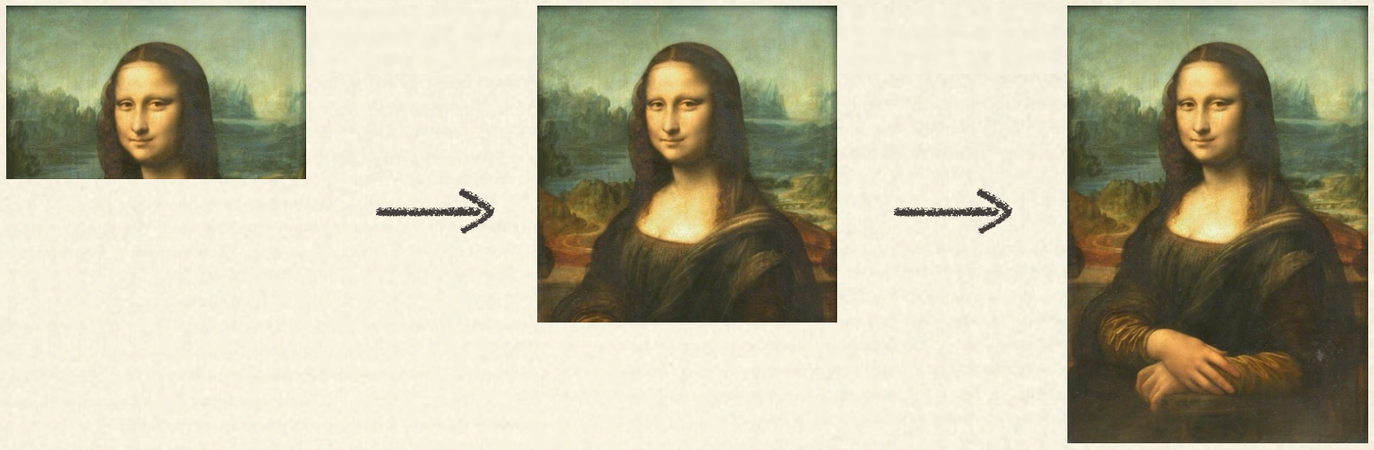
Fail

Baby Steps

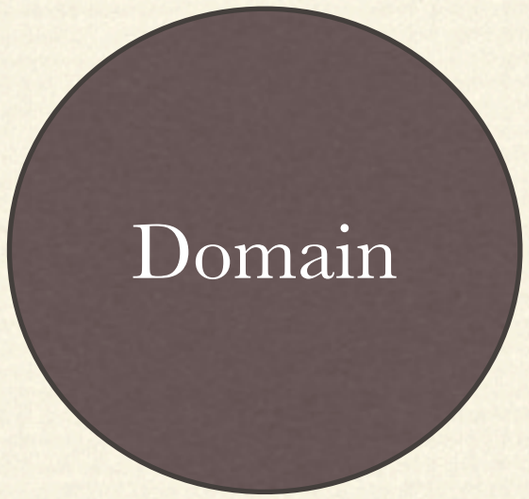
Iterative

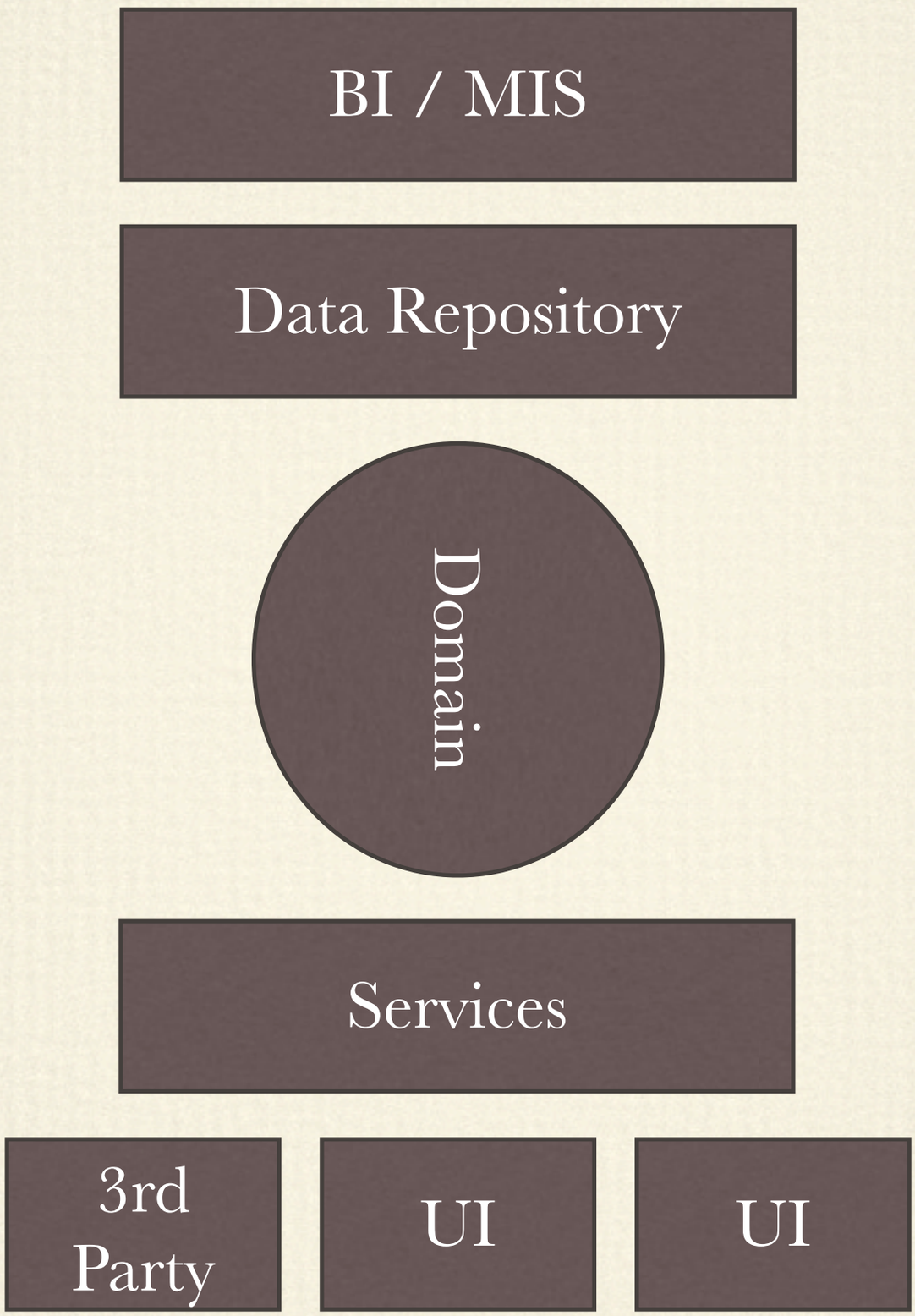
Iteration 01 Iteration 02 Iteration 03

Incremental



Iterative


Domain


BI / MIS
Data Repository
Domain
Services
3rd Party
UI
UI

Fail

Baby Steps

Iterative

Continuous Improvement

Fail

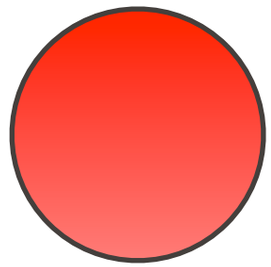
Baby Steps

Iterative

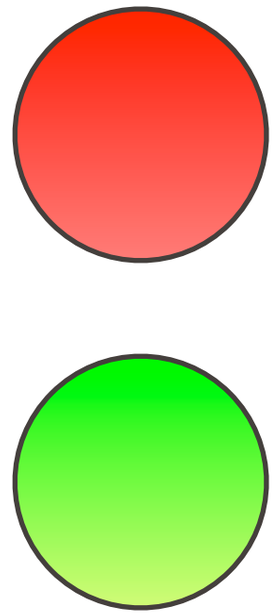
Continuous Improvement

Craft

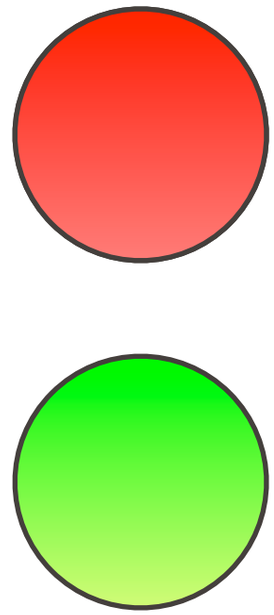
```
[TestFixture]
public class AccountTests
{
    [Test]
    public void CanCreateAccountWithOpeningBalance ()
    {
        // Arrange
        int balance = 1000;

        // Act
        Account a = new Account (balance);

        // Assert
        Assert.That (a.CurrentBalance, Is.EqualTo (balance));

    }
}
```



```
[TestFixture]
public class AccountTests
{
    [Test]
    public void CanCreateAccountWithOpeningBalance ()
    {
        // Arrange
        int balance = 1000;

        // Act
        Account a = new Account (balance);

        // Assert
        Assert.That (a.CurrentBalance, Is.EqualTo (balance));

    }
}

public class Account
{
    int _balance = 0;

    public Account (int balance)
    {
        _balance = balance;
    }

    public int CurrentBalance
    {
        get { return _balance; }
    }
}
```



```
[TestFixture]
public class AccountTests
{
    [Test]
    public void CanCreateAccountWithOpeningBalance ()
    {
        // Arrange
        int balance = 1000;

        // Act
        Account a = new Account (balance);

        // Assert
        Assert.That (a.CurrentBalance, Is.EqualTo (balance));

    }
}

public class Account
{
    int _balance = 0;

    public Account (int balance)
    {
        _balance = balance;
    }

    public int CurrentBalance
    {
        get { return _balance; }
    }
}
```



Refactor!

```
public class Account2
{
    int _balance = 0;

    public Account2 (int balance)
    {
        _balance = balance;
    }

    /// <summary>
    /// Gets the current balance of the Account
    /// </summary>
    /// <value>
    /// The current balance.
    /// </value>
    public int CurrentBalance {
        get { return _balance; }
    }
}
```

```
/// <summary>
/// Makes the withdrawal from the Account.
/// </summary>
/// <param name='amount'>
/// The amount to withdraw (in cents)
/// </param>
public void MakeWithdrawal (int amount)
{
    // Reduce balance by specified amount
    _balance = _balance - amount;

    // Send notification to the account holder
    SmsNotifier notify = new SmsNotifier ();
    notify.Send (
        _accountHolder.PhoneNumber,
        string.Format ("{0} was withdrawn from your account. Your new balance is {1}.", amount, _balance)
        );

}
```

```
public void MakeWithdrawal (int amountInCents)
{
    reduceBalanceBy (amountInCents);
    notifyAccountHolder (amountInCents, _balance);
}

private void reduceBalanceBy (int amountInCents)
{
    _balance = _balance - amountInCents;
}

private void notifyAccountHolder (int amountInCents, int newBalance)
{
    SmsNotifier notify = new SmsNotifier ();
    notify.Send (
        _accountHolder.PhoneNumber,
        string.Format ("{0} was withdrawn from your account. Your new balance is {1}.",
        amountInCents, newBalance) );
}
```

```
public class Account4
{
    Person _accountHolder;
    int _balance = 0;

    public Account4 (Person accountHolder, int balance)
    {
        _accountHolder = accountHolder;
        _balance = balance;
    }

    public int CurrentBalance {
        get { return _balance; }
    }

    public void MakeWithdrawal (int amountInCents)
    {
        reduceBalanceBy (amountInCents);
        notifyAccountHolder (amountInCents, _balance);
    }

    private void reduceBalanceBy (int amountInCents)
    {
        _balance = _balance - amountInCents;
    }

    private void notifyAccountHolder (int amountInCents, int newBalance)
    {
        SmsNotifier notify = new SmsNotifier ();
        notify.Send (
            _accountHolder.PhoneNumber,
            string.Format ("{0} was withdrawn from your account. Your new balance is {1}.",
            amountInCents, newBalance) );
    }
}
```

```
public class Account5
{
    Person _accountHolder;
    int _balance = 0;
    INotifier _notifier;

    public Account5 (Person accountHolder, int balance, INotifier notifier)
    {
        _accountHolder = accountHolder;
        _balance = balance;
        _notifier = notifier;
    }

    public int CurrentBalance {
        get { return _balance; }
    }

    public void MakeWithdrawal (int amountInCents)
    {
        _balance = _balance - amountInCents;

        notifyAccountHolder (amountInCents, _balance);
    }

    public void notifyAccountHolder (int amountInCents, int newBalance)
    {
        _notifier.Send (
            _accountHolder.PhoneNumber,
            string.Format ("{0} was withdrawn from your account. Your new balance is {1}.",
            amountInCents, newBalance));
    }
}
```

Fail

Baby Steps

Iterative

Continuous Improvement

Craft

See Code Differently...

- Business requirements, written in the language of the computer
- Very hard and manual to change or debug
- Massive focus on optimisation and minimal use of computing resources
- Not that different to hardware

- Business requirements, **written in the language of the business**
- A real-time feedback mechanism to drive change
- Massive focus on Self Documenting, Self Testing code
- **Soft**ware

Today, Code is...

Today, Code is...

Alive

Domain Driven Design - Eric Evans

The Art of Agile - James Shore & Shane Warden

Clean Code - Robert C Martin

Driven Software

kevint@drivensoftware.net

@kevintrethewey