

Code Quality

Steigerung der Codequalität mit
Visual Studio & TFS

Tobias Richling



- 30 Jahre
- Wohnhaft im Münsterland
- Softwareentwickler
seit Amiga Basic
- „Microsoftie“
- Logisiksoftware
 - Silverlight
 - TFS
- Trainer, Speaker, Autor

Agenda



Clean Code



Code Analysis



Checkin Policies

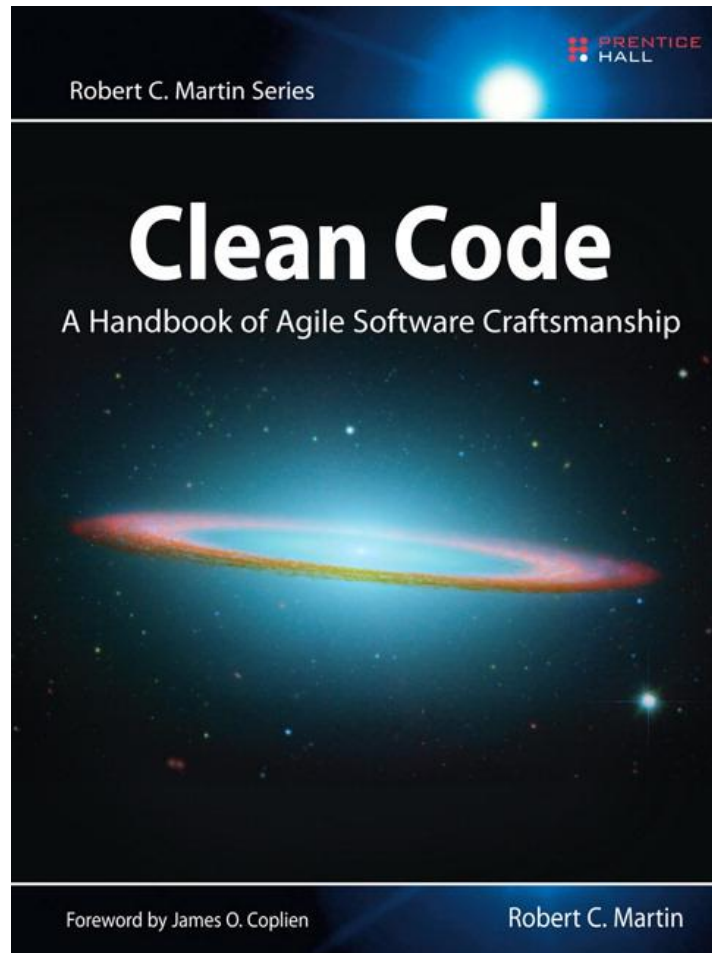


Ausblick: Roslyn



CLEAN CODE

Wer kennt dieses Buch?



„Uncle Bob“

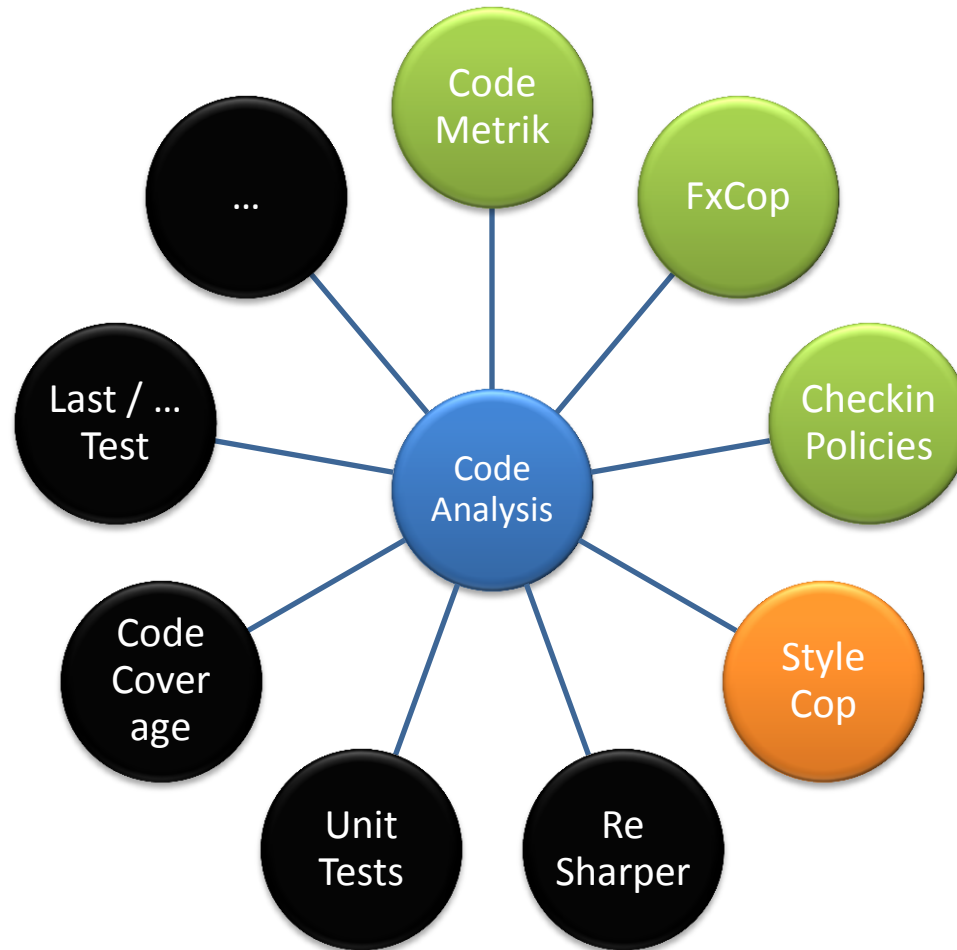
Einige Beispiele für Clean Code

- Funktionen
 - Einheitliches Abstraktionsniveau
 - KURZ!
- Law of Demeter
 - Keine Referenzketten
 - Schüchterner Code
- Tell dont ask
 - Öffentliche Properties vermeiden



CODE ANALYSIS

Code Analysis ist ein weites Feld



Einige Spielarten der Code Analyse

- Code Metriken
 - Berechnung von Kennzahlen über den Code
- Visual Studio Code Analysis
 - Prüfung von Architektur- / Sicherheits-Konventionen
 - basiert auf FxCop
- Checkin Policies
 - Prüft Dateien / Formalien beim Checkin
- StyleCop
 - Prüfung von Code-Konventionen

Code Metriken

- Maintainability Index
 - Zusammengesetzter Wert
- Cyclomatic Complexity
 - Wie viele Pfade gibt es durch eine Methode
- Depth of inheritance
 - Vererbungstiefe
- Class coupling
 - Wie viele Klassen werden verwendet
- Lines of code
 - Codezeilen ohne Kommentare

StyleCop vs. FxCop

StyleCop

- Kein MS Produkt
- Nicht in VS integriert
- Source Code Analyse (Text)
- Derzeit nur C#
- Kann z.B. feststellen
 - Stellung von {/}
 - Namenskonventionen
 - Leerzeilen

```
PrisonersDilemmaAllInOne.cs  Resolve Conflicts  Prisoner.cs
Dojo.PrisonersDilemma.SingleFile.Strati  AlwaysCooperate

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Dojo.PrisonersDilemma.SingleFile
{
    public enum Strategy
    {
        AlwaysCooperate,
        AlwaysDefect,
        TitForTat,
        Random
    }

    public class Prisoner
    {
        private Random rnd = new Random(DateTime.Now.Ticks);
        int pOverallPenalty = 0, p2OverallPenalty = 0;
        bool? pLastDecision = null, p2LastDecision = null;
        Strategy p1Strategy, p2Strategy;

        public void Run(Strategy p1Strategy, Strategy p2Strategy,
            int numRounds)
        {
            Run(p1Strategy, p2Strategy, nextRoundProb, 1, 2, 4, 6);
        }

        public void Run(Strategy p1Strategy, Strategy p2Strategy,
            double nextRoundProb, int p1, int p2, int p4, int p6)
        {
            if (!temptation > reward && reward > punishment && punishment > suckerPayoff)
                throw new Exception("Invalid payment schema");

            this.p1Strategy = p1Strategy;
            this.p2Strategy = p2Strategy;
            this.temptation = temptation;
            this.reward = reward;
            this.punishment = punishment;
            this.suckerPayoff = suckerPayoff;

            Console.WriteLine("Using {0} number of rounds game location", numRounds);
            Console.WriteLine("Prisoner 1 using {0} strategy", p1Strategy);
            Console.WriteLine("Prisoner 2 using {0} strategy", p2Strategy);
            Console.WriteLine();

            RunNextRound();

            Console.WriteLine("p1 penalty {0} - p2 penalty {1}", p1OverallPenalty, p2OverallPenalty);
        }

        public void Run(Strategy p1Strategy, Strategy p2Strategy,
            int numRounds, double nextRoundProb)
        {
            Run(p1Strategy, p2Strategy, numRounds, 1, 2, 4, 6);
        }

        public void Run(Strategy p1Strategy, Strategy p2Strategy,
            double nextRoundProb, int p1, int p2, int p4, int p6)
        {
            if (!temptation < reward && reward < punishment && punishment > suckerPayoff)
                throw new Exception("Invalid payment schema");
        }
    }
}
```

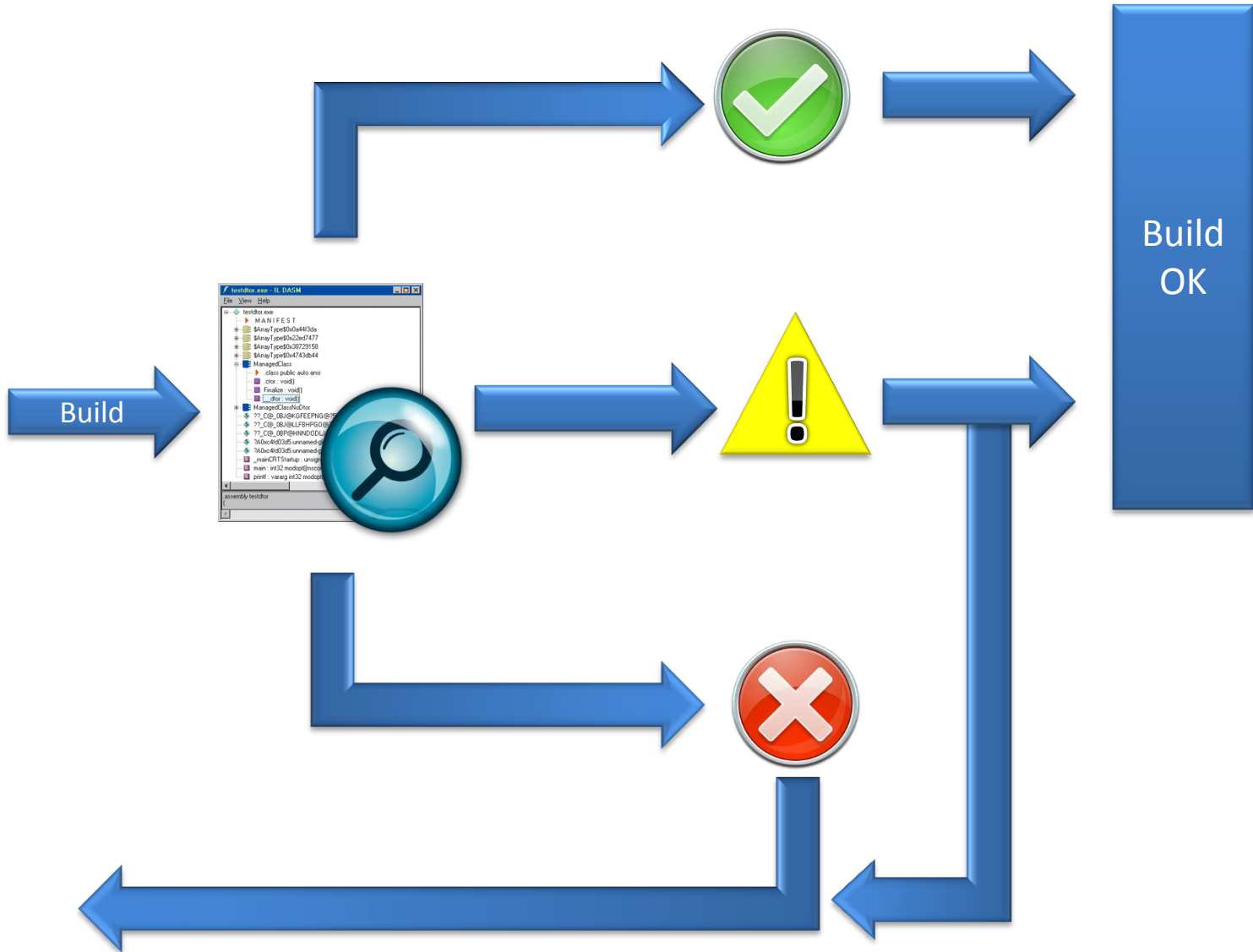
FxCop

- MS Produkt
- Integriert in VS
- IL Code Analyse (Opcodes)
- Alle IL Sprachen
- Kann z.B. feststellen
 - Strong Name
 - Boxing / Unboxing
 - Typ geworfener Exceptions

```
testdtor.exe - IL DASM
File View Help
testdtor.exe
  MANIFEST
  class public auto ansi
    Finalize: void()
    ctor: void()
    77_C0_0BJ@KGFEENP@75ManagedClass?%CI?%CFp?%CJ?
    77_C0_0BJ@LFF@P@75ManagedClass?%CI?%CFp?%CJ?
    740xc4fd03d5.unnamed-global-0: public static valuetype $Array1
    740xc4fd03d5.unnamed-global-1: public static valuetype $Array1
    _mainCRTStartup: unsigned int32[]
    mainCRTStartup: unsigned int32[]
    print: vararg int32 modopt([mscorlib]System.Runtime.CompilerServices.C
    print: vararg int32 modopt([mscorlib]System.Runtime.CompilerServices.C
assembly testdtor
(
```

Es gibt Überschneidungen!

FxCop - Ablauf



Out of the box

- FxCop seit VS2010 teil des Visual Studio
 - C:\Program Files (x86)\Microsoft Visual Studio 11.0\Team Tools\Static Analysis Tools\FxCop
- Große Anzahl an vorgefertigten Regeln
 - Thematisch gruppiert
- Einige Regeln für den Anfang auswählen
 - Sonst kann man „erschlagen“ werden

Beispielhafte Coderegeln

- Übermäßige Komplexität vermeiden
 - Zyklomatische Komplexität
- Ungenutzte Methoden entfernen
- Ungenutzte Felder entfernen
- Methoden statisch machen
 - Falls Methode keine Klassenmember nutzt, sollte diese statisch definiert werden

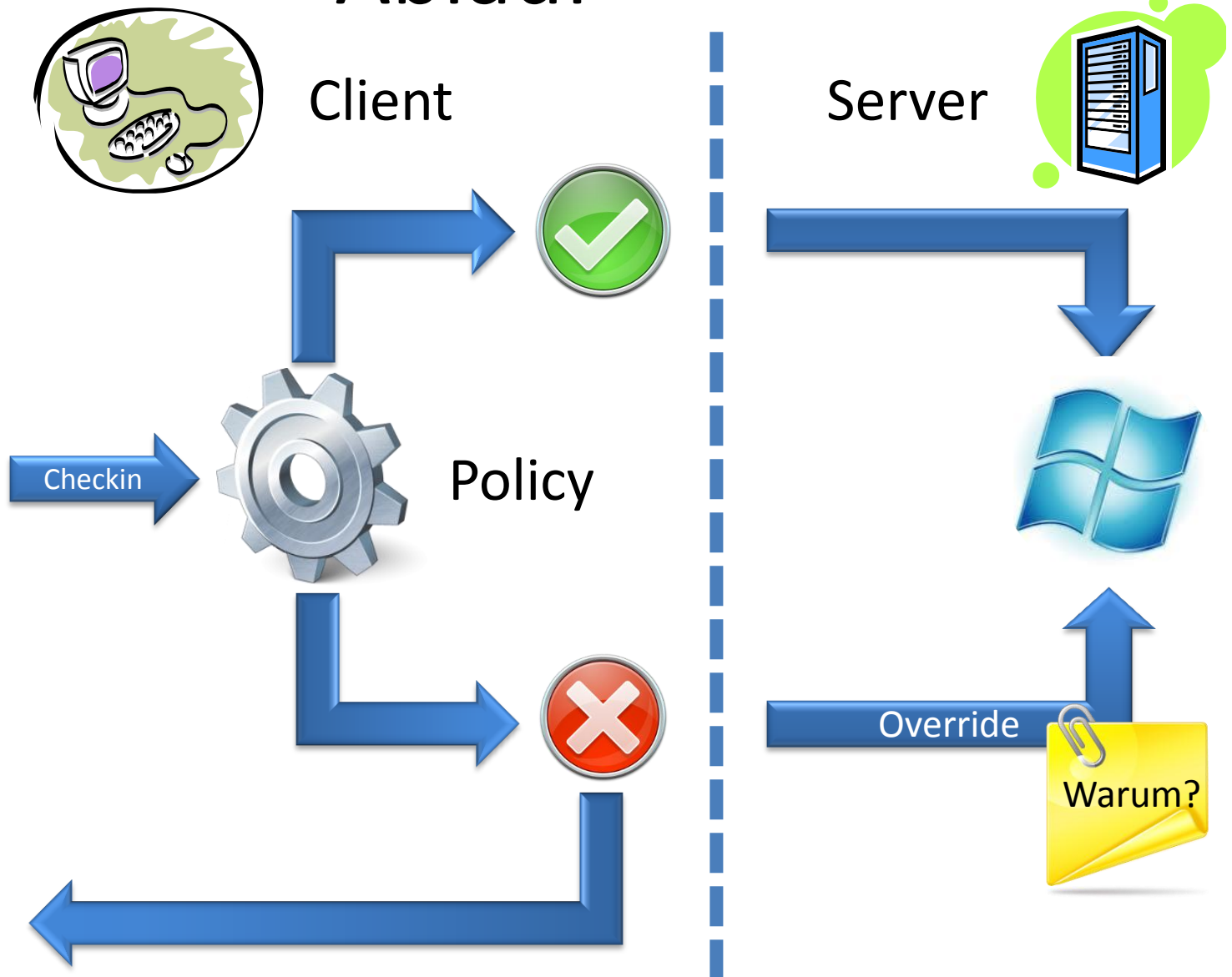
Eigene Regeln entwickeln

- Ableiten von BaseIntrospectionRule
- Überschreiben der Check-Methode
 - Untersuchung von Typen, Members etc.
 - Reflection-Level analyse
- Überschreiben der Visit-Methode
 - Untersuchung des IL Codes
 - Introspection
 - Visitor-Pattern



CHECKIN POLICIES

Ablauf



Out of the Box

- Eine Handvoll Policies werden mit geliefert
 - Work Item
 - Code Comment
- Die TFS Power Tools bringen einige weitere
 - Work Item Query
 - Custom Path
- Diese kann man auch mit TFS 11 Beta verwenden

Eigene Checkin Policies entwickeln

- Implementiere
 - IPolicyDescription & IPolicyExecution oder
 - Erben von PolicyBase
- Deployment auf jeden Client!
 - Z. B. einchecken im TFS
- Registrieren in der Registry

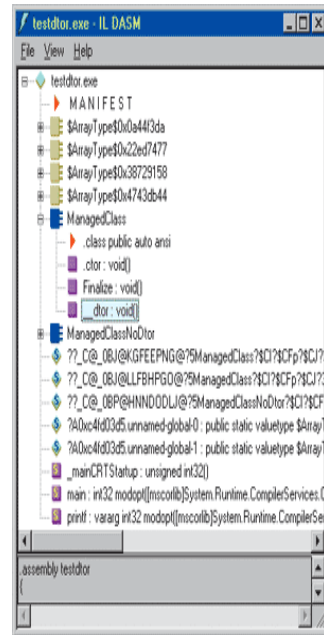


ROSLYN

Compiler Pipeline



Compiler
(Black Box)



Compiler API's

Parser

Symbols

Binder

IL Emitter

Syntax Tree API

- Outlining
- Colorizer
- Formatter

Symbol API

- Navigate To
- Object Browser
- Completion List

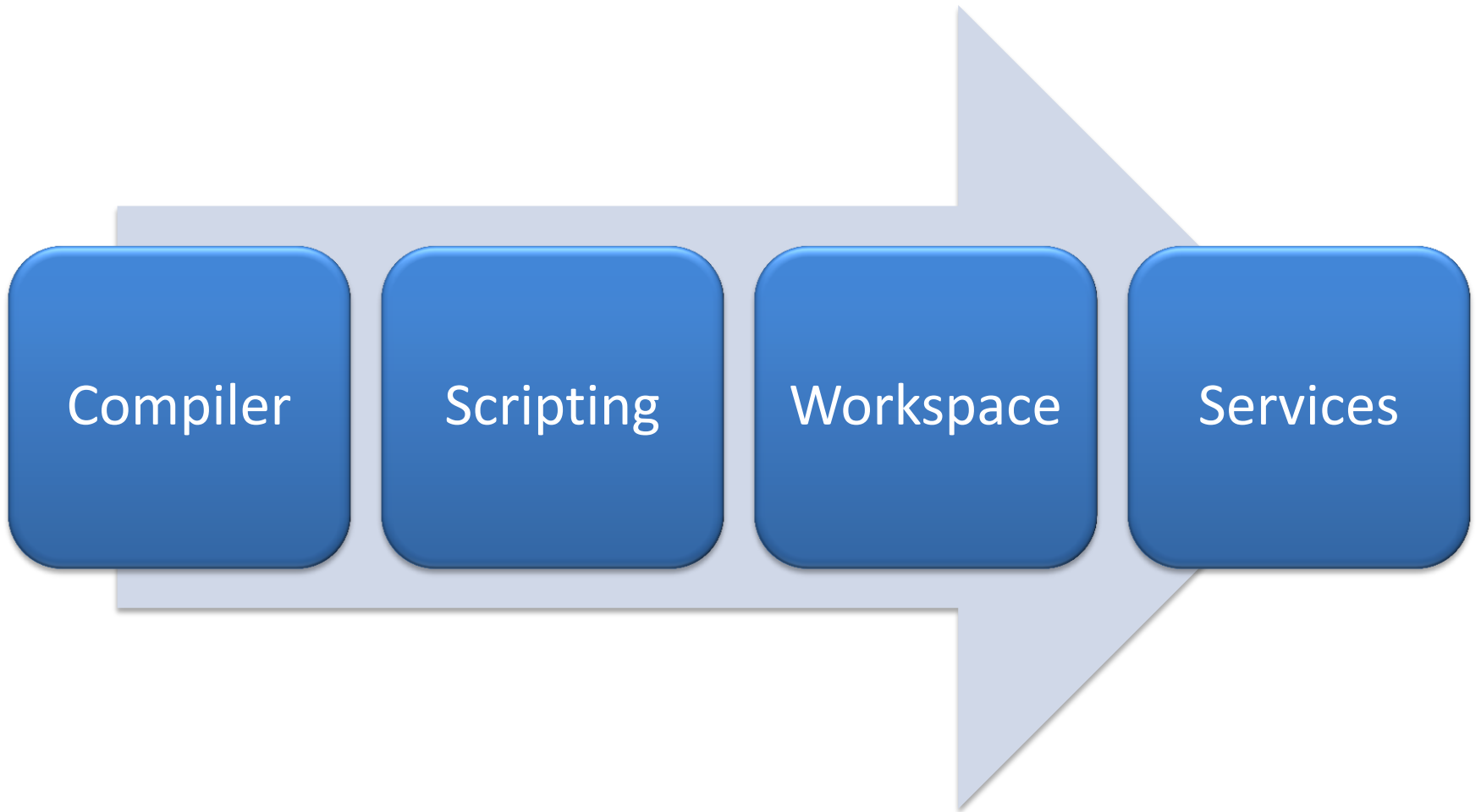
Binding & Flow API

- Extract Method
- Find References

Emit API

- Edit and Continue

Roslyn API Layers



Was geht?

- Code Analyse „on the fly“
 - a la Resharper
- Automatische Korrektur von Code Issues
 - a la Visual Studio Smart Tag
- Eigene Refactorings bauen

FAZIT

Fazit

- Code Analysis
 - „Code Knigge“ – unerlaubtes kritisieren
 - API schlecht dokumentiert
- Checkin Policies
 - „Türsteher“ – unerlaubtes nicht rein lassen
 - Deployment ist umständlich
- Roslyn
 - Cool! C# Interactive Window!
 - Abwarten und Tee trinken :-)

Noch Fragen



Vielen Dank für Ihre
Aufmerksamkeit

