

Projekt: wt4u  
  
Metriken

Andreoli Dario (dandreol@hsr.ch)

Schiepek Richard (rschiepe@hsr.ch)

Zahner Tobias (tzahner@hsr.ch)

# Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 22.05.2014 | 1.0 | Inititaldokument | rschiepe |
| 24.05.2014 | 1.1 | Ergänzung um Kapitel Structure101 | rschiepe |

# Inhalt

[Änderungsgeschichte 2](#_Toc388969887)

[Inhalt 3](#_Toc388969888)

[1. Codestatistik 4](#_Toc388969889)

[1.1 Kennzahlen von Visual Studio 4](#_Toc388969890)

[1.2 Kennzahlen von Ndepend 4](#_Toc388969891)

[2. Analyse mit Structure101 5](#_Toc388969892)

[2.1 Abhängigkeiten aller Packages 5](#_Toc388969893)

[2.1.1 Fazit / Schlussfolgerung 5](#_Toc388969894)

[2.2 Abhängigkeiten des Packages BusinessAccess 5](#_Toc388969895)

[2.2.1 Fazit / Schlussfolgerung 5](#_Toc388969896)

[3. Analyse mit Ndepend 6](#_Toc388969897)

[**3.1** Grössen der Klassen/Methoden 6](#_Toc388969898)

[3.1.1 Fazit / Schlussfolgerung 6](#_Toc388969899)

[3.2 Abhängigkeiten auf fremde Packages (Microsoft) 7](#_Toc388969900)

[3.2.1 Fazit / Schlussfolgerung 7](#_Toc388969901)

[3.3 Distance Diagramm 8](#_Toc388969902)

[3.3.1 Fazit / Schlussfolgerung 8](#_Toc388969903)

[3.4 Codeanalyse, Kritische Sektionen 9](#_Toc388969904)

[3.4.1 Method too complex – critcal 9](#_Toc388969905)

[3.4.2 Methods with too many parameters – critical 9](#_Toc388969906)

# Codestatistik

Zur Erinnerung: Unsere .NET Solution besteht aus den 2 Projekten wt4u und wt4u.Testing.

Sämtlicher HTML- und Javascriptcode wurde bei dieser Codestatistik nicht eingerechnet.

## Kennzahlen von Visual Studio

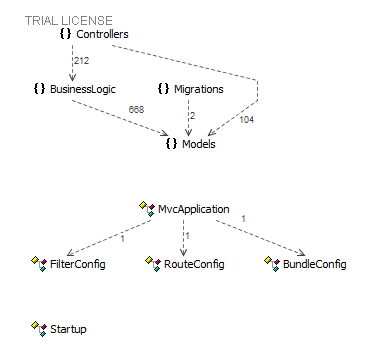
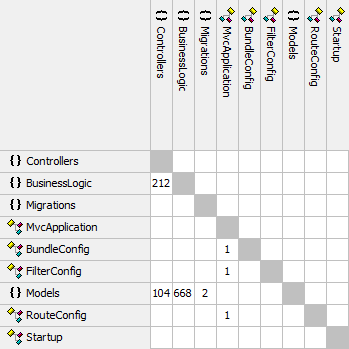
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Beschreibung | Zyklomatische Komplexität | Vererbungstiefe | Klassenkopplung | Zeilen an Code |
| Solution | 1179 | 5 | 321 | **2683** |
|  |  |  |  |  |
| Projekt wt4u | 1024 | 4 | 268 | 2171 |
| Projekt wt4u.Testing | 155 | 1 | 53 | 512 |
|  |  |  |  |  |
| **Details Projekt wt4u** |  |  |  |  |
| Wt4u | 16 | 2 | 33 | 31 |
| Wt4u.BusinessLogic | 502 | 1 | 133 | 1452 |
| Wt4u.Controllers | 324 | 4 | 131 | 493 |
| Wt4u.Migrations | 2 | 3 | 3 | 2 |
| Wt4u.Models | 180 | 4 | 37 | 193 |

## Kennzahlen von Ndepend

|  |  |
| --- | --- |
| Beschreibung | Anzahl |
| Assemblies | 2 |
| Namespaces | 7 |
|  |  |
| **Total (inkl. Allen Microsoft Libraries)** |  |
| Klassen | 170 |
| Methoden | 581 |
|  |  |
| **Wt4u** |  |
| Klassen | 18 |
| Klassenvariablen (Fields) | 70 |
| Methoden | 142 |
|  |  |
| **Wt4u.Testing** |  |
| Klassen | 6 |
| Klassenvariablen (Fields) | 33 |
| Methoden | 119 |

# Analyse mit Structure101

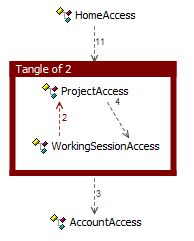
## Abhängigkeiten aller Packages



### Fazit / Schlussfolgerung

Aufgrund der einfachen Package Struktur sind keinerlei zyklische Abhängigkeiten vorhanden (auch fast nicht möglich). Der Controller hat eine Abhängigkeit auf das Models Packages aufgrund der Nutzung des Entity Frameworks.

## Abhängigkeiten des Packages BusinessAccess

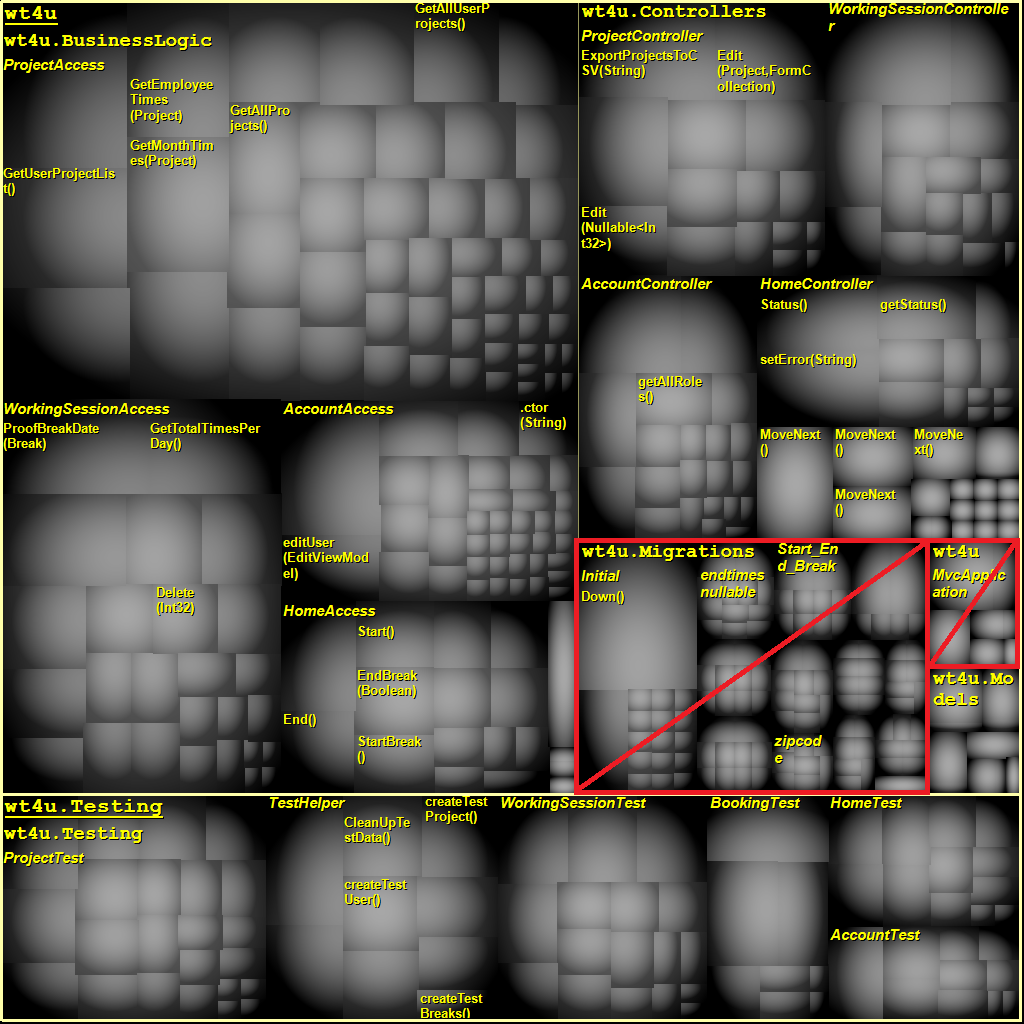


### Fazit / Schlussfolgerung

Innerhalb des Packages BusinessAccess in dem sich die ganze Logik der Applikation befindet gibt es nur sehr wenige Abhängikeiten. Diese wurde bewusst minimal gehalten um Bei eine Instanzierung einer Accessklasse durch den Controller, weitere Instanzierungen innerhalb zu verhindern.

# Analyse mit Ndepend

## Grössen der Klassen/Methoden

****

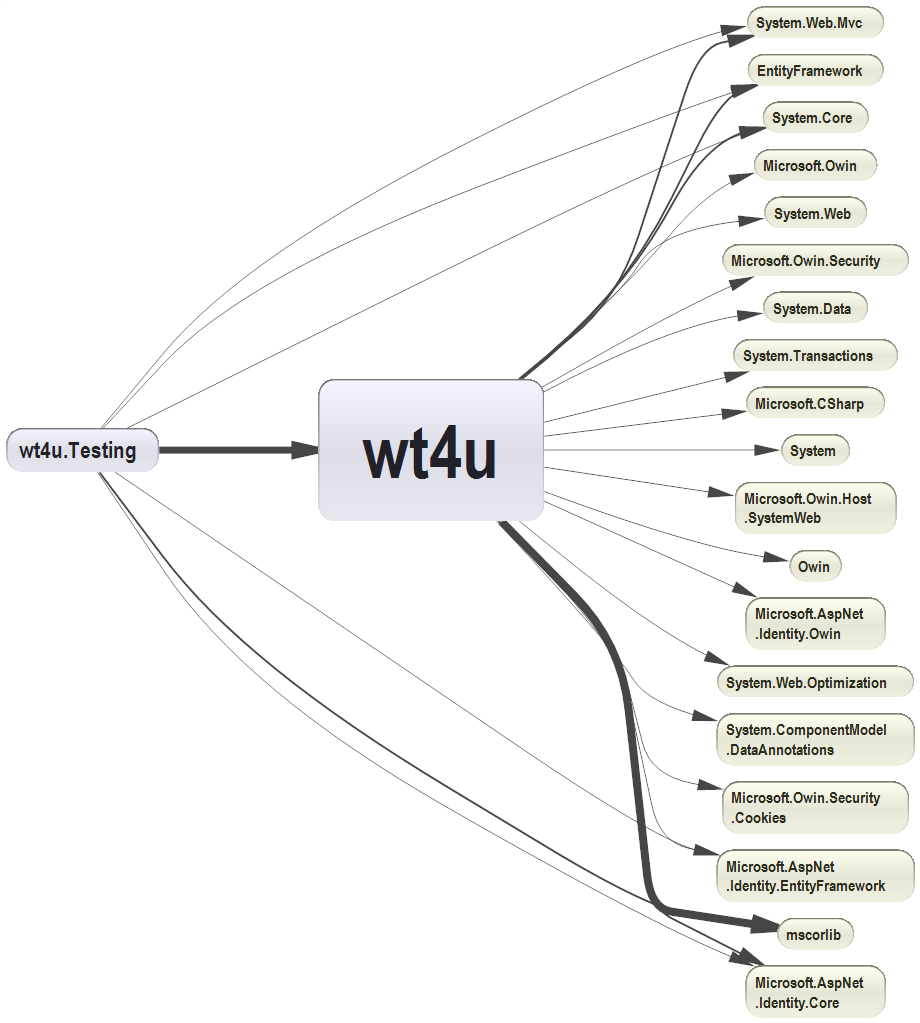
### Fazit / Schlussfolgerung

Die BusinessLogic ist das eigentliche Herzstück der Applikation und nimmt wie erwartet am meisten Platz für den Code ein.

Beim Codeteil wt4uMigrations und wt4u handelt es sich um generierten Code und ist deshalb nicht zu beachten. Die Models nehmen einen sehr kleinen Platz in der Statistik ein, da keine Methoden enthalten sind sondern nur Properties.

Erstaunlicherweise fällt ein relativ grosser Teil des Kuchens auf das Testing.

## Abhängigkeiten auf fremde Packages (Microsoft)

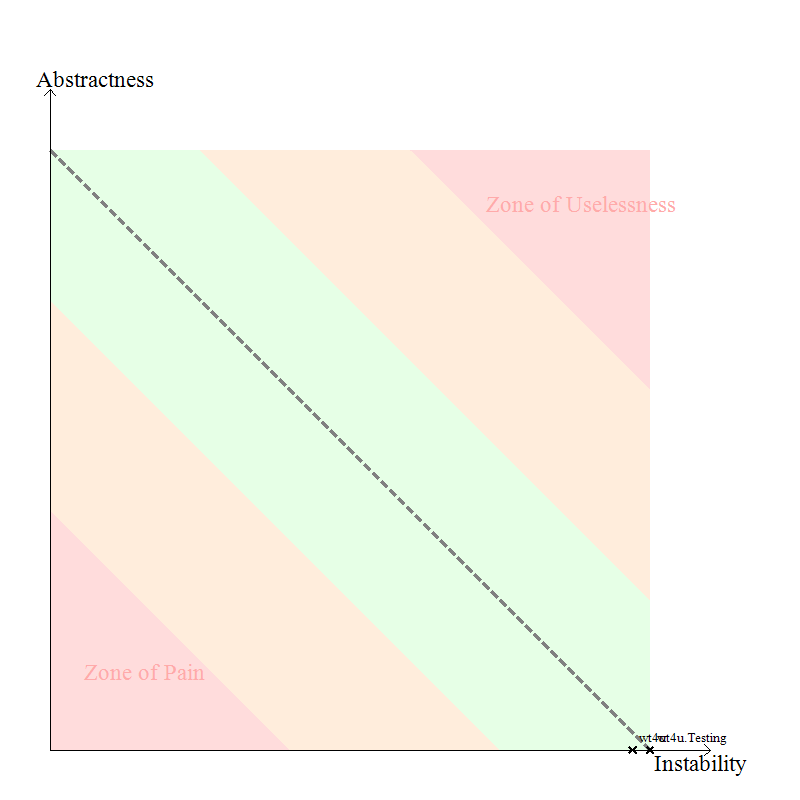


### Fazit / Schlussfolgerung

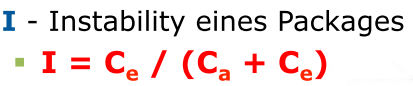
Mit unserer Applikation werden viele Packages von Microsoft API’s mitgeliefert. Dies liegt daran, dass die Webapplikation auf der Basis von ASP.NET MVC5 programmiert wurde.

Abgesehen von Transactions waren alle anderen Packages von Anfang an in die MVC5-Applikation integriert.

## Distance Diagramm

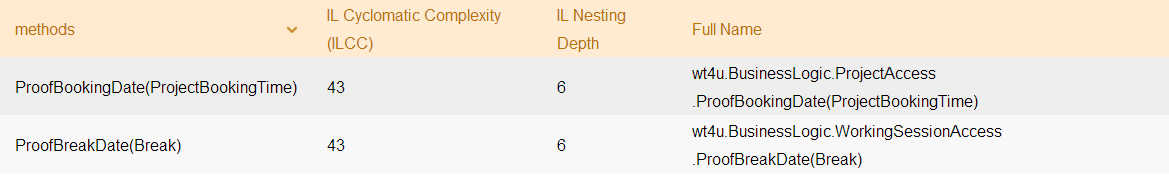


### Fazit / Schlussfolgerung

Da wir in unserer Applikation keine abstrakten Klassen verwenden, ist der Wert der Abstrakheit logischerweise auf 0. Die Instabilität ist auf praktisch auf dem Maximalwert. Dies liegt daran, dass wir innerhalb der Packages Controller und BusinessLogic viele Abhängigkeiten auf das Model haben und innerhalb des Controllerpackages viele Abhängigkeiten auf die Access-Klassen. Somit ergibt sich ein hohes efferent Coupling.

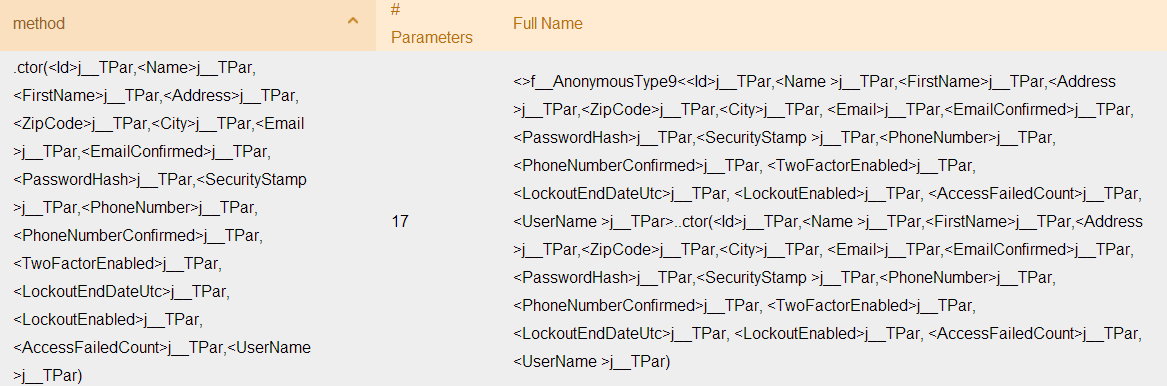
## Codeanalyse, Kritische Sektionen

### Method too complex – critcal



In diesen Methoden wird geprüft, ob ein geändertes Datum einer Projektbuchung oder Pause Komplikationen ergibt. Hierzu müssen leider alle Datensätze aus Breaks/Projectsbookings aus der Datenbank geholt werden und geprüft werden, ob das neue Datum keien Überschneidungen mit vorhandenen Daten ergibt. Dies ist leider nötig um sicher zustellen, keine falschen Zeiten gespeichert zu haben.

### Methods with too many parameters – critical



Eine Methode zur Erstellung von Usern braucht 17 Parameter. Dies ist aber vom vorgegeben User-System ASP.NET so implementiert.