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

1. 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.

1.1 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

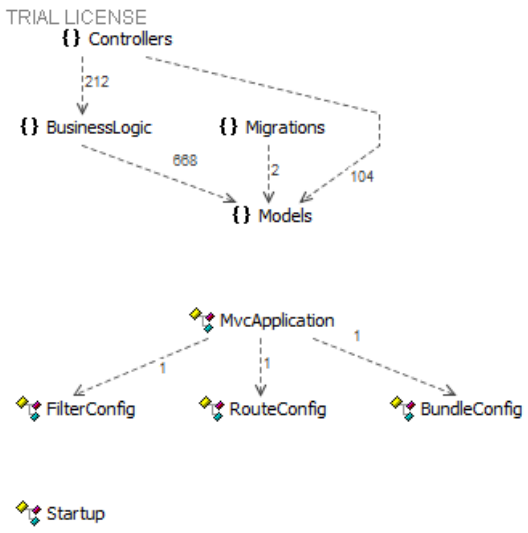
1.2 Kennzahlen von Ndepend

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

2. Analyse mit Structure101

2.1 Abhängigkeiten aller Packages

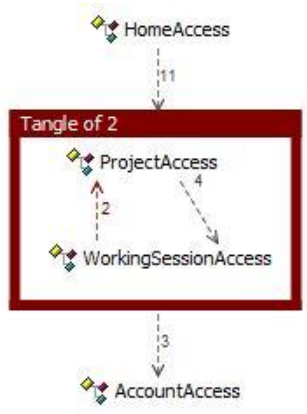
	Controllers	BusinessLogic	Migrations	MvcApplication	BundleConfig	FilterConfig	Models	RouteConfig	Startup
Controllers									
BusinessLogic	212								
Migrations									
MvcApplication									
BundleConfig				1					
FilterConfig				1					
Models	104	668	2						
RouteConfig				1					
Startup									



2.1.1 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.

2.2 Abhängigkeiten des Packages BusinessAccess

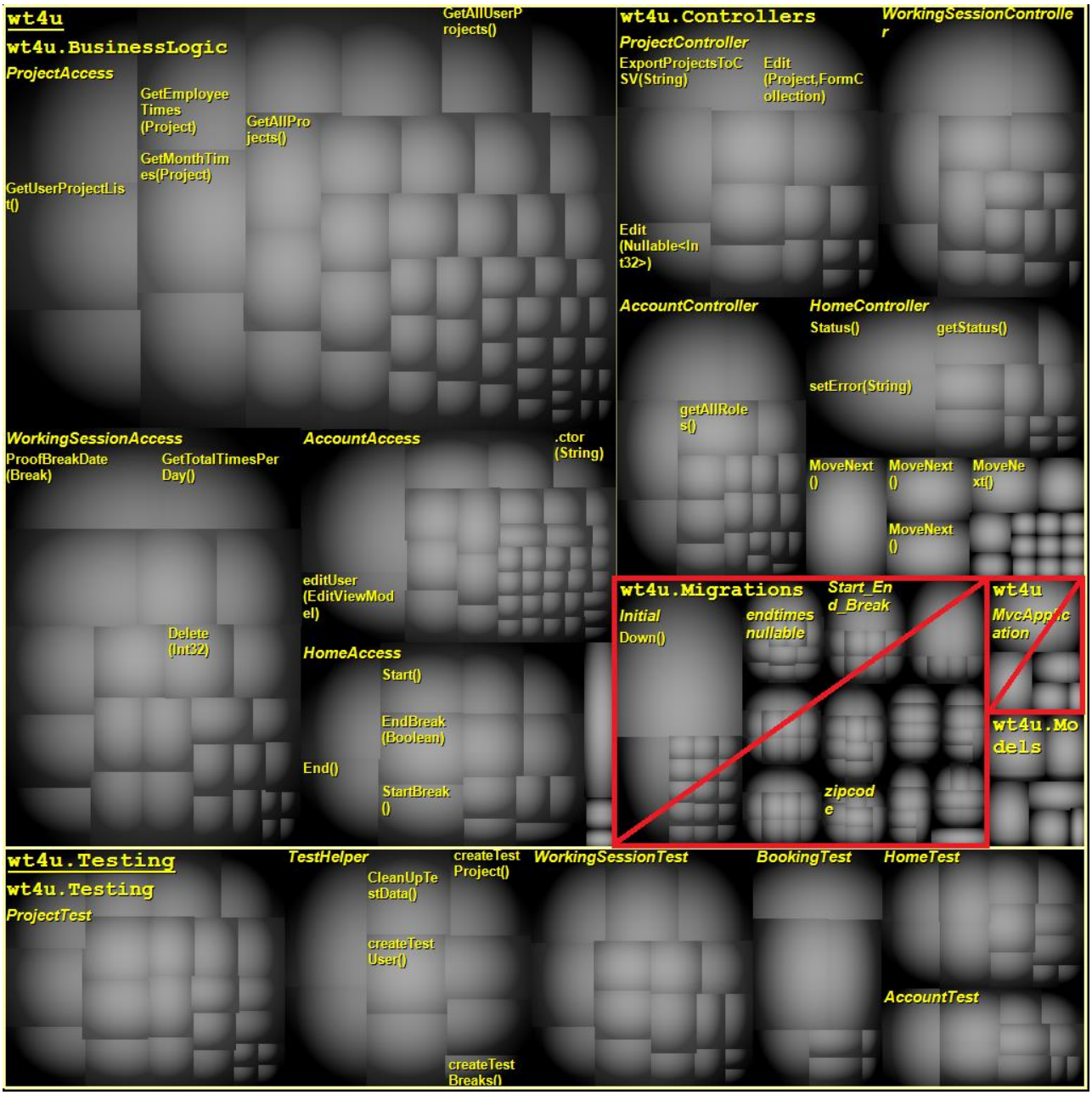


2.2.1 Fazit / Schlussfolgerung

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

3. Analyse mit Ndepend

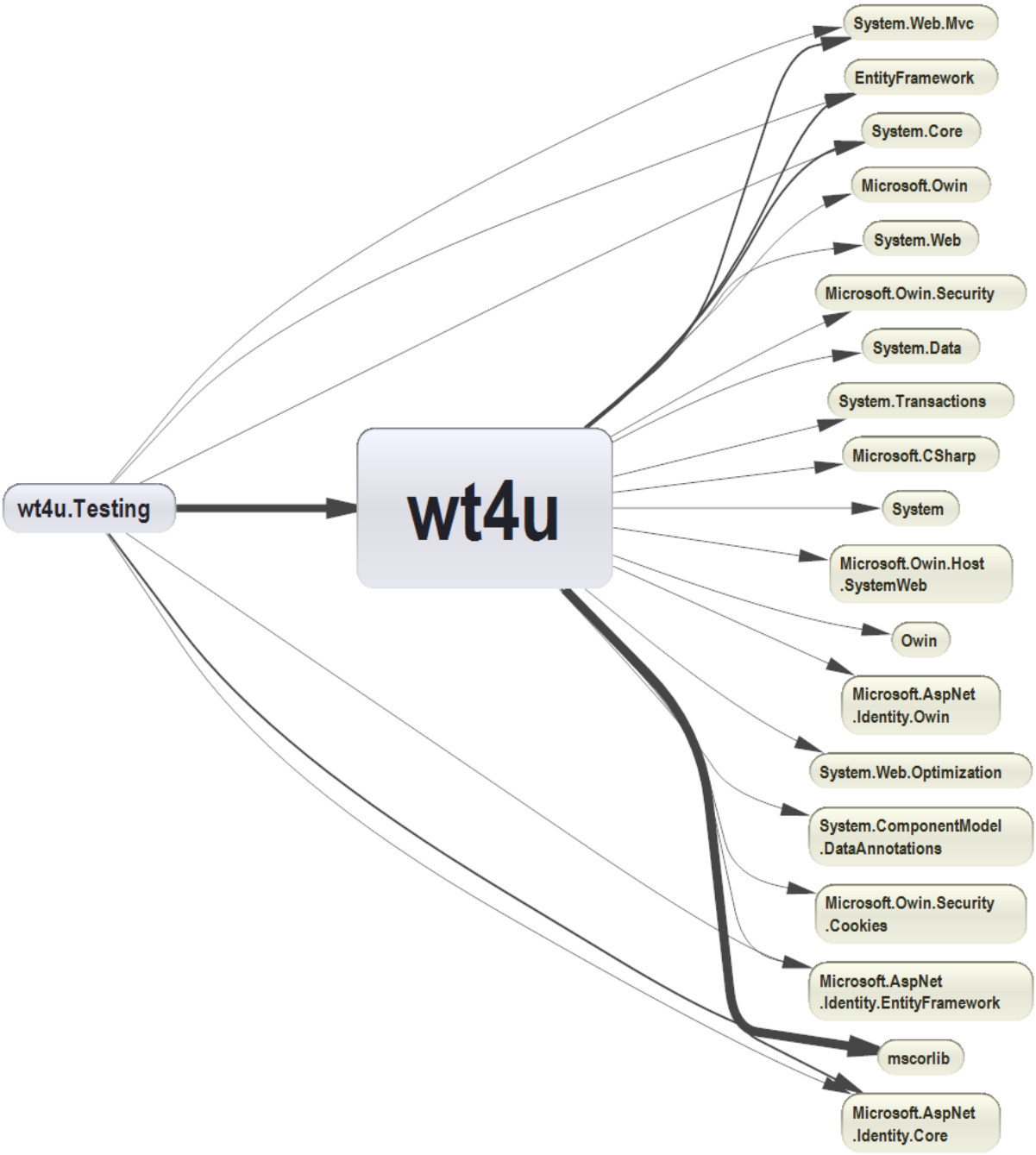
3.1 Grössen der Klassen/Methoden



3.1.1 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.

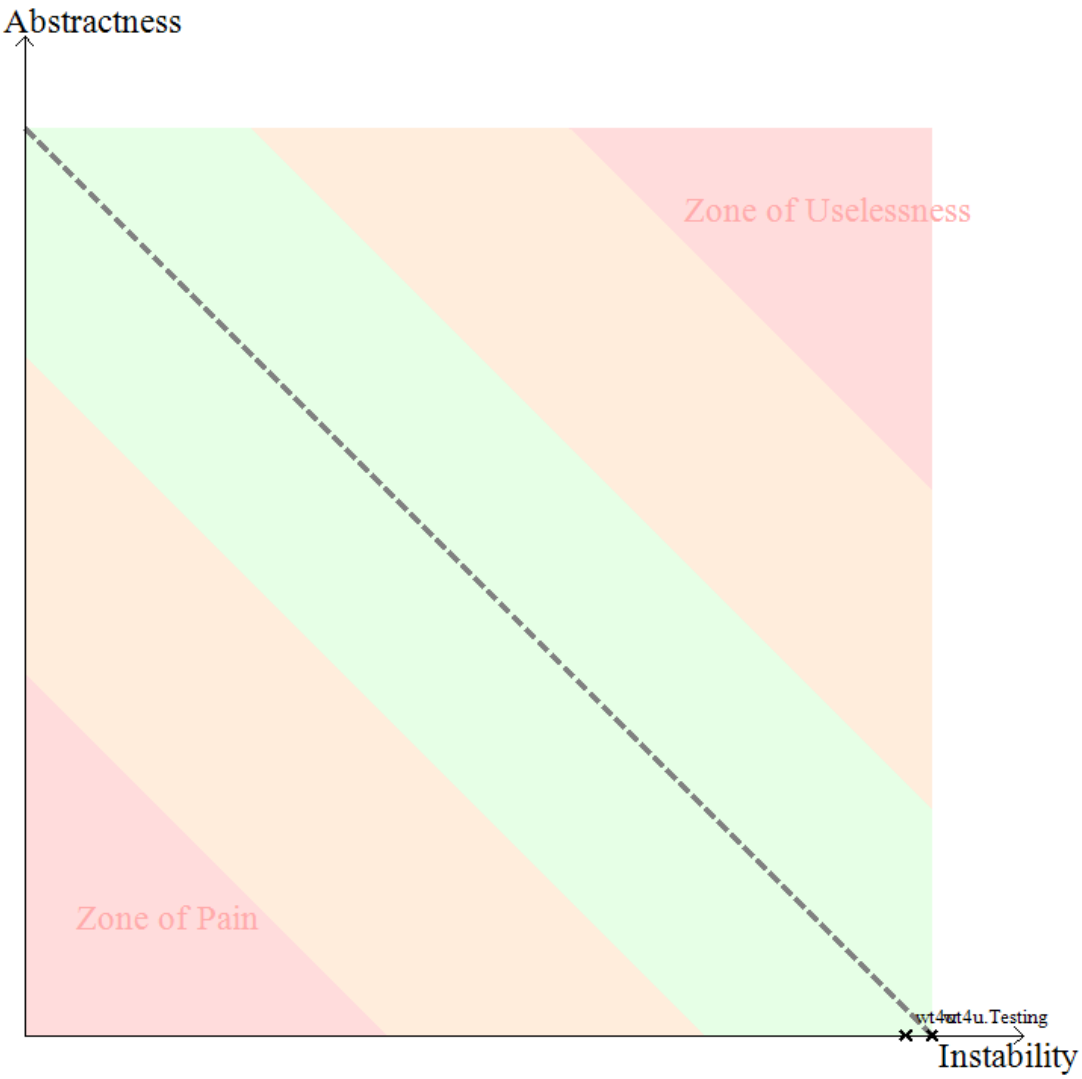
3.2 Abhängigkeiten auf fremde Packages (Microsoft)



3.2.1 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.

3.3 Distance Diagramm



3.3.1 Fazit / Schlussfolgerung

A - Abstractness

- **A = Anzahl abstrakter Klassen / Anzahl Klassen eines Packages**

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.

I - Instability eines Packages

- **$I = C_e / (C_a + C_e)$**

3.4 Codeanalyse, Kritische Sektionen

3.4.1 Method too complex – critcal

methods	IL Cyclomatic Complexity (ILCC)	IL Nesting Depth	Full Name
ProofBookingDate(ProjectBookingTime)	43	6	wt4u.BusinessLogic.ProjectAccess .ProofBookingDate(ProjectBookingTime)
ProofBreakDate(Break)	43	6	wt4u.BusinessLogic.WorkingSessionAccess .ProofBreakDate(Break)

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.

3.4.2 Methods with too many parameters – critical

method	# Parameters	Full Name
.ctor(<Id>j__TPar,<Name>j__TPar, <FirstName>j__TPar,<Address>j__TPar, <ZipCode>j__TPar,<City>j__TPar,<Email >j__TPar,<EmailConfirmed>j__TPar, <PasswordHash>j__TPar,<SecurityStamp >j__TPar,<PhoneNumber>j__TPar, <PhoneNumberConfirmed>j__TPar, <TwoFactorEnabled>j__TPar, <LockoutEndDateUtc>j__TPar, <LockoutEnabled>j__TPar, <AccessFailedCount>j__TPar,<UserName >j__TPar)	17	<>f__AnonymousType9<<Id>j__TPar,<Name >j__TPar,<FirstName>j__TPar,<Address >j__TPar,<ZipCode>j__TPar,<City>j__TPar, <Email>j__TPar,<EmailConfirmed>j__TPar, <PasswordHash>j__TPar,<SecurityStamp >j__TPar,<PhoneNumber>j__TPar, <PhoneNumberConfirmed>j__TPar, <TwoFactorEnabled>j__TPar, <LockoutEndDateUtc>j__TPar, <LockoutEnabled>j__TPar, <AccessFailedCount>j__TPar, <UserName >j__TPar>..ctor(<Id>j__TPar,<Name >j__TPar,<FirstName>j__TPar,<Address >j__TPar,<ZipCode>j__TPar,<City>j__TPar, <Email>j__TPar,<EmailConfirmed>j__TPar, <PasswordHash>j__TPar,<SecurityStamp >j__TPar,<PhoneNumber>j__TPar, <PhoneNumberConfirmed>j__TPar, <TwoFactorEnabled>j__TPar, <LockoutEndDateUtc>j__TPar, <LockoutEnabled>j__TPar, <AccessFailedCount>j__TPar, <UserName >j__TPar)

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