Projektpräsentation „Warenversandhaus“

Modellierung von Anwendungssystemen | SoSe 2008
Mathias Slawik | Jan Omar | Robert Galanty | Serkan Yilmaz
21.07.2008 | FB4 WI | FHTW Berlin

Gliederung

- Verwendete Technologien
- Vorgehen
- Ergebnisse
- Präsentation Prototyp

Verwendete Technologien (1)

- Modellierung
  - Anwendung: **ObjectiF 7.0**
  - Daten: **PowerDesigner 12**
- Implementierung
  - Programmiersprache **C#**
  - IDE **Visual Studio 2005**
  - Datenbank **MS SQL Server**
  - Persistenzschicht **Gentle.NET + MyGeneration**
  - Versionierungssystem **Subversion**

Verwendete Technologien (2)
Warum C# & .NET und nicht Java & J2EE?

- C# an der FHTW stark unterrepräsentiert, obwohl sehr weit verbreitet
  - Möglichkeit „über den Tellerrand“ hinaus zu blicken
  - Einseitigkeit (= nur Java) im Berufsleben nachteilig
- Sprache mächtiger, bequemer, griffiger, flexibler und besser an Windows angepasst als Java
- Einstieg sehr leicht
  - Dokumentation der Sprache und des .NET-Frameworks massiv umfangreicher, als bei Java & J2EE (übrigens auch komplett in Deutsch ☺)
  - Viel mehr „Werkzeugklassen“ als in Java
  - Weniger Entwicklungszeit und Code notwendig
  - Viele Sprachneuerungen erscheinen zuerst in C#

Verwendete Technologien (3)
Warum Visual Studio 2005?

- Übliche Funktionen einer IDE vorhanden
- Hervorragende Editoren für Code, XML, Icons, etc.
- Hervorragender GUI-Editor
  - Generierter GUI-Code musste nie verändert werden!
- „Veröffentlichen“-Funktion erstellt Windows-Installationsprogramme
- „Edit and Continue“ - Codeänderungen im laufenden Betrieb möglich
- Integrierter Datenbrowser für SQL Server
- MSDN – Library mit 2GB+ an Artikeln, Tutorials und Referenzen zu allen Microsoft-Technologien

Verwendete Technologien (4)
Persistenzschicht

- Gentle.NET
  - Eine von vielen Persistenzschichten für C#.NET – Anwendungen
  - Einzige von ObjectiF unterstützte Persistenzschicht
  - Vemittelt zwischen Objekten und Datenbanken
    - Kunde k = new Kunde(); - Erstellt neues Kundenobjekt
    - k.Special = true; - Macht Kunden zum Stammkunden
    - k.Persist(); - Speichert Kunden in Datenbank
    - k.Remove(); - Löscht Kunden aus Datenbank
    - Kunde k = Kunde.Retrieve(1); - Holt Kunden mit der ID = 1
    - IList kunden = Kunde.RetrieveAll(); Holt alle Kunden

Verwendete Technologien (5)
Persistenzschicht

- Gentle.NET
  - Breite Unterstützung diverser Datenbanken (MS SQL Server, MySQL, PostgreSQL, etc.)
  - Leider seit mehreren Jahren nicht mehr weiterentwickelt, sehr schlechte und spärliche Dokumentatiton
  - Seit C# 3.0 übernimmt Microsoft LinQ viele Funktionen
- MyGeneration
  - OpenSource Codegenerierungstool für diverse C# Persistenzschichten

Vorgehen (1)
Design

- PowerDesigner
  - Datenmodell erstellen
  - Datenbank generieren
- MyGeneration
  - Gentle.NET – Klassen aus Datenbank generieren
- ObjectiF
  - Generierte Klassen in ObjectiF-System importieren
  - Klassendiagramm erstellen
  - C# - Projekt generieren

Vorgehen (2)
Implementierung

- Subversion
  - Generiertes C# - Projekt in Repository importieren
- Visual Studio
  - Anwendung programmieren
  - Codeverwaltung über Subversion

Vorgehen (3)
Nebenläufige Tätigkeiten

- Pflege
  - Stundenlisten
  - Projektdokumentation
- Test des Prototyps
- Meetings / Mails zur Abstimmung des weiteren Vorgehens

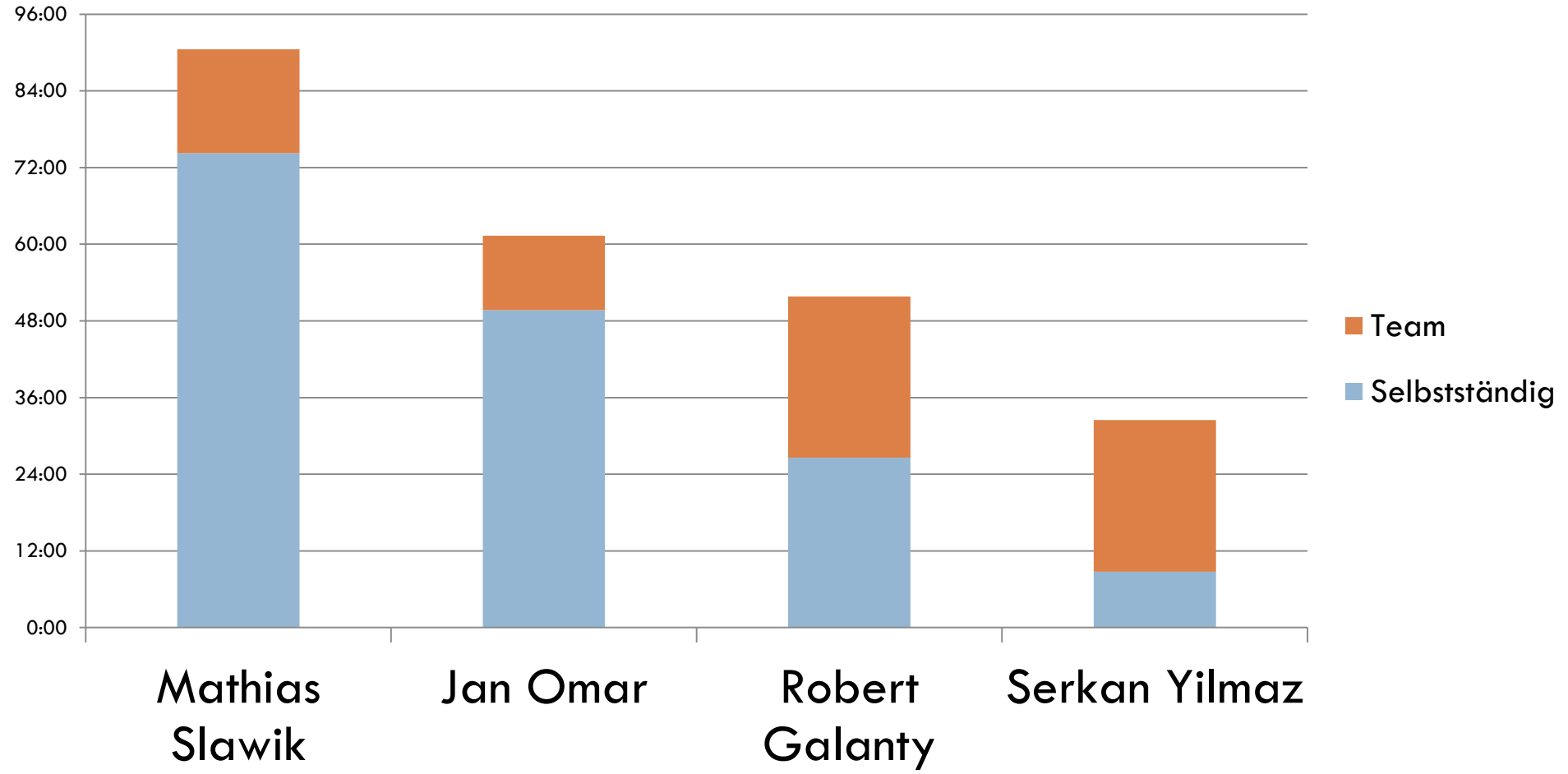
Rollenverteilung

- Projektleitung
  - Mathias Slawik
- Software Design
  - alle
- Implementierung
  - Programmierung: Mathias Slawik, Jan Omar
  - Qualitätssicherung & Tests: Robert Galanty, Serkan Yilmaz
- Dokumentation
  - Vorlagen & Grundstruktur: Robert Galanty, Serkan Yilmaz, Jan Omar
  - Fertigstellung & Qualitätssicherung: Mathias Slawik
- Testszenarien
  - Erarbeitung: Robert Galanty, Serkan Yilmaz, Jan Omar
  - Fertigstellung: Mathias Slawik

Zusammenfassung Ergebnisse

- Aufwand
  - Insges. 236 Stunden = ca. 30 MT
  - Ungleiche Verteilung auf Teammitglieder
    - Ursachen
      - Unt. Kompetenzfelder (Programmierung, BWL)
      - Unt. Leistungsbereitschaft
      - Unt. Qualität der Leistungen
- Ergebnis
  - Ca. 60 Seiten Dokumentation, Pläne, Zeichnungen, etc.
  - Prototyp
    - 35 Klassen
    - 7940 Codezeilen (außer Kommentare und Leerzeilen)
    - Davon ca. 7500 Zeilen generiert durch MyGeneration und VS GUI-Editor

Verteilung Aufwand



Vergleich mit Plan (1)

Vorgangsname	MS P	MS I	MS D	JO P	JO I	JO D	RG P	RG I	RG D	SY P	SY I	SY D	S P	S I	S D	Begründung
Pflege Dokumentation & Projektplan	60	21	- 39	50	3	- 47	30	19	- 11	30	9	- 21	170	52	- 118	**Keine begleitende Pflege, Weniger Gesamtaufwand**
Besprechung der zukünftigen Aufgabenverteilung	2	10	+ 8	2	10	+ 8	2	10	+ 8	2	10	+ 8	8	40	+ 32	**Mehr Abstimmungsaufwand, als gedacht**
Projektplan erstellen	1	2	+ 1	1		- 1	1	2	+ 1	1	2	+ 1	4	6	+ 2	**Einarbeitung in Project, genauere Planung**
Klassendiagramme erstellen	5		- 5	5	2	- 3	2		- 2	2		- 2	14	2	- 12	**Weniger Aufwand als gedacht**
Klassenbeschreibung anlegen	3	4	+ 1	3		- 3	3		- 3	3		- 3	12	4	- 8	**Weniger Aufwand als gedacht**
Datenbank Design erstellen	3	5	+ 2	3	3			3	+ 3				6	11	+ 5	**Komplizierteres Datenmodell als geplant**
Style Guidlines Entwickeln	1	1		1		- 1	3	5	+ 2	3	3		8	9	+ 1	**Keine Guidelines, sondern Entwürfe der Dialoge erstellt**
Testszenarien erstellen	2		- 2	2		- 2	5	2	- 3	5		- 5	14	2	- 12	**Keine genaue Zuordnung, wahrscheinlich mehr ...**
Programmierung der Software & Tests durchführen	70	49	- 21	100	40	- 60	10	10		10	9	- 1	190	108	- 82	**Geringerer Funktionsumfang d. Prototypen, Einstieg einfacher als gedacht**
Erstellen der Präsentation	4	4											4	4		
Projektleitung		4	+ 4											4	+ 4	**Fehlt im Plan**
Sonstige					2						1	+ 1		3	+ 1	**Fehlt im Plan**
Ergebnis	**151**	**100**	**-51**	**167**	**60**	**-109**	**56**	**51**	**-5**	**56**	**34**	**-22**	**430**	**245**	**-187**	

Projektmitglieder
MS - Mathias Slawik
RG - Robert Galanty
JO - Jan Omar
SY - Serkan Yilmaz

Schlüssel
P - Plan
I - Ist
D - Differenz
S - Summenspalte

Anmerkungen

Alle Zahlen sind nicht vollständig korrekt, da bei der Zusammenfassung der Stundenlisten (Siehe Stundenlisten.xslx) nicht die gleichen Kategorien verwendet wurden, wie bei dieser Liste. Daher können nicht alle Stunden genau zugeordnet werden. Für eine genaue Auflistung und Einteilung der Stunden bitte die Datei Stundenlisten.xslx verwenden. Alle Werte sind gerundet - Rundungsdifferenzen beachten!

Vergleich mit Plan (2)

- Zusammenfassung
  - Weniger Stunden notwendig gewesen, als geplant
- Größte Abweichungen
  - Mehr Stunden bei Abstimmung und Datenmodell notwendig
  - (Viel) weniger bei Pflege Dokumentation und Implementierung
- Ursachen
  - Geringerer Umfang als geplant
  - Leichterer Einstieg in C# als gedacht
  - Kaum begleitende Pflege von Projektplan notwendig gewesen

Livepräsentation Prototyp

- Anmeldung
- Artikel anlegen
- Lieferant anlegen
- Artikel bestellen

- Starten

Ende

Vielen Dank für Eure Aufmerksamkeit