

nette **Bet**



Bericht

des Online Betting Systems *BetGetter und Viveo*

Version: 1.0
Kontext: Laborprojekt "Online Betting System", Distributed Computing Group, ETH Zürich, 2003/04
Autoren: Gregor Bättig, Stefan Bollmann, Andreas Diener, Fabian Nart und Christian Wassmer
Betreuer: Ruedi Arnold
Kontakt: rarnold@inf.ethz.ch
Datum: 26. Februar 2004

Zusammenfassung

Wir haben als 5-köpfiges Team im Laufe eines Semesters ein Online Betting System als Webapplikation basierend auf *Clippee* erstellt. In den folgenden Berichten werden die einzelnen Arbeiten, dabei aufgetretene Probleme und deren Lösungen sowie persönliche Erfahrungen von den Teammitgliedern aufgezeigt.

Aufgabenstellung

Es war unsere Aufgabe, in einer Gruppenarbeit ein Online Betting System in Form eines Peer-to-Peer Systems zu bauen, welches zum Beispiel für die Fussball Europameisterschaft 2004 in Portugal verwendet werden kann. Als oberste Maxime galt es am Ende des Semesters ein lauffähiges System vorweisen zu können. Weitere Anforderungen waren:

- verteilte Lösung
- Ausfallsicherheit:
 - Benutzer sollte beim Ausfall eines oder mehrerer Server weiterhin wetten können
 - kein Datenverlust bei Ausfall aller Server (Datensicherung)
- Tipper erhält verschiedene Auswertungen und Statistiken über das Spiel
- Tippgemeinschaften werden unterstützt

Umsetzung

Als Frontend wurde aus Benutzerfreundlichkeit eine Webapplikation gewählt, wodurch ein Download wie bei einer Stand-Alone Variante entfällt. Mittels Java Server Pages wird mit dem Webserver (Tomcat) über eine Schnittstelle auf die darunterliegende Spiellogik zugegriffen. Die Spiellogik nutzt die im Forschungsprototypen *Clippee* vorhandene Verteilung und Updates der Daten auf alle Server aus.

Aufgrund der Architektur teilten wir uns in zwei Sub-Teams auf, wovon eines für die Spiellogik und das andere für das Frontend zuständig war. Um trotz dieser Aufteilung eine gute Kommunikation zu ermöglichen, wurden neben regelmässigen Sitzungen und Emails auch ICQ genutzt, sowie alle Schnittstellendefinitionen und weitere Dokumente auf dem BSCW (Basic Support for Cooperative Work) abgelegt. In einer Testwoche wurde das System erfolgreich getestet.

Die Ausgangslage

Nach einer außergewöhnlichen Erfahrung betreffend Teamarbeit in der Vorlesung Global Information Systems habe ich mich kurzfristig entschieden, vom Algorithms track auf den Systems track des Masters in verteilten Systemen umzusteigen und in einem weiteren, grösseren Teamprojekt, einem so genannten Labor, mitzumachen.

Die Aufgabenstellung liess uns viele Freiheiten. So mussten wir nicht zwingend auf bestehendem Code aufbauen und konnten sogar die Technologie und einen Grossteil der Funktionalität selbst bestimmen. Diese Freiheiten waren für mich sehr motivierend, erlaubten sie mir doch kreative Gedanken und führten dazu, dass wir vieles, was häufig gegeben ist und einfach akzeptiert wird, noch einmal überdachten und erst nach einer bewussten und begründeten Entscheidung einsetzten. Natürlich brachte dieser Start von Null auf auch einige Unannehmlichkeiten mit sich. So dauerte es insgesamt drei Wochen bis wir unsere Infrastruktur, die aus drei Laptops bestand, und die Schlüssel zum Laborraum organisiert hatten.

Die oberste Direktive, am Ende des Semesters ein funktionierendes Produkt präsentieren zu können, verlieh unserer Arbeit eine Praxisnähe und war bei vielen Entscheidungen mittragend.

Sport, insbesondere Fussball zählt nicht gerade zu meinen Hauptinteressen. Dennoch reizte mich die Aufgabe, ein System verteilt, ausfallsicher und lastbalanciert zu erstellen, sehr. Das Laborprojekt startete mit vier für mich unbekanntem Gesichtern und einer Aufgabenstellung, die eine Unmenge von Möglichkeiten offen liess.

Der Entwicklungsprozess

Kick-off

In den ersten Semesterwochen hatten wir uns teamintern keine speziellen Rollen zugeteilt. Es ging darum, die Aufgabenstellung zu konkretisieren und genau zu definieren, wie wir was machen wollten. Um uns auf die Funktionalität unseres Produktes zu einigen und diese geeignet darzustellen, entschieden wir uns dazu, zuerst Use Cases und dann ein Hypertext Model von unserem Online Betting System zu erstellen. Da ich diesbezüglich schon einige Erfahrungen mitbrachte,

erstellte ich eine erste Version der Use Cases und entwarf dann zusammen mit Stefan die Hypertext Darstellung des Backends.

Nachdem wir spezifiziert und formuliert hatten, was wir genau machen wollten, gingen wir der Frage nach, wie dies nun zu realisieren sei. Die Anforderungen an unser Produkt waren gegeben: Ausfallsicherheit und Lastbalancierung durch Verteiltheit und, als oberste Direktive, ein funktionierendes Produkt bis Ende Semester.

Kritische Entscheidungen

Die erste wichtige Entscheidung war diejenige, welche wir zwischen Web-Applikation vs. Stand-alone treffen mussten. Wir entschieden uns hauptsächlich aus Gründen der Benutzbarkeit für die Web-Applikation. Ein zwingender Download einer Software hätte wohl viele potentielle Benutzer abgeschreckt.

Für die Verteilung der Daten auf allen Rechnern hätten wir entweder den Peer-to-Peer Forschungsprototypen Clippee von Ruedi und Keno oder eine bestehende verteilte Datenbank einsetzen können. Wir entschieden uns für Clippee.

Ein weiterer Aspekt, der in Betracht gezogen werden musste, war, dass alle funktionierenden Peers und nur die funktionierenden Peers über das WWW erreicht wurden und dies zusätzlich lastverteilt geschah. Dafür konnten wir uns einerseits Dynamisches DNS kombiniert mit Round Robin und andererseits Windows Server 2003 als mögliche Lösungen vorstellen. Es war meine Aufgabe herauszufinden, was Windows Server 2003 diesbezüglich anbot und wie dies realisiert war. Diese Recherchenarbeit gestaltete sich nicht ganz einfach, da die Internetseiten von Microsoft voll von Werbetexten, Pressemitteilungen und Bildern sind. Ich konnte aber das Buch "Microsoft Windows 2000 Server, Administrator's Companion" ausleihen und fand erste Anhaltspunkte. Nach weiterem Suchen fand ich ein Dokument mit dem Titel "Network Load Balancing Technical Overview" welches die gesuchte Information enthielt. Wir entschieden uns aber dennoch gegen die kommerzielle Lösung. Unter den Gründen befand sich einmal mehr das Argument der Zeit. Es hätte sicherlich einige Wochen gedauert, bis wir auf den Laptops eine saubere Serverinstallation mit funktionierendem Load Balancing zu Stande gebracht hätten. Dazu hätte der Hauptteil unserer Arbeit aus Konfiguration und Installation bestanden und 3 Serverlizenzen von Microsoft hätten uns etwa 500 sFr. gekostet.

Frontend und Backend Team

Nachdem sowohl unser Ziel, als auch der Weg dorthin bekannt waren, ging es an die eigentliche Realisierung. Um unsere Energie gebündelt einsetzen zu können und damit nicht jeder in allen Gebieten den Überblick behalten musste, schlug ich eine Aufteilung des Teams vor. Wir entschieden uns, ein Front- und ein Backend Team zu bilden. Das Frontend Team war fortan für alle Graphik- und Darstellungsaspekte verantwortlich, das Backend Team für die Datenhaltung, den Austausch unter den verschiedenen Peers und die gesamte Funktionalität. Ich gehörte dem Backend Team an.

Am ersten Backend-Team Meeting entwarfen wir eine grobe Übersicht über die geplanten Java Klassen und die in ihnen enthaltene Funktionalität. Während Gregor und Christian mit dem Programmieren loslegten, erstellte ich ein UML Diagramm unserer geplanten Arbeit, welches auch im weiteren Verlauf des Laborprojektes regelmässig auf den neuesten Stand gebracht werden musste. Meine Programmieraufgabe war unter anderem das Erstellen der

Filterfunktionalität für die im System enthaltenen Matches. Dazu verwendete ich das Kompositum Entwurfsmuster, welches fortan ein freies Zusammenstellen der Filterfunktionalitäten ermöglichte.

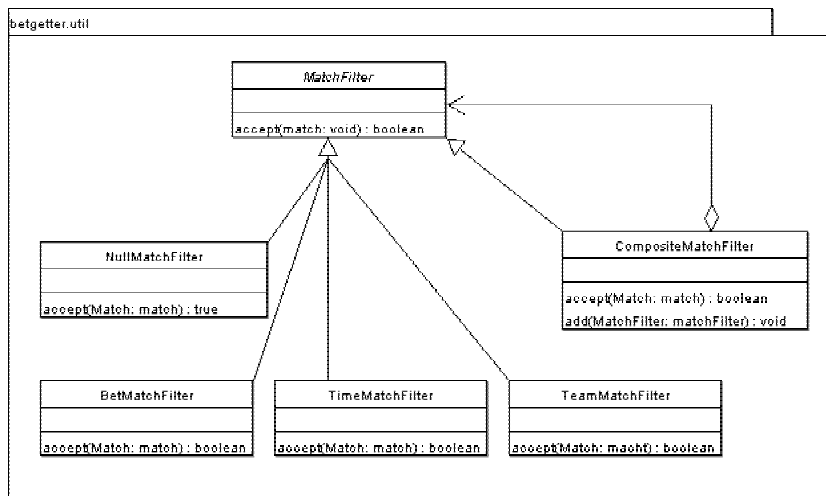


Abbildung 1 Filterfunktionalität in UML Darstellung

Weitere Methoden, die ich implementierte waren die folgenden:

- `List getMatches(int, int, int, String)`
- `int createAccount(String, String, String)`
- `int login(String, String)`
- `int placeBet(String, int, int, int)`
- `int create Match(String, String, Date, String, String)`

Wir benutzten verschiedene Wege um mit dem Frontend Team zu kommunizieren und unsere Arbeit abzustimmen. Da waren einmal unsere Interface Definitionen; das Frontend Team erstellte ein Dokument mit den benötigten Konstanten und Methodensignaturen, welche implementiert werden mussten. Über das CVS konnten die Design- und Darstellungsspezialisten natürlich auch jederzeit unseren Code und unsere Kommentare einsehen. Zusätzlich trafen wir uns weiterhin zwei Mal wöchentlich um anstehende Fragen und Probleme zu diskutieren.

Protokolle

Um Probleme und Entscheidungen nicht mehrmals ausdiskutieren zu müssen, wurden alle Beschlüsse, die in den über 50 Sitzungsstunden getroffen wurden, in Protokollen festgehalten. Ruedi, Gregor und ich mussten während des Semesters krankheitshalber einigen Sitzungen fernbleiben. Gerade in diesen Momenten waren die Protokolle sehr nützlich, denn sie ermöglichten einen schnellen Wiedereinstieg in das Projekt. Allerdings war das Schreiben des Protokolls nicht die beliebteste Arbeit!

Testen

Es war uns wichtig, dass wir in der Testwoche nicht mit einem fehlerhaften Produkt Peinlichkeiten erleben mussten. Deshalb wurde auf verschiedene Art und Weise getestet: Bevor neuer Code ins CVS eingekcheckt wurde, testete jeder Entwickler lokal auf seiner Maschine den neu erstellten Code. Da ein Debuggen

in verteilten Systemen eine beinahe unmögliche Angelegenheit ist, wurde im Backend wo immer möglich mit automatischen Unit Tests gearbeitet.

Nach einigen abgeschlossenen Entwicklungsphasen führten wir auch grössere Tests durch, in welchen mehrere Leute das System benutzten und kleine Wettrunden durchspielten.

Ich war überrascht, wie viel Zeit das Testen und Verbessern des bestehenden Codes benötigte.

Zeitplan

Die oberste Direktive lautete wie bereits erwähnt "ein fertiges Produkt bis Ende Semester". Um dieses Ziel einzuhalten, erstellten wir zu Beginn der Implementierungsphase einen detaillierten Zeitplan, in welchem wir alle Features mit entsprechenden Terminen festhielten. Wir planten auch eine Woche für Testphasen und Bugfixes ein. Es bedurfte einiger schlafloser Nächte, doch unser System war zu Beginn der geplanten Testwoche komplett funktionstüchtig fertig gestellt, worauf wohl nicht nur ich sehr stolz war!

Testwoche

Die Testwoche verlief relativ ruhig und ohne grössere Probleme. Als Kontaktperson gegen aussen erhielt ich vor allem Mails mit dem Hinweis, dass auf Macintosh Computern Darstellungsprobleme bestanden. Ein ETH Assistent machte uns darauf aufmerksam, dass bei einer kommerziellen Benutzung zudem rechtliche Aspekte abzuklären seien. Und dann gab es einige Pechvögel, die genau während eines System Updates auf viveo.ethz.ch zugreifen wollten und einen Tomcat Fehler erhielten. Im Allgemeinen sah ich die Testwoche mit weit über hundert registrierten Benutzern, womit wir uns übrigens ein gediegenes Abendessen bei Ruedi verdient hatten, als grossen Erfolg!

Dynamisches DNS

Ursprünglich wollten wir für unser System Dynamisches DNS einsetzen. Die Informatikdienste der ETH boten diesen Service aber leider nicht direkt an; wir hätten unseren eigenen Nameserver betreiben müssen. Ich organisierte eine Woche vor unserem Testlauf Bücher über DNS, installierte auf einem unserer LAB-Computer den Nameserverdienst Bind9 wurde dann aber leider krank. Somit blieb nicht genügend Zeit, um diese Aufgabe abzuschliessen.

Teamwork

Noch vor Semesterbeginn erhielten wir von unserem Betreuer Ruedi die einfache Aufgabe, einen Termin zu bestimmen, an dem alle Laborteilnehmer anwesend sein konnten. Es bedurfte unglaublicher 17 Mails, bis wir uns einigen konnten! Die Terminfindung war aber aus meiner Sicht auch der einzige negative Aspekt unserer Teamarbeit. Die Zusammenarbeit und die Akzeptanz im Team waren sehr gut. Ich empfand das Klima als konstruktiv und motivierend und konnte vieles lernen. Wir haben die Aufgaben stets so verteilt, dass jeder die Aufgaben machen konnte, welche er gerne machte und in welchen er sich sicher fühlte. So ergänzten wir uns ideal und erbrachten meiner Ansicht nach eine ansehnliche Leistung.

Für die Zusammenarbeit und Dokumentverwaltung setzten wir die Groupware BSCW (Basic Support for Cooperative Work) ein. Für mich war die Benützung eines derartigen gemeinsamen virtuellen Arbeitsbereiches eine neue und sehr interessante Erfahrung. Zusätzlich installierte jedes Teammitglied die Instant Messenger Applikation ICQ, welche uns einen ortsunabhängigen Wissensaustausch ermöglichte und von dem wir auch regen Gebrauch machten. Eine unserer Sitzungen fand übrigens im "englishquiz" Chatroom von Bluewin statt, da zu diesem Zeitpunkt ein physisches Treffen nicht möglich war!

Während des ganzen Projektes gab es in unserem Team keinen Führer oder Koordinator. Für mich war es zu Beginn unvorstellbar, dass eine solche Arbeit ohne zentrale Organisation funktionieren würde. Es tauchten auch kleine Probleme auf: an Sitzungen wurden offene Aufgaben zusammengetragen und erfasst aber niemandem zugeteilt. Um dies zu verhindern, schlug ich vor, dass der Protokollführer verantwortlich war, für alle anstehenden Aufgaben ein Teammitglied zu finden, was dann gut funktionierte.

Abschliessend möchte ich erwähnen, dass wir eine Menge Spass hatten und dass all die unbekanntenen Gesichter zu guten Kollegen wurden! Danke viveo Team!

Fazit

Das Laborprojekt war für mich persönlich ein grosser Erfolg und übertraf meine Erwartungen bei Weitem! Ich habe gelernt, dass Projektmanagement auch ohne Leader möglich ist, wie man Java Code sauber dokumentiert, was Unit Testing ist, wie Modelle bei der Entwicklung eines grösseren Projektes helfen können, wie wertvoll Teamarbeit sein kann und hatte viele lustige, interessante und aufhellende Momente.

Erledigte Arbeiten

Business-Logic

Wir teilten unsere Arbeit in einen Frontend- und einen Backend-Teil auf. Als Mitglied des Backend-Teams war ich für Funktionalitäten der Business-Logic sowie für bestimmte Tools verantwortlich. Im folgenden Abschnitt werden einige Funktionalitäten und Tools erläutert, welche von mir implementiert wurden.

Bootstrapping Init:

Beim Start von einzelnen Peers stellt sich die Frage, ob der Peer der erste ist oder ob schon weitere Peers im System existieren. Um dies herauszufinden, sind im Config-File (siehe weiter unten) die IPs potentieller Peers aufgelistet. Ein beliebiger neuer Peer versucht sich nun der Reihe nach mit jedem Peer in dieser Liste zu verbinden. Gelingt keine Verbindung, so startet er die notwendigen Initialisierungsroutinen für einen ersten Peer. Ansonsten startet er nur diejenigen Initialisierungs-Routinen, die alle Peers ausführen müssen – egal ob einer der erste ist oder nicht.

Datenmodell

Wir teilten die Implementierung unseres Wettsystems in drei Phasen auf. In Phase 1 und 2 stand der einzelne Benutzer im Vordergrund. Nach Abschluss der Phase 2 sollte er Wetten auf beliebige Spiele abschliessen können. Dazu implementierte ich folgende Klassen, die darin enthaltene Funktionalität sowie die Clippee-Serialisierungsmethoden:

- Bet
- Match
- MatchBet
- MatchBet1X2
- Result
- Team

Schnittstellen-Funktionalität

Der Hauptteil der Spiel-Funktionalität befindet sich im `ApplicationBean`. Meine Hauptaufgabe – neben den Quoten und Cliques, welche weiter unten beschrieben werden - war es, Methoden sowohl für User- als auch für

Cliquenranglisten sowie sonstige sortierte Listen zu implementieren. Eine Auswahl davon zeigt folgende Aufzählung:

- `List getUserRanking()`
- `List getUserRanking(int count)`
- `int getUserRank(String userID)`
- `List getUsers(int order, String firstLetter)`
- `List getTeams()`

Cliquen

In Phase 3 galt es unter anderem die gesamte Funktionalität für die Cliquen auf die Beine zu stellen. Dabei waren auch Änderungen im bisherigen Datenmodell nötig. Zum Beispiel mussten `User` wissen, in welchen Cliquen sie dabei sind und in welchen sie sich beworben haben. Das verlangte nach Änderungen in den Serialisierungsmethoden der `User`-Klasse. Ansonsten gab es dabei keine grösseren Probleme.

Quoten

Die von uns für alle Spiele und alle deren Ergebnisse festgelegte Fixquote von 3.0 stiess auf grosse Kritik. Deshalb sahen wir uns gezwungen und wurden auch gedrängt, eine Lösung für dieses Problem zu finden. Die Variante, die sich durchsetzte, war, dass man zu jedem Spiel ein Zahlentriplett als Quote für die einzelnen Ergebnisse (Sieg Heimteam, Unentschieden, Sieg Gastteam) des Spieles angeben kann.

Da diese Forderung erst knapp zwei Wochen vor der Schlusspräsentation bekannt wurde, blieb uns nur noch etwa eine Woche, um dies zu implementieren und zu testen. Die Schlusspräsentation und diverse Tests mussten ja schliesslich auch noch vorbereitet werden.

Tools

Im Folgenden werden einige Tools erläutert, welche die Realisierung und Administration von `BetGetter/Viveo` vereinfachten und von mir implementiert wurden.

Config-File

Um das System flexibel zu halten, entstand ein Config-File, welches bei jedem Start eines Peers neu eingelesen wurde. Zum Beispiel stand darin, über welchen Port die Peers horchen sollen, wie viele Threads `Clippee` zur Verfügung haben soll und mit welchen anderen Peers ein neuer Peer einen Verbindungsaufbau versuchen soll. Die Standard-Wettquote von 3.0, welche später durch verschiedene Quoten ersetzt wurde, sowie das Startguthaben der Punkte waren unter anderem auch in diesem Ini-File vertreten. Weitere Details können direkt dem Config-File entnommen werden.

DataReader

Einzelne Spiele – zum Beispiel Finalspiele, welche zu Beginn einer Meisterschaft noch nicht bekannt sind – kann man bequem über das Administrator-Interface eingeben. Möchte man jedoch mehrere Spiele direkt einlesen, so gestaltet sich dieser Weg zu umständlich. Deshalb entstand der `DataReader`, welcher nicht nur Spiele, sondern auch `User` und `Teams` einlesen kann. Die Spiele, `User` und `Teams`

stehen dabei zeilenweise in einem CSV-File (Comma-Separated-Values). Eine Vorlage für ein solches CSV-Datafile befindet sich im CVS im Verzeichnis /div.

Parameter-Klassen

Clippee gleicht seine Objekte ab, indem es jedes Feld einer Klasse serialisiert. Für diese Serialisierung werden so genannte Parameterklassen benötigt, welche für einen bestimmten Feldtyp Methoden zur (De-)Serialisierung zur Verfügung stellen. Clippee bietet jedoch nur drei solche Parameterklassen an (`String8Parameter`, `StringParameter` und `IntParameter`). Wir benötigten noch weitere. Deshalb implementierte ich Parameterklassen für folgende Datentypen:

- `boolean` - `BooleanParameter`
- `Date` - `DateParameter (*)`
- `double` - `DoubleParameter`
- `float` - `FloatParameter (*)`
- `long` - `LongParameter (*)`

Die mit einem Stern markierten Parameterklassen wurden im System verwendet und haben diesen Praxistest bestanden. Für eine definitive Aufnahme in das Clippee-Framework sollten jedoch alle zum Beispiel mit Unit-Tests genau auf Herz und Nieren überprüft werden.

Fortlaufende Arbeiten

Bugfixing

Während der Entwicklungsphase gab es zwei grössere Probleme zu lösen. Das eine war, dass sich die Objekte irgendwie nicht abglichen. Dies deshalb, weil nach einem `updateObject()` eines Objektes auf einem Peer der Lock dieses Objektes auf dem gleichen Peer weiter bestand. Mit der Hilfe von Ruedi Arnold und Keno Albrecht konnten wir dieses Problem lösen – nach dutzenden von Stunden vergeblichen Probierens. Man musste nach einem `updateObject()` das Objekt jeweils explizit wieder freigeben.

Das andere Problem bestand darin, dass wir eine veraltete Version von Clippee verwendeten. Darin war ein eigentlich bekannter Bug in den Serialisierungsmethoden von `DataObject` enthalten, welchen wir aber relativ rasch beheben konnten.

Daneben existierten natürlich weitere kleinere Probleme, welche wir allesamt innert nützlicher Frist lösen konnten. Zum Beispiel gab es meinerseits Flüchtigkeitsfehler in den (De)Serialisierungsmethoden der Objekte.

Testdaten

Während der Implementationsphase unseres Wettsystems wurde es immer wieder nötig, das jeweils bisher Erreichte zu testen. Dazu waren Testdaten mit virtuellen Spielen nötig, welche auch zu den jeweiligen Testzeiten statt fanden. Die Spiele wurden dann mit dem `DataReader` ins System eingelesen.

Resultateingabe

Während der Testphase gab ich jeweils frühmorgens die Resultate der NHL-Spiele der jeweiligen Nacht über das Administrator-Interface von Stefan ein.

Danach verfasste ich jeweils entsprechende Newsbeiträge der vergangenen Spielnacht für die Startseite von viveo.ethz.ch und erstellte die Rangliste unmittelbar nach der Resultateingabe.

Statistiken

Nach Ablauf der Testwoche war es meine Aufgabe, aus den Statistik-Log-Einträgen und aus den im System vorhandenen Objekten aussagekräftige Zahlen herzuleiten. Die Log-Einträge, welche über verschiedene Files verteilt waren, fasste ich in einem File zusammen und bearbeitete sie so, dass ich sie anschliessend mit OpenOffice in einer Pivot-Tabelle auswerten konnte. Um an die Objekte heranzukommen, schrieb ich den `StatisticAnalyzer`, welcher sich mit dem laufenden System verband und die Objekte abglich. Anschliessend wurden diese Objekte mittels verschiedener Methoden analysiert. Jedoch in einer Art und Weise, die nicht an strukturiertes Programmieren erinnert.

Aus den Daten des `StatisticAnalyzer` und aus der Pivot-Tabelle entstand der statistische Bericht von Viveo, der im Manual zu finden ist.

Erfahrungen

Projektarbeit im Team

Es war toll, in einem solchen Team, wie wir es waren, zu arbeiten. Wir ergänzten uns ausserordentlich gut und jeder konnte seine Fähigkeiten und Stärken optimal einsetzen. Das gegenseitige Vertrauen war enorm; und dies auch zu Recht. Es kam nicht vor, dass jemand etwas Versprochenes nicht erfüllte und so die anderen Teammitglieder an der Nase herumführte. Jeder erfüllte seine ihm zugetragenen Aufgaben zum Wohle des ganzen Teams.

"Viele Personen" gleich "viele Meinungen": das war auch bei uns so. Jedoch fanden wir immer einen guten Kompromiss mit dem alle leben konnten. Dass es viele Meinungen gab, war auch nützlich. Eigene Ideen konnte man mit Inputs von den anderen in Richtungen weiterentwickeln, auf welche man alleine nicht gekommen wäre.

Tools

Eclipse/CVS

Das Gespann "Eclipse-CVS" erwies sich als grosse Hilfe. Ein grösseres Projekt ist ohne benutzerfreundliche Entwicklungsumgebung wie Eclipse gar nicht mehr zu realisieren, ohne dass man ständig den Überblick verliert. Ein Versioning-System wie CVS ist beim Programmieren in Teams nicht wegzudenken.

BSCW

Das vom Fraunhofer Institut für Angewandte Informationstechnik entwickelte BSCW (Basic Support for Cooperative Work) erwies sich als nützliches Tool. Insbesondere die Möglichkeit, Dateien abzulegen, damit diese von allen Teammitgliedern gelesen, bearbeitet und ergänzt werden können, erwies sich als überaus brauchbare Hilfe. Man musste nicht mehr mühsam die richtige Datei in der richtigen Version aus den knapp 300 Mails herausuchen, sondern konnte sie direkt aus dem Shared Workspace holen.

Die Kalenderfunktion fand ich hingegen weniger brauchbar, da unser Projekt - was die Deadlines und die Teamgrösse betrifft - überschaubar blieb und sie deshalb aus meiner Sicht eigentlich gar nicht nötig gewesen wäre.

Fazit

Das vom Fraunhofer Institut für Angewandte Informationstechnik entwickelte BSCW (Basic Support for Cooperative Work) erwies sich als nützliches Tool. Insbesondere die Möglichkeit, Dateien abzulegen, damit diese von allen Teammitgliedern gelesen, bearbeitet und ergänzt werden können, erwies sich als überaus brauchbare Hilfe. Man musste nicht mehr mühsam die richtige Datei in der richtigen Version aus den knapp 300 Mails heraussuchen, sondern konnte sie direkt aus dem Shared Workspace holen.

Die Kalenderfunktion fand ich hingegen weniger brauchbar, da unser Projekt - was die Deadlines und die Teamgrösse betrifft - überschaubar blieb und sie deshalb aus meiner Sicht eigentlich gar nicht nötig gewesen wäre.

Einleitung

Im Wintersemester 2003/04 habe ich an der Veranstaltung “Labor für Verteilte Systeme” (Lab) teilgenommen. In einer fünfköpfigen Gruppe von Studenten sollten wir ein Projekt durchführen, welches zum Ziel hatte, ein “Online Betting System” aufzubauen. Genau diese beiden Punkte, Projekt- und Gruppenarbeit, waren für mich die Motivation, am Lab teilzunehmen (für mich war die Veranstaltung im Gegensatz zu den meisten andern Teilnehmern nicht obligatorisch).

Der folgende Bericht soll als persönlicher Rückblick auf das Projekt dienen. In einem ersten Teil ist ihm ist zu entnehmen, welches meine Tätigkeitsbereiche innerhalb des Projektes waren. Im darauf folgenden Teil gehe ich detaillierter auf Bereiche ein, die mich vor Probleme gestellt haben und bei denen ich besonders wertvolle Erfahrungen sammeln konnte. Der letzte Teil ist der für ETH-Informatik-Studenten doch eher ungewohnten Arbeitsweise des Teamworks gewidmet.

Ich bin froh, am Lab teilgenommen zu haben, denn meine Erwartungen sind dahingehend erfüllt worden, dass ich Erfahrungen im projektorientierten Teamwork machen konnte. Darüber hinaus war die Arbeit spannend und dank dem guten Team sehr angenehm.

Aufgabenbereiche

Hyperlink-Struktur Userinterface

Ziemlich früh im Projekt kristallisierte sich heraus, dass eine Auftrennung des Webbasierten GUIs in einen geschützten Administrator- und einen öffentlichen Benutzerteil sinnvoll ist. Diese beiden Teile sollten völlig unabhängig von einander funktionieren und nur indirekt mittels des ApplicationBeans (Interface zwischen Back-End und Front-End) miteinander interagieren.

Mit Christian zusammen plante ich die logische Struktur des Benutzerteils des GUIs, die so genannte Hyperlinkstruktur.

Webbasierte Userinterface

Für die Realisierung des gesamten GUIs waren Stefan und ich verantwortlich. Zusammen haben wir die Hilfe/Spielregel-Texte und den Dokumentations-Teil (Impressum) erstellt, für den Grossteil der Arbeit hielten wir uns an die sich anbietende Trennung zwischen Admin- und Userbereich. Ich übernahm die Detailplanung und Implementierung des letzteren.

Graphik-Design / Usability

Am Anfang des Projektes machten alle Teammitglieder Vorschläge für das Corporate Design unserer Site. Die meistversprechenden entwickelte ich weiter und erstellte dazu passende Graphiken und Style-Sheets.

Von entscheidender Bedeutung für die Güte des Systems war die Benutzbarkeit. Um diese sicherzustellen, führte ich einige Usertests mit unbeteiligten Freunden (nicht nur Informatiker) durch.

Umfrage Testwoche

Am Ende der Evaluationswoche führten wir eine kleine Online-Umfrage durch, deren Resultate ich auswertete.

Installation/Konfiguration Wllaps

Die drei Rechner wllap13, wllap14 und wllap15, welche wir für fortlaufende Tests und während der Testwoche für den Live-Einsatz verwendeten, mussten mit den entsprechenden Installationen versehen werden. Nachdem die Suche nach geeigneten Tools abgeschlossen und die Entwicklungs- und Laufzeitumgebung bestimmt war, installierte und konfigurierte ich diese auf den genannten drei Rechnern und später auch auf dem Rechner wllap11, der nach der Testwoche weiter online blieb, damit sich die Teilnehmer weiterhin News und Ranglisten anschauen konnten.

Die Umgebung bestand im Wesentlichen aus Eclipse, Tomcat, den entsprechenden Eclipse-Plugins sowie verwendeten Java-Libraries.

Installations-Anleitung

Da ich die Umgebung schon aufgesetzt hatte, lag es nahe, dass ich dies auch entsprechend dokumentierte.

Endpräsentation

Zusammen mit Andreas bereitete ich die Endpräsentation vor und präsentierte den zweiten Teil derselben. Thematische Schwerpunkte dabei waren erstens die Testwoche, in diesem Zusammenhang aufgetretene Probleme, die daraus resultierenden Lehren und statistische Auswertungen und zweitens die Planung und Realisierung von Quoten in unserem System.

Probleme/Erfahrungen

Installation

Als recht schwierig stellte sich die Suche nach einem Editor heraus, der ein komfortables JSP-Programmieren erlaubt. IntelliJs IDEA schien am besten

geeignet, war jedoch nicht gratis erhältlich. Die frei erhältliche Beta-Version hatte ihre Tücken, ausserdem war das mitgelieferte CVS-Tool mächtiger als jenes von Eclipse, was mich als CVS-Neuling überforderte. Da nicht nur die Suche und Installation eines Editors sondern auch die Integration des Tomcat-Servers in selbigen problematisch war, verbrauchte ich sehr viel mehr Zeit dafür als mir lieb war.

Das Problem der Tomcat-Integration, welches darin bestand, dass diese auf einem einzigen (meinem privaten) Rechner funktionierte und auf allen anderen nicht, löste sich, als ich merkte, dass Tomcat 4.1.29 gegenüber Tomcat 4.1.27 um einen Bug angereichert worden war. Lange Zeit war mir überhaupt nicht bewusst gewesen, dass ich verschiedene Versionen benutzte.

Dass ich letzten Endes (nach Tagen erfolglosen Installierens) Eclipse den Vorzug vor IDEA gab, lag an drei Dingen: Erstens war Eclipse zu dem Zeitpunkt, als ich mich zu entscheiden hatte, bereits auf allen Rechnern installiert, zweitens war mir Eclipse lieber, weil ich mit IDEA bereits ein CVS-Modul zerstört hatte und drittens war Eclipse ein Garant für die Kompatibilität mit den andern Team-Mitgliedern, welche mehrheitlich auch Eclipse benutzten.

Mit einigen Eclipse-Plugins liess sich auch ein gewisser Programmier-Komfort erreichen, allerdings denke ich im Nachhinein, dass IDEA trotz diesen Plugins für die JSP-Entwicklung besser geeignet gewesen wäre. Bleibt zu hoffen, dass die besagten Plugins noch entscheidend verbessert werden.

Zusammenfassend ist zu sagen, dass es ziemlich schwierig ist, auf drei oder mehr Rechnern exakt die gleiche Kombination von Programmen zu installieren, wenn man – wie ich es tat – naiv an die Sache rangeht und nicht damit rechnet, dass ein Anbieter seine Website genau im entscheidenden Zeitraum updatet und einem unbemerkt eine neue Version unterjubelt.

GUI

Bei der Realisierung des GUIs war die Vorgehensweise problematisch. Meine Pläne stellten sich im Nachhinein als zu wenig detailliert und oft als nicht realistisch heraus, so dass das Endprodukt zuweilen recht stark von den ursprünglichen Plänen abweicht. So war zum Beispiel das Navigations-Menu, wie es nun im System zu sehen ist, anfänglich gar nicht geplant!

Meine Idee war gewesen, dass sich das GUI in einem ersten Schritt auf einer logischen (Verlinkungs-) Ebene völlig unabhängig von Darstellungsaspekten planen lässt. Dies stellte sich aber als falsch heraus, denn Grösse und Form der einzelnen GUI-Komponenten (also die Darstellung) beeinflussen auch die logische Struktur der Oberfläche.

Eine vernünftiger Herangehensweise dürfte wohl die folgende sein:

- Welche Information muss dargestellt werden? (Inhalt)
- Wie kann diese Information dargestellt werden? (Darstellung)
- Wie (und auf wie vielen Screens) können die verschiedenen Informations-Komponenten angeordnet werden? (Darstellung)
- Wie werden die verschiedenen Screens verlinkt? (Struktur)

Alle diese Fragen sind natürlich unter den Gesichtspunkten Brauchbarkeit, Benutzbarkeit und Schönheit zu beantworten.

Eigentlich liegt es auf der Hand, dass man vor der Planung eines solchen Projektes auch entsprechende Fachliteratur zu Rate zieht. Ich zögere diesbezüglich oft, wenn ich mich unter Zeitdruck fühle, denn die Beschaffung und die Einarbeitung in die Literatur verbraucht Ressourcen, von denen man nicht weiss, ob man sie später zurückerhalten wird.

Nachdem ich wie im vorhergehenden Abschnitt über Installations-Probleme beschrieben zeitlich bereits hinter meiner Planung lag, wollte ich dieses Risiko nicht eingehen.

Präsentation

Eine Erwähnung wert scheinen mir auch die Probleme, vor die mich die mich die Endpräsentation stellte. Nur kurze Zeit, etwa zwei Tage lang, dafür umso intensiver, litt ich unter der Angst davor, im entscheidenden Moment zu nervös zu sein und damit alles zu vermasseln. Das war übrigens auch der Grund für unser zu spätes Eintreffen an der Präsentation; in der Annahme, dass die Vorträge erst um 13.15 beginnen würden, wollte ich keinen Moment zu früh da sein, um mich möglichst lange ablenken zu können.

Letztendlich hielt sich die Nervosität in Grenzen und die Präsentation gelang besser als befürchtet. Hoffentlich erinnere ich mich daran, wenn ich das nächste Mal einen Vortrag halte!

Zusammenarbeit

Team

Die Arbeit im Team hat mir sehr viel Spass gemacht. Allein schon die Tatsache, dass mir drei meiner vier Kollegen vor Beginn des Labs unbekannt gewesen waren, machte die Zusammenarbeit spannend. Jedes Teammitglied hatte seine Stärken und seine Schwächen und glücklicherweise konnte ich von den Stärken der andern viel profitieren, während die meisten ihrer Schwächen kein Problem für mich darstellten.

Ich denke, dass wir Glück hatten in der Zusammenstellung unserer Gruppe und uns oft in idealer Weise ergänzten. Ich möchte behaupten, dass wir etwas geschaffen haben, was jeder einzelne von uns, vier mal geklont, nicht zu Stande gebracht hätte.

Betreuer

Auch die Zusammenarbeit mit Ruedi war sehr angenehm. Freundschaftlich und humorvoll begleitete er uns durch das Semester, trotzdem blieb er stets fordernd und motivierend. Das Nachtessen bei ihm zu Hause war (aus sozialer Sicht) der Höhepunkt des Labs und wohl auch unserer Gesangskarriere! □

Mein Beitrag

Neben verteilten Systemen interessiert mich speziell auch das Programmieren im Grossen, wovon einem in der Vorlesung Informatik IV einige Grundlagen vermittelt wurden. Nachdem ich mich bereits während eines Praktikums intensiv mit flexiblem und erweiterbarem Systemdesign auseinandergesetzt habe, besitze ich diesbezüglich mittlerweile etwas Erfahrung und einige klare Vorstellungen, welche ich als Mitglied des Backend-Teams beim Design unseres Systems mit einbringen konnte. Mir war hierbei speziell die einfache Anpassbarkeit an künftige Anforderungen wichtig, wie zum Beispiel verschiedene Arten von Wetten.

Zu meinen Aufgaben gehörte dann auch das Implementieren verschiedener Backend-Klassen und Methoden, was grösstenteils keine übermässig grosse Herausforderung darstellte. Diese bestand mehr darin, unser System mit Clippee zu verteilen. Ich habe mich dann auch einiger der aufgetreten Probleme angenommen, und Teile von Clippee recht intensiv studiert. Dies resultierte nicht immer in produktiver Arbeit, so habe ich zum Beispiel die Update-Funktion von Clippee, welche Daten auf allen Peers aktualisiert, etwas genauer unter die Lupe genommen. Dabei habe ich bemerkt, dass Clippee nur mit den eigenen DataObjects und nicht mit Subklassen davon direkt umgehen kann, was uns zuvor nicht bewusst war. Ich habe mir dann überlegt, wie man dies ändern könnte, denn ein typisierter Datenraum würde meiner Meinung nach einige Vorteile bringen. Einerseits müsste man nicht mehr von jedem DataObject den Namen der konkreten Subklasse kennen, nur um es verwenden zu können, und andererseits wären Abfragen auf Clippee der Art "Gib mir eine Liste aller User-Objekte" möglich. Letzteres würde einiges an Objekt-Verwaltungsaufwand in unserem Code ersparen. Clippee in diese Richtung hin zu erweitern, wäre zum einen ziemlich aufwändig zu programmieren gewesen und hätte noch einiges mehr an Testaufwand mit sich gebracht. Da es für unsere Zwecke nicht schwierig war, um diese Limitation herum zu programmieren, und das oberste Ziel ein lauffähiges System war, wurde Clippee so belassen.

Dieser Entscheid machte dann den oben erwähnten Objekt-Verwaltungsaufwand nötig. Wir müssen separat von Clippee Listen unter anderem aller User und aller Matches führen. Damit diese Listen auch immer aktuell sind, habe ich einen zusätzlichen Clippee-Listener implementiert, welcher alle neuen und "interessanten" Objekte in die passende Liste einfügt.

Ein weiteres Thema, mit dem ich mich beschäftigt habe, ist Persistenz. Solange zu jeder Zeit mindestens ein Peer läuft, sind alle Daten irgendwo vorhanden. Da sich Clippee-Daten bloss im Arbeitsspeicher befinden, würden sie bei einem Totalausfall aller Peers jedoch verloren gehen. Dies wäre bei einem produktiven System natürlich inakzeptabel, weshalb wir beschlossen, die Daten zusätzlich auf der Festplatte zu speichern. Hierfür habe ich eine Erweiterung auf Clippee-Ebene implementiert, welche ermöglicht, jede Änderung am Datenraum zu loggen oder auch eine Momentaufnahme des Datenraumes in eine Datei zu "dumpen". Das Logging funktioniert so, dass nur derjenige Peer ein Update loggt, von welchem es ausging. Sollte tatsächlich einmal ein Totalausfall eintreten, muss einfach jedes Logfile wieder eingelesen werden. Die Reihenfolge spielt keine Rolle, da Clippee jedem Update eine Versionsnummer zuordnet und nur die neuste Version eines Objektes behalten wird.

Um die Persistenz implementieren zu können, habe ich mich mit einigen für mich neuen Java API's auseinandergesetzt, namentlich mit `java.nio`, der neuen, block-orientierten IO-Bibliothek, sowie `java.util.regex`, um die Logfiles mittels Regular Expressions wieder einlesen zu können. Die eigentlichen Daten, die geloggt werden sollen, existieren als Byte Array und werden vor dem Schreiben noch Base64 kodiert, damit der Regular Expression Parser mit Sicherheit nur ASCII-Zeichen antrifft, was korrektes Einlesen garantiert.

Das neue Wissen konnte ich dann gleich wieder einsetzen, als es darum ging, separate Logfiles für Statistiken zu erstellen. Für diese Statistiken logge ich alle interessanten von VIVEO aufgerufenen Methoden inklusive aller Parameter. Chregi hat sich im Anschluss darum gekümmert, die geloggt Daten zu interpretieren, um zum Beispiel Aussagen machen zu können, wie häufig ein VIVEO-Benutzer sich in das System einloggt.

Mein letzter grösserer Brocken war die Durchführung von Performancetests für unser System, speziell im Hinblick auf die Schlusspräsentation. Die Idee war, über eine HTTP-Verbindung VIVEO anzusprechen und die Zeit zu messen, die benötigt wurde, um gewisse Aktionen durchzuführen. Es sollte dann die Anzahl simultaner Verbindungen stetig erhöht werden, um Aussagen über die maximale Anzahl gleichzeitiger Benutzer machen zu können. Ursprünglich wollte ich messen, wie lange es dauert, einen neuen Benutzer zu erstellen und einige Wetten für ihn abzuschliessen. Der Einfachheit halber, und da sowohl einen Benutzer erstellen als auch eine Wette abschliessen im Wesentlichen bedeutet, ein Objekt in Clippee einzufügen, beschloss ich dann, bloss die benötigte Zeit zu messen, einen neuen Benutzer zu erstellen.

Leider konnte ich nicht innert nützlicher Frist brauchbare Resultate erzielen. Die benötigte Zeit zum Erstellen eines Benutzers schwankt laut meinen Messungen im Bereich zweier Grössenordnungen, unabhängig von der Anzahl gleichzeitiger Anfragen, während sich die Geschwindigkeit beim Benutzen des Systems via Browser meist recht zügig anfühlt. Gemessen wird allerdings bloss die Zeit, die eine Methode aus `java.net` benötigt, um die Anfrage durchzuführen und eine Antwort zu erhalten. Insofern könnte ich mir vorstellen, dass mit der Verwendung eines anderen HTTP-Clients realistischere und brauchbarere Resultate erzielt werden könnten. Dies auszuprobieren lag leider aus Zeitgründen nicht mehr drin. Es stellt sich allerdings auch die Frage, wie aussagekräftig auch realistische

Resultate wären, da wir kein Vergleichssystem besitzen, welches dieselbe Aufgabe zum Beispiel mit einer verteilten Datenbank an Stelle von Clippee löst.

Schwierigkeiten

Einige aufgetretene Probleme habe ich bereits oben erwähnt. Die grösste Schwierigkeit bestand für mich aber in der Vielschichtigkeit unseres Systems und in der Verteiltheit, wobei einige Schichten nicht von uns selbst entwickelt wurden, insbesondere Clippee und der als Java Server verwendete Tomcat. Es ist im Allgemeinen bestimmt einfacher, auf ein selbst geschriebenes System aufzubauen, als auf bestehenden Code, doch mit den Autoren von Clippee hatten wir zum Glück die Experten auf diesem Gebiet im selben Hause, und eine Alternative zu Clippee selbst zu implementieren wäre weder sinnvoll, nötig noch zeitlich realistisch gewesen.

Interessant war es festzustellen, wie einige Dinge beim Einsatz auf nur einem Peer problemlos funktionierten und dann beim Verteilen auf mehrere Peers Schwierigkeiten bereiteten. Zu Beginn war es beispielsweise nicht möglich, von verschiedenen Peers schreibend auf dasselbe Objekt zuzugreifen, bis wir feststellten, dass die von Clippee für das Schreiben akquirierten Locks explizit freigegeben werden müssen. Die Verteiltheit macht das Debuggen bestimmt nicht einfacher.

Eine weitere Schwierigkeit bestand darin, dass einzelne Aufgaben relativ spät erwähnt wurden, was dann mit der bisherigen Planung nicht unbedingt problemlos vereinbar war. Das Problem war vielleicht, dass fast alles schön zu Beginn erwähnt wurde, und lange Zeit auch nichts neues dazukam. Wahrscheinlich habe ich mich etwas zu sehr an die sehr gute Betreuung gewöhnt, womit dann die paar wenigen suboptimalen Dinge etwas unerwartet kamen.

Teamwork

Die Zusammenarbeit in unserem Team empfand ich als sehr angenehm. Als erstes haben wir uns in Frontend- und Backend-Teams aufgeteilt, was ohne grössere Reibungen gelang. Im Folgenden geschah die Entwicklung relativ autonom, nachdem in den Sitzungen zu erledigende Arbeiten bestimmt und verteilt wurden. Jeder war dann verantwortlich, dafür zu schauen, dass seine Komponenten funktionierten, und musste man mit jemandem etwas besprechen, wurde das jeweils direkt gemacht, ohne das ganze Team miteinzubeziehen. Alles in allem eine recht dezentralisierte Organisation, was meines Erachtens gut zum Trend weg vom Client-Server- und hin zum Peer-to-Peer-Modell passt, und damit auch zum Thema unseres Labs.

Es gab aber auch Momente intensiveren Zusammenarbeitens, etwa während unserer Intensiv-Testrunde vor der öffentlichen Testwoche, als wir zusammen im Lab-Raum sassen und unser System mit Wetten bombardierten, um mögliche Bugs oder Verbesserungsmöglichkeiten aufzuspüren. Das war richtig spassig!

Tools

Ein wichtiges Tool war BSCW, Basic Support for Cooperative Work, eine Web-basierte Groupware, wo wir alle Dokumente ablegen konnten, so dass jeder darauf zugreifen konnte. Ich fand dies sehr hilfreich, vor allem für die Sitzungs-Protokolle, die sich bei zwei Sitzungen in der Woche doch recht anhäuferten.

Als Entwicklungsumgebung habe ich IntelliJ IDEA verwendet, während der Rest des Teams auf Eclipse gesetzt hat. Beides sind sehr gute Tools, und da sie ebenfalls beide File- und nicht Repository-basiert funktionieren, sowie dank der Versionsverwaltung mit CVS, stellte das Verwenden der verschiedenen Umgebungen kein Problem dar.

Wie oben erwähnt wurde CVS zur Versionsverwaltung eingesetzt, was meiner Meinung nach absolut unverzichtbar ist für jedes Projekt mit mehr als einem Entwickler (und sicher auch eine gute Idee ist für ein Einzelprojekt). Es gäbe sicher Alternativen zu CVS, doch hat sich CVS gut bewährt und führte sicher auch dank der vorhandenen Unterstützung in den Entwicklungsumgebungen zu keinen grösseren Problemen.

Weiters haben wir beschlossen, uns an einheitliche Coderichtlinien zu halten und den Code mit JavaDoc zu dokumentieren. Beides sind meiner Meinung nach sehr sinnvolle Beschlüsse.

Fazit

Alles in allem hat mir das Lab gut gefallen, obwohl der Aufwand mit gut 300 Arbeitsstunden sowohl mehr als erwartet als auch mehr als erhofft war. Ich habe einiges gelernt im Bereich von Verteilten Systemen und auch einige neue Java API's kennengelernt. Auch fand ich die Arbeit und Aufteilung in der Gruppe interessant, grössere Gruppenarbeiten sind ja während des Informatikstudiums nicht unbedingt üblich.

Ich bin auch mit unserem Resultat zufrieden. Unser System funktioniert und hat den ersten richtigen Einsatz während der Testwoche gut überstanden. Um unser System wirklich produktiv einsetzen zu können, wird allerdings noch etwas Arbeit nötig sein, vor allem auch Tests der Sicherheit.

Der Grösste Erfolg war bestimmt die hohe Anzahl Benutzer während der Testwoche, welche uns zu einem Festessen mit Älplermagronen verhalfen!

Meine Aufgabe

Die DNS-Auflösung der URL (Abbildung von einer URL auf eine Server IP-Adresse aufgrund eines DNS-Eintrags) sollte jederzeit garantiert werden können. Um die Ausfallsicherheit zu “garantieren”, schlug ich Round-Robin vor und machte weitere Abklärungen über eine Realisierung:

Round-Robin soll die Belastung auf alle Server mehr oder weniger gleichmässig verteilen. Die Statistik der Testwoche konnte dies auch bestätigen. Dadurch soll vermieden werden, dass ein einzelner Rechner alle Anfragen zu bearbeiten hat und durch Überlastung ausfällt.

Hingegen können Ausfälle von Servern mittels Round-Robin nicht behandelt werden. Dazu ist eine stetige Überwachung der Server notwendig und ein Mechanismus, welcher die ausgefallenen Servern aus dem DNS-Eintrag entfernt. Die Überwachung kann in Clippee geschehen und mittels Dynamic DNS (DDNS) der ausgefallene Servern aus dem DNS-Eintrag entfernt werden. Als Protokoll für eine Lösung wäre *bind* vorzuschlagen.

Trotz der bisherig beschriebenen Massnahmen kann es noch vorkommen, dass eine Aufruf der Homepage nicht gelingt, weil aufgrund von Caching (Zwischenspeichern) noch mit alten Werten gearbeitet wird. Das Setzen von Parametern wie *no caching* oder *TTL* niedrig/Null löst das Problem theoretisch, aber nicht alle DNS-Client (Browser, DNS-Server) achten darauf. Kann man Webbrowser und DNS-Server selber kontrollieren wie z. B. unter Laborbedingungen, so funktioniert diese Lösung, wie wir in einem kurzen Test zeigen konnten.

Im Team mit Fabian erstellte ich das Frontend. Zuerst hatte ich dazu mit Chregi (Backend-Part) einen Prototyp erstellt, um die Interaktion von Tomcat und Clippee zu testen. Nach dem gemeinsamen Erstellten der Interface-Definition mit Fabian, war ich primär für den Administrator Bereich des Webauftritts zuständig. Schwierigkeiten entstanden mit der Konfiguration von Eclipse und dem Einbinden von Tomcat und des Lomboz Plugin. Des weiteren stellten sich einige Knacknüsse mit der Verlinkung der einzelnen Projekte (clippee, dcg, betgetter, viveo) und der Bereitstellung der Klassen. Dabei konnte ich glücklicherweise oft auf Fabian zurückgreifen.

Ein offener Punkt ist, dass die Darstellung der mit Stylesheet entwickelten JSP-Seiten unter Mozilla nicht korrekt angezeigt wird.

Und kleinere Maleurs traten auch auf, wie z. B. dass Veränderungen an einem Match nur lokal und nicht auf allen Servern gespeichert wurde, weil das Aufrufen der update-Methode vergessen wurde.

Folgende Einschränkungen sind bei der Bearbeitung von Matchdaten vorhanden:

- Matchdaten und Quoten sind nur bis zu Spielbeginn bearbeitbar
- Resultate können erst nach Spielbeginn eingegeben werden

Erfahrungen

Die Entwicklung der Webseiten wurde erschwert, weil in Eclipse keine sinnvolle Unterstützung für das Debugging von Java Server Pages (JSP) existiert. Das zugrundeliegende Problem ist, dass in Eclipse nur Java-Code debuggen möglich ist. Grundsätzlich könnte also jede JSP in ein Servlet und somit eine Java-Klasse umgewandelt werden. Auf diesen Umweg wurde verzichtet.

Als Frontendler merkte ich teilweise auch die Abhängigkeit vom Backend stark. V. a. in der Startphase, als ich noch nicht so geübt mit den Tools war, verbrauchte ich teilweise unnütze Zeit, um im nachhinein herauszufinden, dass die Funktion so nicht existierte. In diesem Zusammenhang lernte ich, dass es manchmal besser ist, etwas vorläufig zu unterbrechen und bei nächster Gelegenheit nachzufragen. Hingegen überraschte mich die in der Theorie hochgelobte Trennung von Anwendungsprogramm (hier JSP-Seite mit ApplicationBean) und der zugrundeliegenden Logik (extended Clippee). Eigentlich reichte für die Entwicklung des Frontends das Definieren des Interfaces und das ganze Backend konnte vernachlässigt werden.

Auch wenn ich Fächer wie Projektplanung besucht hatte und mich ein wenig mit der Organisierung eines Projektes auskannte, so konnte ich viel in dieser Richtung lernen. Z. B. wenn auch eine Dokumentation der Ziele des Projekts (Use Cases, Interface) und ihrer Ausprägung einiges Zeit benötigten, so halfen sie, ein einheitliches Verständnis zu geben. Sie konnten dann für die Entwicklung als Leitlinie verwendet werden.

Neben der vielen Zeit, die für Meetings aufgewendet werden musste, sah ich zum ersten Mal, dass auch die Konfiguration eines Systems mit einem riesigen Aufwand verbunden sein kann.

Teamwork

Ich fand das Klima innerhalb des Team sehr gut. Man unterstützte sich gegenseitig, wo Hilfe gerade nötig war. Die häufigen Sitzungen schweissten uns neben der eigentlichen Aufgabe stark zusammen. Dass es dabei auch lustig zu und herging, lockerte die Stimmung sicher auf. Besonders anzusprechen war die gemeinsame Test des Systems mit einer Game Session im WLAN-Raum, das war ein Gaudi! Auch das eher kollegiale denn hierarchische Verhältnis zu Ruedi, unserem Berater, gefiel mir.

Die Aufteilung in ein Team für das Frontend und Backend war vom logischen Aufbau des Systems sinnvoll und liess eine Spezialisierung zu. Dadurch wurde die Zeit effizienter genutzt. Auch ohne eine Teamleitung funktionierte die Kommunikation innerhalb und zwischen den Teams gut. Dadurch wurde auch an die Eigenverantwortung jedes einzelnen appelliert. Dies klappte gut, weil jeder

seinen Teil zuverlässig erledigte. Traten Probleme auf, wurden diese (an der Sitzung) offen gelegt und eine zeitliche Verzögerung begründet. Ebenso wurden die freien Kapazitäten von der Teammitgliedern berücksichtigt und gegebenenfalls Aufgaben umverteilt.

Der Informationsaustausch wurde durch BSCW für die Dokumentationen und Email und ICQ für neue Informationen und auftretende Fragen auf hohem Niveau gehalten.

Tools

Das BSCW unterstütze uns, in der systematischen Ablegung von Dokumenten. Somit standen sie allen Teammitgliedern zur Verfügung. Gerade das Anfügen von Kommentaren zu den Dokumenten erlaubte ein einfaches Suchen. Schlussendlich gilt aber als Schwachpunkt zu erwähnen, dass die ganze Struktur nicht exportiert und zur Projekt-Dokumentation angefügt werden kann, ohne die BSCW-spezifischen Metainformationen wegzulassen

Eclipse und CVS waren eine grosse Hilfe und erlaubten einen Rückgriff auf funktionierende Programme... Dank des Lombok Plugin konnte auch die Entwicklung von JSP's direkt in Eclipse erledigt werden, welches die Handhabung erleichterte.

Persönliches Fazit, Kritik

Das Projekt hat Spass gemacht. Es bestärkt mich in meiner Meinung, beruflich in Projekten arbeiten zu wollen. Gerade die Erfahrungen von Team-Arbeit (soziales Wissen, Teamfähigkeit...) machen den Mehrwert eines Laboratorium gegenüber einer Semesterarbeit aus, wo nur das technische Wissen gefordert wird.

Neben dem Thema war für mich als besonderer Anreiz die Zielsetzung, das Spiel an der Fussball EM laufen zu lassen. Das Feedback und die rege Teilnahme während der Testwoche machen mich dazu zuversichtlich. Der erste Schritt ist nun gemacht, der Erfolg gebührend bei Ruedi gefeiert wo auf ein erfolgreiches Weiterführen angestossen wurde:

viva la viveo!