Reflexionsbericht

FST Projekt „Eventalizer“ Team 5

Verbundstudium, Master of Science
Fortgeschrittene Softwaretechnologien
SS 2012

Matthias Beer, Alexander Benölken, Martin Garrels, Felix Schulze Mönking, Felix Wessel, Patrick Wiebeler
11.06.2012
Dozent: Prof. Dr. Mario Winter

I INHALTSVERZEICHNIS

II Angabe zur Aufteilung der Prüfungs-leistung nach § 17 Absatz 2 MPO

Seiten	Autor	Matrikel-Nr.
1-7	Matthias Beer	7074314
8-15	Alexander Benölken	7076339
16-23	Martin Garrels	7076304
24-31-	Felix Schulze Mönking	7078989
32-38	Felix Wessel	7071784
39-46	Patrick Wiebeler	7076308

III Abkürzungsverzeichnis

Abkürzung	Erläuterung/Definition
BPMN	Business Process Model and Notation
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
DAO	Data Access Object
EPK	Ereignisgesteuerte Prozesskette
GPS	Global Positioning System
GUI	Graphical User Interface
http	Hypertext Transfer Protocol
DIE	Integrated Development Environment
JSP	Java Server Pages
MSDNAA	Microsoft Developer Network Academic Alliance
RUP	Rational Unified Process
UML	Unified Modeling Language

IV ABBILDUNGSVERZEICHNIS

V TABELLENVERZEICHNIS

1 Einleitung

Der vorliegende Reflexionsbericht entstand im Rahmen des berufsbegleitenden Masterstudiums im Verbundstudium der Fachhochschulen Köln und Dortmund. Ziel dieses Berichts ist die Reflexion des eigenen Verhaltens, Vorgehens und der Erfahrungen, die während des seminaristischen Softwareentwicklungsprojekts im Modul “Fortgeschrittene Softwaretechnologien” gewonnen wurden.

Inhaltlich richtet sich der Bericht an Leser mit Grundlagenkenntnissen im Bereich der Wirtschaftsinformatik und allgemeiner IT-Technik, insbesondere der Softwareentwicklung und des Projektmanagements. Es werden gewisse Fachtermini und Zusammenhänge vorausgesetzt. Die Verwendung von Anglizismen ist bei dieser Thematik unumgänglich.

Fokus dieses Berichts liegt zunächst auf eine Darstellung und Erläuterung der wesentlichen Entscheidungen, die zu Beginn des Projekts getroffen wurden und die das weitere Vorgehen im Projekt definiert haben. Die Analyse und Reflexion dieser Entscheidungen und die Ergebnisse der einzelnen Artefakte erfolgt im anschließenden Kapitel.

Nach der Analyse erfolgen zum Abschluss des Berichts eine zusammenfassende und übergreifende Reflexion des Projekts, sowie eine Darstellung der weiteren Entwicklungsschritte und Potentiale der Plattform.

2 Projektinitialisierung

2.1 Projektidee

Zu Beginn des Projektes musste eine Projektidee ausgearbeitet werden. Mittels eines kurzen Brainstormings wurden drei Projektideen kurz schematisiert und beschrieben:

- Plattform zum Werkzeugverleih
  Auf dieser Internetplattform sollten Werkzeuge ver- und gemietet werden können.
- Plattform zur Event-Organisation
  Auf dieser Internetplattform sollten durch Benutzer (Gruppen-)Events eingestellt werden können, an denen andere Benutzer wiederum teilnehmen können.
- Browserspiel
  Es sollte ein Mehrspieler-Spiel, welches im Browser läuft, entwickelt werden.

Die Idee, ein Browserspiel im Projekt zu entwickeln, wurde schnell verworfen, da weder das Know-how für Spieleprogrammierung noch die Zeit für eine - wenn auch nur prototypische - Implementierung vorhanden gewesen wäre. Die Idee, eine Plattform zur Event-Organisation zu entwickeln, auf der Benutzer Events einstellen und an Events anderer Benutzer teilnehmen können, fand bei allen Projektmitglieder schnell Gefallen. Diese Idee hätte eine Praxisrelevanz gehabt und stellte einen sinnvollen Anwendungsbereich da. Ferner würde dieses Projekt auch eine prototypische Implementierung zulassen. Daher erfolgte zeitnah die Festlegung auf diese Projektidee. Die Basis hierfür bildete jedoch eine noch sehr rudimentäre Vorstellung, bei welcher noch genügend Interpretationsspielraum in vielen Themenbereichen vorhanden war.

2.2 Aufgabenverteilung und –durchführung

Zu Beginn des Projektes wurde diskutiert, wie die Aufgabenverteilung aussehen sollte. Im Rahmen dieser Diskussion wurde gemeinschaftlich entschieden, dass es keine auf die Projektmitglieder aufgeteilten Aufgabenbereiche geben soll und es Ziel des Projektes sein sollte, dass jeder möglichst in alle Bereiche des Softwareentwicklungsprozesses beteiligt ist. Um dem gerecht zu werden sollten die Aufgaben gleichmäßig aufgeteilt werden und jedes Teammitglied sollte in möglichst jedem Aufgabenbereich der Softwareentwicklung, d.h. angefangen bei der Anforderungsermittlung, über die Softwarespezifizierung, die Architekturkonzeption, den Entwurf, bis hin zu der Implementierung und der Qualitätssicherung einmal tätig werden.

2.3 Entscheidungskompetenz

Durch das Fehlen eines Auftraggebers wurde zu Beginn des Projektes auf einen Projektleiter, der die entsprechende Kommunikation zwischen Auftraggeber und der Projektgruppe übernommen hätte, verzichtet. Innerhalb der Projektgruppe wurden keine Hierarchieeben aufgebaut, so dass alle Projektmitglieder in der Entscheidungsfindung gleichberechtigt waren. Im Rahmen der Umsetzung einzelner Artefakte war der jeweilige Bearbeiter in seiner Entscheidung frei. Sofern eine Entscheidung Auswirkungen auf andere Artefakte hat, so wurde diese Entscheidungen im Jour fixe vorgestellt. Grundsätzliche und für den weiteren Projektverlauf betreffende Entscheidungen wurden grundsätzlich im Jour fixe vom gesamten Team mehrheitlich entschieden.

2.4 Jour fixe

Beim Kick-off des Projektes wurde ein Jour fixe, d.h. ein fester, wöchentlicher Termin für das Treffen des Projektteams festgelegt. Der Mittwoch wurde als Tag ausgewählt, an dem aktuelle Themen besprochen, Ergebnisse vorgestellt, Entscheidungen getroffen und weitere Aufgaben verteilt wurden. Neben dem Mittwoch wurde festgelegt, sich bei Bedarf auch freitags zu treffen. Für Projektmeetings wurde ein Büroraum mit Beamer, Flipchart, etc. reserviert. Die Dauer der Treffen bewegte sich, in Abhängigkeit von der Anzahl der Themen bzw. Entscheidungen oder ob innerhalb dieser Treffen Ergebnisse vorgestellt, ausgearbeitet und finalisiert wurden, zwischen 30 Minuten und drei Stunden. Jedes Meeting wurde dabei protokolliert, um auch fehlenden Mitgliedern die Möglichkeit zu bieten, die besprochenen Themen, die Ergebnisse, sowie die Entscheidungen nachverfolgen zu können.

2.5 Infrastruktur

Zur Umsetzung des Projekts musste zunächst eine Projektinfrastruktur als Arbeitsumgebung aufgebaut werden. Die Projektinfrastruktur besteht aus einer Dateiaustauschplattform zur Synchronisation der Artefakte und Ergebnisse innerhalb der Projektgruppe, der Modellierungswerkzeuge für die Anforderungsermittlung und der Softwarespezifikation, sowie der Softwareentwicklungsplattform für die Implementierung Im Folgenden wird auf diese drei Bestandteile näher eingegangen.

2.5.1 Dateiaustauschplattform

Arbeitsergebnisse sollten von Beginn an zeitgemäß über einen zentralen Server verwaltet und verteilt werden. Von besonderer Bedeutung war, dass alle Projektmitglieder jederzeit Zugriff auf alle Dokumente der anderen Projektmitglieder haben sollen. Es wurden das kostenlose Quellcode-Repository GitHub[1], Google Docs[2] sowie der Cloudspeicherdienst Dopbox[3] zur Auswahl gestellt.

Über den verwendeten Dienstleister sollten Office Dokumente und UML-Diagramme, aber auch Quellcode und beliebige weitere Dokumenttypen ausgetauscht werden. Zudem sollte gerade für Quellcode-Dateien die Synchronisierung kontrolliert erfolgen, also nicht automatisiert nach einem Speichervorgang. Bei genauerer Betrachtung der Anbieter konnte die Auswahl folgendermaßen vorgenommen werden:

Google Docs bietet einen gut funktionierenden Online-Editor für Office-Dokumente, an dem mehrere Bediener gleichzeitig arbeiten können. Problematisch ist der Im-/Export von Dokumenten insbesondere in Bezug auf die Formatierung. Zudem müssen alle Projektmitglieder für jede Datei, die sie lesen oder bearbeiten können sollen, explizit über einen erforderlichen Google-Account für diese berechtigt werden. Alternativ hätte die Datei für anonymes lesen und bearbeiten freigegeben und der entsprechende Dokumentenlink verteilt werden müssen. Ein Anonymes bearbeiten ist aus Datenschutzgründen und Schutz vor Vandalismus nicht gewünscht gewesen. Das Verteilen von Dokumentenlinks erschwert die Übersicht und das Handling der zahlreichen Dokumente und Artefakte.

Ein Dropbox-Verzeichnis bietet hinsichtlich des Hinzufügen und Ändern von Dateien den größten Komfort. Es gibt keine Einschränkung bezüglich des Dateiformats und die Synchronisierung mit dem Online-Speicher erfolgt unmittelbar mit dem Speichern der Datei im lokalen Dateisystem. Dies stellt sich bei der Bearbeitung von Quellcode, bei dem mehrere Dateien nur parallel (z.B. eine fertige Komponente) eingestellt werden sollen oder Änderungen verworfen werden sollen, als Nachteil heraus.

[1] Siehe https://github.com.
[2] Siehe http://docs.google.com/?hl=de.
[3] Siehe https://www.dropbox.com.

GitHub ist ein Webhoster für ein originäres Repository für Softwareentwicklungsprojekte. Die Verwaltung beliebiger Dateien erfolgt in einem so genannten Branch. Neue Dateien müssen manuell hinzugefügt werden und Änderungen manuell bestätigt werden. Als lokales GUI-basiertes Synchronisationswerkzeug wird zur Unterstützung TortoiseGit[4] verwendet. Damit können Änderungskonflikte in Text-Dateien leicht zusammengeführt werden. Sofern innerhalb der Textdatei die Änderungen in verschiedenen Bereichen vorgenommen wurden, erfolgt eine automatische Zusammenführung. Nur bei Konflikten in gleichen Bereichen in der Textdatei muss ein manueller Eingriff erfolgen, die Änderungen werden dabei von TortoiseGit besonders hervorgehoben. Bei binären Dateien steht eine solche Funktion nicht zur Verfügung. Durch das Repository wird in diesem Fall die Synchronisierung der Änderung in das Online-Verzeichnis unterbunden. Dieser Konflikt muss durch einen manuellen Eingriff lokal aufgelöst werden.

Für die Verwaltung der Arbeitsergebnisse wurde das Tool GitHub ausgewählt. Dieses Tool kombiniert die Unterstützung beliebiger Dateiformate mit der vollen Benutzerkontrolle bei Synchronisationsvorgängen. Die Dropbox wurde aufgrund der eingeschränkten Nutzbarkeit für die Speicherung von Quellcode (gezieltes Publizieren eines Quellcode-Stand vs. automatischer Upload nach dem Speichern) abgelehnt. Bei paralleler Bearbeitung von Quellcode hätten durch versehentliche Speichervorgänge unnötig viele Synchronisierungskonflikte auftreten können. Google Docs wurde auf Grund der nur eingeschränkten Unterstützung von wenigen Dateitypen ausgeschlossen.

2.5.2 Modellierungswerkzeug

Um UML-Modellierungs- und Quellcodeartefakte synchron zu entwickeln und homogen zu speichern, wurde zu Beginn des Projektes ein einheitliches Modellierungswerkzeug ausgewählt. Mit diesem Werkzeug sollte ein automatisierter Abgleich zwischen den Artefakten stattfinden können, um zum einen Fehler schneller erkennen zu können und zum anderen die Arbeit zu vereinfachen. Als UML-Modellierungswerkzeug wurde die kostenfreie Open-Source-Software Modelio aufgrund der guten Integration in das ausgewählte Softwareentwicklungswerkzeug und der intuitiven Bedienung gewählt. Der ausführliche Auswahlprozess

[4] Siehe http://code.google.com/p/tortoisegit/.

ist in dem angehängten Dokument „Auswahl des UML-Modellierungswerkzeug“[5] beschrieben.

Die Oberflächenmodellierung sollte zunächst möglichst einfach erfolgen, so dass zu diesem Zweck Microsoft PowerPoint[6] verwendet werden soll. Dadurch soll die Einarbeitung in ein spezielles Mock-up- oder Oberflächenmodellierungs-Werkzeug eingespart werden.

2.5.3 Softwareentwicklungswerkzeug

Die Entwicklung der Software soll in Java erfolgen, als IDE soll Eclipse[7] verwendet werden. Dies ist die einzige Plattform/Entwicklungsumgebung, bei der im Projektteam bereits Vorwissen vorhanden war. Weiterhin wurde auch die Roo Shell benutzt. Diese kann sowohl nativ als eigenständige Kommandoshell als auch per Plug-In in die Eclipse-IDE importiert werden.[8]

2.6 Vorgehensmodell

Für das Softwareprojekt sollte ein weit verbreitetes und etabliertes Vorgehensmodell verwendet werden. Bei der Auswahl möglicher Vorgehensmodelle wurde auf eine Auswahl etablierter Modelle aus dem Studium zurückgegriffen: Wasserfallmodell, V-Modell, Scrum oder der Rational Unified Process (RUP).

Das sequenzielle Wasserfallmodell sollte in diesem Projekt nicht verwenden. Aufgrund des sehr strikten Projektablaufs fehlt die notwendige Flexibilität, zumal die Anforderungen und der Umfang zu Beginn des Projekts noch nicht vollständig bekannt waren. Das Wasserfallmodell ist besser für Projekte mit klar definierten Anforderungen verwendbar, weil ein Rücksprung in eine vorherige Phase nicht bzw. nur eingeschränkt möglich ist.[9]

Aufgrund der klaren Strukturen, der dadurch guten Planbarkeit und der umfangreichen Spezifikation der Artefakte ist das V-Modell bzw. V-Modell XT als Vorgehensmodell für dieses Projekt besser geeignet. Jedoch erfolgen bei diesem flexiblen Vorgehensmodell intensive Zusammenarbeit und Abstimmungen mit dem - im Projektseminar nicht existierenden - Auf-

[5] Siehe Artefakte auf CD „Auswahl Modellierungswerkzeug.pdf“.
[6] Siehe http://office.microsoft.com/de-de/powerpoint/.
[7] Siehe www.eclipse.org/.
[8] Vgl. o.V. (2012).
[9] Vgl. Schatten, A.; Biffl, S. et al., S. 48f.

traggeber. Ebenso fehlte im Projekt die entsprechende Ausschreibung und Anforderungsspezifikation durch den Auftraggeber, welches häufig bei dem Vorgehen nach dem V-Modell XT bereits vorliegt bzw. in der Analysephase berücksichtigt wird.[10] Aus diesen beiden Gründen wurde sich ebenfalls vom V-Modell bzw. V-Modell XT als Vorgehensmodell für dieses Projekt distanziert.

Ein agiles Vorgehensmodell erschien als deutlich geeigneter. Es wurden die beiden Vorgehensmodelle Scrum und RUP näher betrachtet:

Ein Vorgehen nach Scrum ist äußert flexibel auf Änderungen von Anforderungen durch entsprechende Dokumentation und Priorisierung im Produkt-Backlog. Ebenfalls ist Scrum sehr gut für kleine Entwicklungsteams geeignet und durch die kurzen Sprints können sehr schnell erste Ergebnisse vorgestellt werden.[11] Nachteil dieses Vorgehensmodell und damit Ausschlusskriterium für die Verwendung in diesem Projekt ist, dass de facto ein Projektmitglied vollständig als Scrum-Master ausgelastet ist und die Planung und Koordination übernimmt.[12] Ferner war es während der Sprints nicht möglich die täglichen Statusmeeting (Daily Scrums) zeitlich einzurichten. Scrum benötigt ein kleines Projektteam was nahezu 100% der zur Verfügung stehenden Zeit für das Projekt aufwenden kann. Abweichungen eines reinen Vorgehens nach Scrum sind dabei verpönt und werden als Scrumbut verschmäht.[13]

Der Rational Unified Process ist in Prozessphasen gegliedert, die iterativ durchlaufen werden. Aufeinanderfolgende Phasen können parallel bearbeitet werden. Darüber hinaus wird ein Großteil der Spezifikationen noch vor der Umsetzung vorgenommen. Für jede Projektphase sind bestimmte Artefakte definiert, in denen die Arbeitsergebnisse festgehalten werden. Bei diesem agilen Vorgehen erfolgen die Spezifikation und die Umsetzung inkrementell. Dies erfordert viel Erfahrung der Projektmitglieder im Bereich der Teamarbeit miteinander.[14] RUP erschien aus dieser Sicht ideal, da durch dieses Vorgehensmodell alle Projektmitglieder den gesamten Softwareentwicklungsprozess erfahren können. Gleichzeitig bietet RUP die notwendige Flexibilität auf Änderungen der Anforderungen zu reagieren. Fer-

[10] Vgl. Schatten, A.; Biffl, S. et al., S. 68.
[11] Vgl. Schatten, A.; Biffl, S. et al., S. 64f.
[12] Dieses Aufgabenteilung sollte in diesem Projekt vermeiden werden, siehe Kap. 2.3 „Entscheidungskompetenz“, S. 3.
[13] Vgl. Schwaber, K. (2012).
[14] Vgl. Schatten, A.; Biffl, S. et al., S.58ff.

ner erschien der hohe Formalisierungsgrad für ein Studienprojekt, bei dem die Arbeitsergebnisse dokumentiert werden sollen, ideal.

2.7 Projektplanung

Wie in Kapitel 2.3 „Entscheidungskompetenz“, Seite 3, bereits erwähnt, wurde zu Beginn festgelegt, dass es innerhalb dieses Projektseminars keinen Projektleiter geben sollte bzw. dass diese Rolle nicht durch ein einzelnes Projektmitglied besetzt werden sollte. Die zugehörigen unterstützenden Aktivitäten und Aufgaben sollten durch ein Teammitglied zusätzlich zu den „normalen” Projektaufgaben bzw. Softwareentwicklungsaktivitäten bearbeitet und mit erledigt werden.

Die grobe Projektplanung sollte dabei in Microsoft Projekt[15] erfolgen und das Vorgehensmodell des Rational Unified Process berücksichtigen. Zur Arbeitserleichterung sollte die Feinplanung, d.h. die Steuerung und Dokumentation aller Aufgaben bzw. Aufgabenpakete, inklusive Bearbeiter und Erledigungszeitpunkt in einer auf Microsoft Excel[16] basierende Aufgabenliste erfolgen.

2.8 RUP-Artefakte

Zu Beginn des Projektes wurde die für den Rational Unified Process relevanten Artefakte identifiziert und hinsichtlich der Umsetzung innerhalb des Projektseminars bewertet.[17]

2.8.1 Visions-Dokument

Das Visions-Dokument soll zum einen das Resultat der Analyse des Problemfeldes sein und zum anderen die Anforderungen der Stakeholder, Kunden oder Endanwender enthalten. Da diese beiden Aspekte sowohl durch das Projekt-Exposé, als auch durch das Lastenheft abgedeckt werden, wurde im Team entschieden, dieses Artefakt nicht zu erstellen bzw. nicht weiter zu berücksichtigen.

[15] Siehe http://www.microsoft.com/project/de/de/default.aspx.
[16] Siehe http://office.microsoft.com/de-de/excel/.
[17] Siehe dazu auch Besprechungsprotokoll vom 21.03.2012, Top 3.

2.8.2 Risikoanalyse

Die Risikoanalyse wird zur Identifikation und Bewertung der Projektrisiken eingesetzt, um präventiv Maßnahmen zur Risikovermeidung ergreifen oder das Risiko steuern zu können. Ein Risiko wird dabei nach der Eintrittswahrscheinlichkeit und der Auswirkung auf das Projekt, in Bezug auf die Faktoren Zeit, Ressourcen und Inhalt/Umfang bewertet. Um diesem auch sehr praxisrelevanten und -nahen Aspekt zu berücksichtigen, einigte man sich darauf, diese Artefakte zu erstellen, um die identifizierten Risiken, deren Auswirkungen sowie die möglichen präventiven Maßnahmen auch für die Projektplanung zu berücksichtigen.

2.8.3 Grobe Projektplanung

Einstimmig wurde im Team entschieden, dass das die grobe Projektplanung als Artefakt unverzichtbar ist und daher erstellt werden soll. Die grobe sowie auch feine Projektplanung ist separat in den Kapiteln 2.7 „Projektplanung“, Seite 8, und 3.7 „Umsetzung der Projektplanung“, Seite 21, beschrieben, daher wird an dieser Stelle nicht weiter auf diese eingegangen.

2.8.4 Kosten-/Nutzenabschätzung

Ohne eine Kosten-/Nutzenabschätzung ist es fast unmöglich Finanzinvestoren zur Unterstützung des Projekts mit Venture Capital zu gewinnen.[18] Aus diesem Grund und zur Überprüfung, ob und unter welchen Rahmenbedingungen ein wirtschaftlicher Betrieb von der Plattform Eventalizer möglich ist bzw. gewesen wäre, wurde im Team festgelegt, dass eine Kosten-/Nutzenabschätzung erfolgen soll.

2.8.5 Anwendungsfalldiagramm

Das Anwendungsfalldiagramm, welches auch als Use Case Modell bezeichnet wird, sollte, einheitlich beschlossen, definitiv umgesetzt werden. Es wurde aber nicht als eigenes Artefakt erstellt, sondern in das Lastenheft unter dem Kapitel “Funktionale Anforderungen”[19] integriert.

[18] Vgl. Kaack, J. (2012).
[19] Siehe Artefakte auf CD „LH_Evantalizer.pdf“, Kap. 2.2 „Anwendungsfalldiagramm“, S. 9.

2.8.6 Definition des Projektziels und Abgrenzung
Überblick über Problembereich und Anforderungen
Stakeholder-Übersicht

Für die für den Unified Process relevanten Artefakte, die zum einen das Projektziel und die Abgrenzung definieren oder zum anderen einen Überblick über den Problembereich und die Anforderungen bzw. die Stakeholder wurde im Projektteam entschieden, dass diese auch umgesetzt werden sollten. Die Artefakte sollten dabei aber, genauso wie das Anwendungsfalldiagramm, mit im Lastenheft umgesetzt werden. Die Artefakte sind daher im Lastenheft unter dem Kapitel "Zielbestimmung und Zielgruppen"[20] zu finden.

2.8.7 Szenarien

Um im Projektteam Unklarheiten bezüglich des Leistungsumfanges des Projektes bzw. der Funktionalitäten der Internetplattform Eventalizer zu klären, wurde zu Beginn des Projektes entschieden, beispielhaft ein paar Szenarien als Artefakte zu erstellen. Szenarien beschreiben in der Softwareentwicklung konkrete Beispiele für Interaktionen zwischen einem Benutzer und dem System. Oft werden die Benutzer dabei als Personae modelliert.[21] Die innerhalb dieses Projektseminars umgesetzten Szenarien sollten dabei auch in Textform, d.h. als User-Stories repräsentiert werden.

2.8.8 Überblick über die zu erbringenden Leistungen

Dieses Artefakt wurde einstimmig als für das Projektseminar relevant angesehen, wurde aber nicht als separates Artefakt umgesetzt, sondern im Pflichtenheft integriert. Im Pflichtenheft[22] sind die zu erbringenden Leistungen beschrieben.

2.8.9 Begriffslexikon/Glossar

Das Begriffslexikon bzw. das Glossar enthält die Definitionen von Begriffen, die für das Projektseminar aus fachlicher oder technischer Sicht relevant sind. Die Begriffe werden so definiert, dass sie ein einheitliches und gemeinsames Verständnis innerhalb des Projektteam

[20] Siehe Artefakte auf CD „LH_Evantalizer.pdf", Kap. 1 „Zielbestimmung und Zielgruppen", S. 4ff.
[21] Vgl. Cooper, A.; Reimann, R. et al. (2010), S. 131.
[22] Siehe Artefakte auf CD „PH_Evantalizer.pdf", Kap. 2 „Funktionale Anforderungen", Seite 4ff und Kap. 3 Qualitätsanforderungen, S. 18ff.

bzw. darüber hinaus auch für Stakeholder, etc. garantieren. Somit werden unterschiedliche subjektive Interpretationen der projektspezifischen Begriffe verhindert. Außerdem kann das Begriffslexikon bzw. das Glossar als Grundlage für die Benutzerdokumentation und die Onlinehilfe verwendet werden.

2.8.10 LASTENHEFT

Als eines der zentralen Artefakte der Anforderungsspezifikation sollte das Lastenheft, auch Requirement Specification genannt, für dieses Projektseminar umgesetzt werden. Das Lastenheft sollte aus Sicht des Anwenders bzw. Auftraggebers, die für die Internetplattform Eventalizer wünschenswerte und relevante Anforderungen beschreiben.

2.8.11 PFLICHTENHEFT

Die vorher in Lastenheft definierten Anforderungen des Auftraggebers werden aus Sicht des Auftragnehmers im Pflichtenheft bzw. in der Software Requirement Specification bezüglich der Art und Weise, wie diese Anforderungen umgesetzt und realisiert werden sollen, näher beschrieben. Das Pflichtenheft ist damit eines der zentralen Artefakte der Softwarespezifikation und sollte, einheitlich beschlossen, auch für dieses Projektseminar umgesetzt werden.

2.8.12 ANWENDUNGSARCHITEKTUR

Das Artefakt für die Anwendungsarchitektur wird auch als Software Architecture Document bezeichnet und beschreibt die grundlegenden Komponenten und deren Zusammenspiel innerhalb des Systems, d.h. der Internetplattform Eventalizer. Dieses Dokument sollte auch für das Projektseminar umgesetzt werden, da es einen umfassenden Überblick über die Architektur, d.h. die Grobgliederung der Komponenten der gesamten Internetplattform, vermitteln.

2.8.13 DESIGNMODELL

Das Design Modell enthält die systemrelevanten Komponenten, Schnittstellen, wichtige Klassen und das Datenmodell. Da diese Aspekte bereits in den Artefakten “Pflichtenheft”[23] und “Anwendungsarchitektur” beschrieben werden, wurde entschieden, dieses Artefakt nicht separat zu erstellen und umzusetzen.

[23] Siehe Artefakte auf CD „PH_Evantalizer.pdf“ und „Anwendungsarchitektur.pdf“.

3 PROJEKTDURCHFÜHRUNG

In diesem Kapitel erfolgt die Umsetzung, Analyse und Reflexion der Entscheidungen aus der Projektinitialisierung. Es werden die wesentlichen Eckpunkte der Umsetzung der einzelnen Artefakte und Entscheidungen beschrieben. Bedeutende Änderungen im Projektverlauf werden aufgezeigt, erläutert und bewertet, sowie zukünftige Verbesserungsvorschläge dargestellt. Die Gliederung orientiert sich an der Struktur und dem Aufbau des vorherigen Kapitels.

3.1 UMSETZUNG DER PROJEKTIDEE

Da für dieses Projekt keine Ausschreibung eines Auftraggebers existierte, auf der sich der potenzielle Anbieter per Abgabe eines Angebotes bewerben könnte, musste im Projektteam die Projektidee eigenständig entwickelt werden. Die Basis hierfür bildete die im Kapitel 2.1 „Projektidee“, Seite 2, beschriebene, gemeinschaftlich per Brainstorming entwickelte, sehr rudimentäre Vorstellung über das Projekt, welche als Mind-Map zusammengefasst wurde:

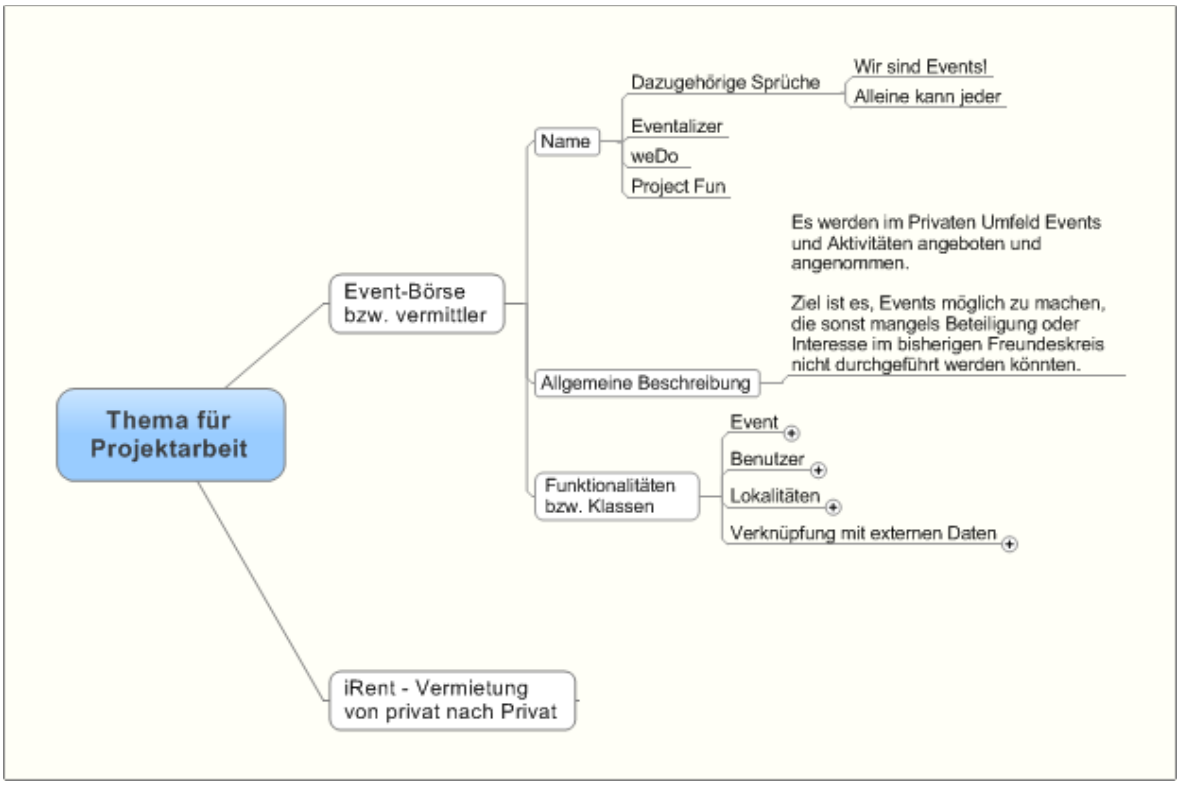

ABBILDUNG 1 BRAINSTORMING: PROJEKTIDEE[24]

[24] Eigene Darstellung.

Diese auf Basis einer Mind-Map entwickelte Vorstellung ließ noch genügend Interpretationsspielraum in vielen Themenbereichen zu und musste erst einmal in wichtigen, für die weitere Entwicklung notwendigen Bereichen konkretisiert werden. Hierfür wurde das Projekt-Exposé[25] geschrieben. Dies beschreibt neben der Realisierungsplattform und dem Vorgehensmodell auch das Projektthema, in welchem auch die Mehrwerte der Internetplattform aufgeführt sind.

3.2 Umsetzung der Aufgabenverteilung und -durchführung

Im Rahmen der Aufgabenverteilung wurde entschieden, dass das Projekt nach dem Motto "Jeder macht alles" durchgeführt wird und somit für jeden Aufgaben im Bereich der Anforderungsermittlung, der Softwarespezifizierung, der Architekturkonzeption, des Entwurfs, der Implementierung und der Qualitätssicherung erledigt wird. Damit sollte jedes Projektmitglied das Maximale aus dem Projektseminar mitnehmen. Dies wurde bewusst - auch unter der Berücksichtigung von einem erhöhtem Zeitaufwand und stressigen Situationen - so entschieden und im weiteren Projektverlauf bei der Verteilung und Durchführung von Aufgaben versucht zu berücksichtigen.

Nichtdestotrotz bildeten sich Aufgabenbereiche heraus, welche primär durch bestimmte Personen bearbeitet wurden. Dies lag zum einen daran, dass einige Personen bei diesen Aufgaben über ein höheres Know-how verfügten oder diese Aufgabe nicht als Team durchführbar waren.

Im Folgenden werden einige Aufgabenbereiche bzw. unterstützende Aktivitäten der Softwareentwicklung aufgeführt, welche primär durch eine oder mehrere Personen zusätzlich zu den „normalen" Projektaufgaben bzw. Softwareentwicklungsaktivitäten bearbeitet und erledigt wurden:

- Projektmanagement und Planung, inkl. Aufgabenverteilung und Aufgabenkontrolle bzw. Aufgabenüberwachung (Alexander Benölken)
- Auswahl eines Repository's und Versionierungstools (Matthias Beer/ Felix Wessel)
- Auswahl eines Modellierungstools (Felix Schulze Mönking)
- Protokollführung und Template-Erstellung (Martin Garrels)

[25] Siehe Artefakte auf CD „Projekt-Expose.pdf".

Ein Vorteil der Entscheidung, dass jedes Teammitglied möglichst jeden Bereich der Softwareentwicklung bearbeitet, war es, das Know-how in die Breite gestreut wurde. Jedes Teammitglied lernte so die sechs grundlegenden Aktivitätsbereiche der Softwareentwicklung (Anforderungsermittlung, Softwarespezifizierung, Architekturkonzeption, Entwurfs, Implementierung und Qualitätssicherung) kennen und vertiefte in diesen Bereichen das Wissen und die Anwendungskompetenz. Somit wurde das Ziel, dass jeder das Maximale aus dem Projektseminar mitnehmen sollte, erreicht. Als nachteilig kann der persönliche, erhöhte Zeitaufwand für das Projektseminar angesehen werden. Aufgrund der breit angelegten Aufgabenbearbeitung musste sich jedes Teammitglied in jeden Aktivitätsbereich der Softwareentwicklung einarbeiten und zurecht finden, um dort auch die quantitativ und qualitativ geforderten Ergebnisse liefern zu können. Darüber hinaus mussten die oben aufgeführten Aufgabenbereiche noch durch einige Teammitglieder parallel erledigt werden.

3.3 Umsetzung der Entscheidungskompetenz

Bei Beginn des Projektes wurde festgelegt, dass wichtige und relevante, den weiteren Projektverlauf betreffende Entscheidungen durch das gesamte Team mehrheitlich entschieden werden sollen. Diese Entscheidung hatte die Auswirkung, dass in mehreren Bereichen viele Sachverhalte oft gemeinsam in kontroversen Diskussionen erörtert wurden, um dann gemeinschaftlich eine Entscheidung zu fällen. Diese teamorientierte Entscheidungsfindung wurde im weiteren Projektverlauf durch einen erhöhten Zeitbedarf bei Treffen und Jour fixes “erkauft”. Im Nachhinein wäre es gegebenenfalls besser gewesen die Entscheidungskompetenz in einigen Bereichen auf ein oder zwei Teammitglieder zu verteilen, um so schlankere und schnellere Entscheidungen fällen zu können. Trotzdem hätten grundsätzliche Entscheidungen nur im kompletten Team getroffen werden können, so dass es für einige Entscheidungen einen erhöhten Abstimmungsbedarf gegeben hätte. Dies wäre entsprechend im Projektplan mit zu berücksichtigen. Für besonders kontrovers diskutierte Themen, bei denen eine Mehrheitsfindung innerhalb der Projektgruppe schwierig war, wäre eine Schiedsstelle bspw. in Form des Projektleiters hilfreich gewesen um zeitnah eine verbindliche Entscheidung zu treffen.

3.4 Umsetzung der Jour fixes

Die Jour fixe wurden mindestens einmal in der Woche durchgeführt, bei Bedarf auch mit einer höheren Frequenz. Insbesondere zu wichtigen Meilensteinen wurde der Freitag als zweiter Jour fixe hinzugenommen. An den Jour fixen war im Normalfall die gesamte Projektgruppe anwesend, so dass Entscheidungen auf breiter Basis getroffen wurden. In diesem Zusammenhang wurde auch der aktuelle Arbeitsstand präsentiert und die weitere Vorgehensweise diskutiert. Als Steuerungselement wurde eine auf Microsoft Excel basierende Aufgabenliste verwendet. Dabei bestand eine Herausforderung in der Abgrenzung der Inhalte der Aufgabenliste und des Protokolls. Die Entscheidungen wurden jeweils ergebnisorientiert im Protokoll vermerkt, daraus resultierende Aktivitäten wurden in die Aufgabenliste aufgenommen. Der Status der jeweiligen Aufgabe wurde innerhalb der Aufgabenliste erfasst, so dass nur wichtige Statusinformationen im Protokoll aufgenommen wurden. Die jeweils aktuelle Aufgabenliste wurde nach dem Jour fixe als Anhang dem Protokoll hinzugefügt.

3.5 Umsetzung der Infrastruktur

Im Rahmen der Projektdurchführung gab es sowohl im Bereich der Dateiaustauschplattform als auch bei den Modellierungswerkzeugen Änderungen. Die Erfahrung mit den in der Projektinitialisierung entschiedenen Tools GitHub, Modelio und Microsoft PowerPoint werden in den nachfolgenden Kapiteln erläutert und die jeweiligen Toolwechsel begründet.

3.5.1 Dateiaustauschplattform

Alle Arbeitsergebnisse wurden einheitlich im zuvor ausgewählten Repository GitHub gespeichert. Die Möglichkeit, Änderungen kontrolliert freigeben zu können und zusätzlich mit einem Kommentar Änderungen an mehreren Dateien zu erläutern, hat sich bewährt. Die Flexibilität, die ein wesentlicher Bestandteil der Benutzung von GitHub ist, hat aber auch gleichzeitig den Arbeitsaufwand erhöht indem es zu Synchronisierungsproblemen gekommen ist. Ursache hierfür ist am häufigsten gewesen, dass zwei Projektmitglieder gleichzeitig eine Version einer Datei bearbeitet haben. Nur der schnellere von beiden konnte seine Änderungen in das zentrale Repository synchronisieren. Der später Synchronisierende kann mit seinen Änderungen nicht die des ersten Benutzers überschreiben, der Upload seiner Datei wird abgelehnt. An dieser Stelle können die Änderungen üblicherweise zusammengeführt werden, um die Änderungen beider Projektmitglieder zu erhalten. Genau hier hat sich eine

Schwäche in GitHub aufgezeigt: Es können nur Text-Dateien zusammengeführt werden, nicht aber binäre Dateien, wie zum Beispiel Microsoft Office Dokumente oder auch Modellierungsartefakte. Der Anschließende Prozess zum manuellen Zusammenführen beider Änderungen ist sehr aufwendig: Die eigenen Änderungen müssen unter einem anderen Dateinamen außerhalb des Repositories gespeichert werden, die Änderungen des ersten Synchronisierenden müssen heruntergeladen werden und die eigene Arbeit muss nun manuell übertragen werden. Bei der Konfliktbereinigung kam es vereinzelnd zu manuellen Fehlbedienungen: Sofern bei einem potentiellen Konflikt vor der Bestätigung der eigenen lokalen Änderungen (commit) die aktuellen Daten vom Server geladen wurden (pull), wurde von TortoiseGit die Daten lokal überschrieben, so dass ein Verlust der eigenen Änderungen erfolgte.

Dieser Umgang mit Synchronisationsproblemen stellt sich bei Nutzung von Dropbox einfacher dar: Unmittelbar mit dem Speichern der Änderungen eines Benutzers im lokalen Dateisystem (hier ist die Funktion “automatische Speichern” der Microsoft Office Produkte ausreichend) synchronisiert Dropbox die Datei. Somit wird die Wahrscheinlichkeit minimiert, dass ein anderer Benutzer noch eine alte Dateiversion zur Bearbeitung öffnet. Kommt es dennoch zu einer mehrfachen Bearbeitung einzelner Dateien durch mehrere Benutzer, wird die Dateiversion eines jeden Benutzers in der Dropbox gespeichert, versioniert und als Konflikt mit unterschiedlichen Dateinamen gekennzeichnet. Ein Datenverlust kann so in jedem Fall verhindert werden. Nach einer kurzen Absprache mit dem jeweils anderen Bearbeiter kann nun ohne weiteren Aufwand das manuelle Übertragen der Änderungen in die aktualisierte Datei erfolgen.

Eine einfachere Bedienung der Dateiaustauschplattform wäre gerade für binäre Dateiformate wünschenswert gewesen, bei diesen konnten die Stärken von GitHub nicht genutzt werden.

Die Erfahrungen aus der Erstellung der ersten Artefakte mit größeren Dokumenten (Lastenheft, Pflichtenheft, Abschlusspräsentation), haben dazu geführt, dass der Reflexionsbericht in Google Docs erstellt wurde. Das Vandalismus-Risiko durch Dritte, die in Kenntnis des Links gelangten, wurde dabei akzeptiert. Google Docs hat sich für die gemeinsame Bearbeitung eines Dokuments als sehr positiv herausgestellt. Jedoch zeigt Google Docs deutliche Schwä-

chen in der Bearbeitung von komplexen Dokumenten: Eine saubere Formatierung und Strukturierung eines Google Docs Dokuments nach den Richtlinien für wissenschaftliches Arbeiten ist nicht möglich, insbesondere in Bezug auf die Erstellung die Generierung des Inhaltsverzeichnis, der Seitenformatierung, der Seitenzahlen und die Nutzung von Kopf-/Fußzeile sowie der Literaturzitate. Daher wurde die Grundstruktur des Reflexionsberichts zunächst mit Microsoft Word erstellt und anschließend zur Füllung der Kapitel in Google Docs übertragen. Das inhaltlich finale Dokument wurde anschließend wieder aus Google Docs nach Microsoft Word exportiert und in Microsoft Word entsprechend der Richtlinien für Wissenschaftliches Arbeiten überarbeitet, was insbesondere die zuvor genannten Schwächen betrifft.

3.5.2 Modellierungswerkzeug

Während der Projektinitialisierung wurde das Modellierungswerkzeug modelio ausgewählt.[26] In der anschließenden Projektdurchführung hat sich herausgestellt, dass sich der Austausch der Arbeitsergebnisse über ein Repository nur mit Hindernissen bewerkstelligen lässt. Es muss ein kompletter eclipse-Workspace ausgetauscht werden, gemeinsames Arbeiten ist auch auf Grund der zuvor beschriebenen Synchronisationsprobleme kaum möglich.

Darüber hinaus wurden die Stärken von modelio durch die Auswahl einer Anwendungsarchitektur, die selbstständig den Abgleich von Modell und Quellcode vornimmt, nicht mehr benötigt. Nun konnte zur Erzeugung der UML-Diagramme ein reines Modellierungswerkzeug verwendet werden. Die Modellierung von UML-Artefakten wurde nun auf Grund der im Rahmen der MSDN Academic Alliance[27] kostenfrei zur Verfügung gestellten Lizenz und des hohen Funktionsumfangs mit Microsoft Visio[28] vorgenommen.

Für die Oberflächenmodellierung konnte zunächst schnell und einfach mit Hilfe von Microsoft PowerPoint Entwürfe erstellt werden. Nach der Rückmeldung unserer Peer-Review-Gruppe, in der korrekterweise bemängelt wurde, dass die Mock-ups nicht realitätsnah sind. Unter anderem wurde angemerkt, dass es sich bei Eventalizer um eine über den Browser aufrufbare Internetplattform handelt, was anhand der Mock-ups nicht zu erkennen war.

[26] Siehe Artefakte auf CD „Auswahl Modellierungswerkzeug.pdf“.
[27] Siehe http://www.microsoft.com/germany/msdn/academic/dreamspark/default.aspx.
[28] Siehe http://office.microsoft.com/de-de/visio/.

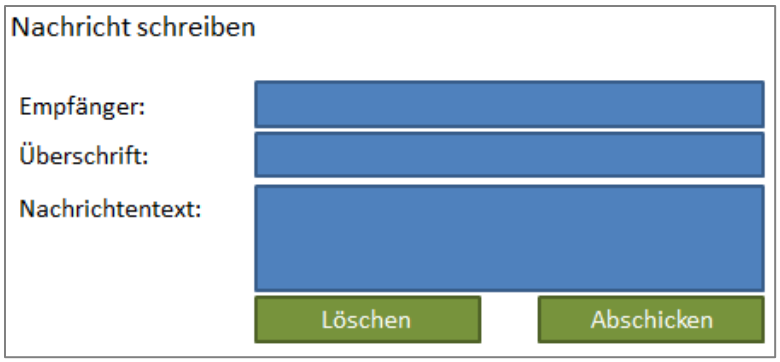

ABBILDUNG 2 MOCK-UP - NACHRICHT SCHREIBEN (POWERPOINT)[29]

Durch einen Wechsel des Werkzeugs für die Erstellung der Oberflächen-Mock-ups von Microsoft PowerPoint auf die Web-Demo-Version von Balsamiq Mock-ups[30] konnte die Hinweise aus dem Peer-Review umgesetzt werden.

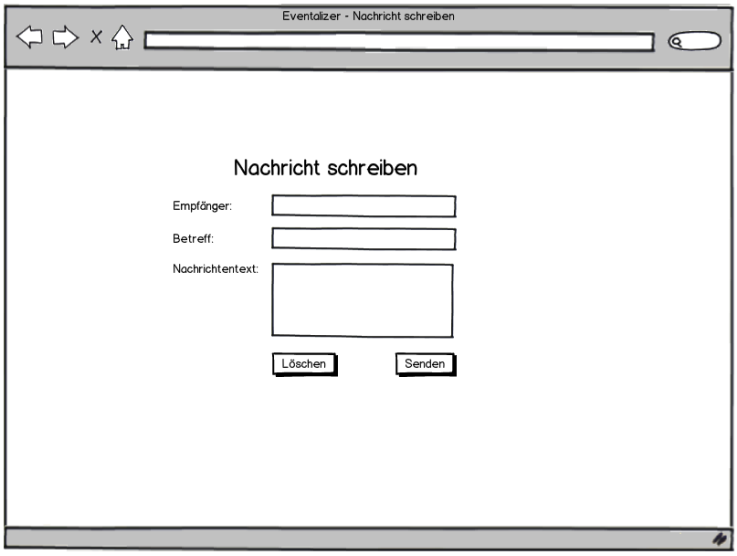

ABBILDUNG 3 MOCK-UP - NACHRICHT SCHREIBEN (BALSAMIQ MOCK-UPS)[31]

Dieses Werkzeug überzeugt durch Funktionsvielfalt und einer deutlich realitätsnäheren Darstellung der Oberflächen. Die Mock-ups wurden zur Erstellung des Domänenklassenmodells herangezogen, jedoch nicht bei der Erstellung von implementierten Oberflächen herangezogen. Hier wurden die von Spring Roo[32] generierten JSPs und CSS-Dateien unangepasst verwendet, da sie die Funktionalität des Prototyps in keiner Weise schmälern.

[29] Eigene Darstellung.
[30] Siehe http://builds.balsamiq.com/b/mockups-web-demo/.
[31] Eigene Darstellung.
[32] Siehe http://www.springsource.org/spring-roo.

3.5.3 SOFTWAREENTWICKLUNGSWERKZEUG

Zur Umsetzung des Projektes wurde das Java Framework „Spring Roo“ verwendet. Spring Roo wurde ausgewählt wegen seiner Merkmale als Rapid-Development-Framework. Die Implementierungszeit sollte so verkürzt werden, um mehr Zeit für das Design der Anwendung verwenden zu können.[33] Spring Roo ist eher eine Sammlung von Generatoren und Prozessen des eigentlichen Spring Frameworks. Daher ist Spring Roo selbst zur Laufzeit nicht im Code eingebunden. Es unterstützt lediglich bei der Generierung.[34] Spring Roo ermöglicht es mit wenigen Befehlen oder Strukturen für das Spring Framework eine Standard-Webanwendung zu erstellen. Vorteilhaft ist das Spring Roo dabei sehr viele Annahmen bereits trifft und sich dadurch der initiale Aufwand ungemein reduziert. So kann man mit jeweils einem Tastaturkommando eine Weboberfläche erzeugen, automatische Roundtrip-Tests für alle Domänenklassen erstellen oder Anmelde- und Security-Mechanismen einbinden.

Die Nachteile von Spring Roo sind das es eben bei diesem Prozess sehr viele Annahmen vom Framework getroffen und auch schon implementiert werden, die man bei weiterer Spezialisierung der Anwendung anpassen muss. Beispielweise unterstützt Spring Roo zwar einen generellen Login-Mechanismus. Wenn jedoch keine eigenen Passwörter verwaltet werden sollen, sondern Login-Mechanismen von Drittanbietern (z.B. Facebook, Twitter) nutzen möchte, sind erheblich Anpassungen notwendig.

Ein weiteres Risiko besteht darin, dass bereits vom Entwickler angepasste Ressourcen bei einer erneuten Generierung durch das Roo Framework überschrieben werden. Dieses Problem wird durch die Trennung von generiertem und vom Entwickler geschriebenem Code in AspectJ-Dateien und Java-Klassen zwar minimiert. Ganz ausschließen lassen sich etwaige Probleme damit jedoch nicht.

Spring Roo wurde dabei gewählt, obwohl noch keine großen Erfahrungen mit dem Framework bestanden. Dies erhöhte zwar das Risiko für das Gelingen des Projektes. Allerdings konnte man so auch im Zuge des Projektes Erfahrung mit „neueren“ Techniken, wie beispielsweise Apache Maven und AspectJ gewinnen, wozu in der beruflichen Praxis nicht immer die Möglichkeiten bestehen.

[33] Vgl. o.V. (2012b).
[34] Vgl. o.V. (2012c).

Spring Roo ist außerdem ein relatives junges Framework (Release der Version 1.0.0 im April 2010)[35], so dass Dokumentation noch im vergleichsweise geringeren Ausmaße zu finden ist.

3.6 Umsetzung des Vorgehensmodell

Innerhalb des Projektseminars wurde versucht nach dem Vorgehensmodell des Rational Unified Process vorzugehen. Es wurde versucht, die Prozessphasen iterativ zu durchlaufen und die für jede Phase relevanten Artefakte umzusetzen.

Bei der Umsetzung stellte sich heraus, dass die notwendige Erfahrung bezüglich der Durchführung anhand des Rational Unified Process nicht in dem Maße vorhanden war, wie es vorher eingeschätzt wurde. Die Bewertung und Festlegung auf die relevanten, umzusetzenden Artefakte, sowie deren Vollständigkeitsgrad gestaltete sich zu Anfangs schwierig.

Beispielsweise wurde sich zu Beginn intensiv und aufwendig den RUP Artefakten zur Kosten-/Nutzenabschätzung und Risikoanalyse beschäftigt. Vorteilhaft war hierbei, dass diese auch in der Praxis relevanten Artefakte exemplarisch im Projektseminar erstellt wurden. Im Nachhinein wurde aber erkannt, dass die damit verbrachte Zeit besser für die in der Softwareentwicklung wichtigeren Artefakte, besonders jene für die Anforderungsspezifikation und Softwarekonzeption, hätte investiert werden sollen. Als sehr positiv wurde dagegen die nach RUP bereits vor der Definition und Modellierung der Anwendungsfälle notwendige Erstellung der Szenarien und des Begriffslexikon/Glossars aufgenommen, da diese ein für die weitere Anforderungsspezifikation enorm wichtiges, gemeinsames Verständnis für bestimmte projektspezifische Begriffe schufen und damit die gleiche Wissensbasis unter den Teammitglieder bildeten.

Innerhalb des Projektseminars wurde sich aber nicht konsequent an den Unified Process gehalten bzw. dieser umgesetzt. Die Anforderungsspezifikation sowie die Softwarekonzeption erfolgten eher dem Wasserfall-Modell entsprechend. Die iterative Umsetzung anhand des Unified Process wurde hier nicht eingehalten. Vielmehr wurde zuerst die Anforderungsspezifikation versucht zu vervollständigen, um anschließen aufbauend auf das Lastenheft, im Rahmen der Softwarekonzeption das Pflichtenheft zu erstellen. Es wurden während der

[35] Vgl. Pronschinske, M. (2010).

Pflichtenhefterstellung zwar auch Änderungen am Lastenheft vorgenommen, trotzdem entsprach dieses Vorgehen eher einem Wasserfallmodell als einem Vorgehen nach dem Unified Process.

Auch die Implementierung eines Prototyps während der Anforderungsspezifikation bzw. der Softwarekonzeption wurde nicht verfolgt. Vielmehr wurde dies erst nach der Erstellung des Pflichtenheftes angegangen. Vorab wurde nur die Realisierbarkeit der Internetplattform über Spring Roo untersucht, gewisse Implementierungsansätze getroffen oder auch, als konkretes Beispiel, nur ein Prof of Concept für den OAuth Login gemacht. Da dieses Projektseminar aber nur eine prototypische Implementierung vorsieht, hatte dieses nicht RUP-konforme Vorgehen geringere Auswirkungen. In der Praxis würden das Fehlen eines Prototyps und der zu späte Implementierungsbeginn folgenreicher sein (Terminverzögerungen, Komplikationen, da bestimmte Funktionen nicht durch einen Prototypen validiert wurden und mit dem gewählten Framework nicht realisiert werden können).

3.7 Umsetzung der Projektplanung

Nach der Einarbeitung in den Rational Unified Process und dessen Phasen und Artefakten wurde auf den Rational Unified Process basierend die erste Grobplanung in Microsoft Projekt vorgenommen. Der erste Entwurf sah neben der Inception- jeweils zwei Iterationen für die Elaboration- und die Construction-, sowie abschließend die Transition-Phase vor. Wie die folgende Tabelle 1 zeigt war die Inception auf 18 Tage, die Elaboration auf insgesamt 20 Tage, die Construction auf 25 Tage und die Transition auf drei Tage geplant:

Vorgangsname	Dauer	Anfang	Fertig stellen
Projektstart	**5 Tage**	**Mo 05.03.12**	**Fr 09.03.12**
Konzeptualisierung / Inception	**18 Tage**	**Mo 12.03.12**	**Mi 04.04.12**
Entwurf / Elaboration: Iteration 1	**12 Tage**	**Do 05.04.12**	**Fr 20.04.12**
Entwurf / Elaboration: Iteration 2	**8 Tage**	**Mo 23.04.12**	**Mi 02.05.12**
Konstruktion / Construction: Iteration 1	**15 Tage**	**Do 03.05.12**	**Mi 23.05.12**
Konstruktion / Construction: Iteration 2	**10 Tage**	**Do 24.05.12**	**Mi 06.06.12**
Einführung und Inbetriebnahme / Transition	**3 Tage**	**Do 07.06.12**	**Mo 11.06.12**

Tabelle 1 Ursprünglicher Projektplan

Aufbauend auf dieser Grobplanung wurden die ersten Aufgaben und Aufgabenpakete identifiziert und in die auf Microsoft Excel basierende Aufgabenliste eingetragen. Außerdem wur-

den die Bearbeiter und die Erledigungszeitpunkt zu den Aufgaben bzw. Aufgabenpaketen hinzugefügt.

In dem ersten Teil der Projektseminars, d.h. während der Inception und der ersten Iteration der Elaboration wurden verstärkt die softwareentwicklungsspezifischen Aufgaben fokussiert. Eine Projektstatuskontrolle und -steuerung erfolgte nicht. Resultierend hieraus wurde nach dem ersten Teil des Projektseminars, genauer gesagt nach der Projektzwischenstandspräsentation, inklusive der vorherigen Zusammenstellung und Qualitätssicherung der Dokumente (Lastenheft, Anforderungsspezifikation, Architekturkonzeption) festgestellt, dass die laut Projektplanung vorgesehenen Ergebnisse noch nicht erreicht wurden und sich das Projekt somit außerhalb des Zeitplans befand.

Daraufhin wurden Projektsteuerungsmaßnahmen ergriffen: Der Funktionsumfang des zu implementierenden Prototypen wurde reduziert und der Fokus wurde verstärkt auf die korrekte und vollständige Spezifikation und Konzeption gelegt. Unterstützende Aktivitäten wurden auf ein Minimum reduziert, um rechtzeitig zur Abschlusspräsentation wenigstens eine funktionale Anforderungen vollständig umgesetzt vorführen zu können. Der Projektplan wurde demnach folgendermaßen angepasst:

Die zweite Iteration der Elaboration wurde von acht auf 15 Tage verlängert. Dies ging zu Lasten der ersten Iteration der Construction, welche von 15 auf neun Tage reduziert wurde. Außerdem wurde die zweite Iteration der Construction vollständig gestrichen und dafür ein Zeitraum von zehn Tagen für die Erstellung des Reflexionsberichtes, inklusive Finalisierung der restlichen Aufgaben der Spezifikation, Konzeption und Implementierung vorgesehen. Die folgende Tabelle 2 zeigt die geänderte Projektplanung:

Vorgangsname	Dauer	Anfang	Fertig stellen
Projektstart	**5 Tage**	**Mo 05.03.12**	**Fr 09.03.12**
Konzeptualisierung / Inception	**18 Tage**	**Mo 12.03.12**	**Mi 04.04.12**
Entwurf / Elaboration: Iteration 1	**12 Tage**	**Do 05.04.12**	**Fr 20.04.12**
Entwurf / Elaboration: Iteration 2	**15 Tage**	**Mo 23.04.12**	**Fr 11.05.12**
Konstruktion / Construction: Iteration 1	**9 Tage**	**Mo 14.05.12**	**Do 24.05.12**
Reflexionsbericht	**10 Tage**	**Do 24.05.12**	**Mi 06.06.12**
Einführung und Inbetriebnahme / Transition	**3 Tage**	**Do 07.06.12**	**Mo 11.06.12**

TABELLE 2 ÜBERARBEITETER PROJEKTPLAN

Resümierend kann man bei der Umsetzung der Projektplanung feststellen, dass es besser gewesen wäre, früher eine Kontrolle des aktuellen Projektstatus durchzuführen. Dann wäre es möglich gewesen, zeitnäher den Verzug im Projekt festzustellen, um daraufhin Steuerungsmaßnahmen zu ergreifen.

Trotz der verspäteten Realisierung des Projektverzuges kann aber positiv angemerkt werden, dass die Anpassung des Funktionsumfangs und damit der Aufgaben bzw. Aufgabenpakete schnell und problemlos erfolgte. Hierdurch war es möglich rechtzeitig zur Abschlusspräsentation zwei funktionale Anforderungen “Anmeldung” (LF20 im Lastenheft, F10 im Pflichtenheft) und “Erstellung eines Events” (LF150 im Lastenheft, F60 im Pflichtenheft) vorführen zu können.

3.8 Umsetzung der RUP-Artefakte

Im Folgenden wird auf die Umsetzung der RUP-Artefakte eingegangen. Es werden dabei nur die Artefakte beleuchtet und aufgeführt, für deren Umsetzung sich im Projektteam entschieden wurde.[36]

3.8.1 Risikoabschätzung

Im Rahmen des Projekts wurde eine kurze Risikoanalyse erstellt[37], die einen ersten Eindruck über die Projektrisiken geben sollte. Dabei wurden sechs Risiken identifiziert und in Bezug auf die Eintrittswahrscheinlichkeit und die Auswirkungen auf das Projekt (Zeit, Ressourcen und Inhalt) klassifiziert. Ein Maßnahmenkatalog für Aktivitäten, die beim Eintreten des Risikos durchgeführt werden, wurde ebenfalls erstellt. Aufgrund der Zielsetzung der Vorlesung, sowie aus zeitlichen Gründen wurden weitere Aktivitäten, die üblicherweise im Projektmanagement zur Risikoanalyse gehören, nicht fortgeführt bzw. nicht erstellt. Dazu gehören weitere Maßnahmen zur präventiven Vermeidung des Risikoeintritts, das regelmäßige Überprüfen der Risiken und Maßnahmen, sowie die ggf. notwendige Erfassung weiterer Risiken.

[36] Siehe Kap. 2.8 „RUP-Artefakte“, S. 8.
[37] Siehe Artefakte auf CD „Risikoanalyse.pdf“.

3.8.2 Kosten-/Nutzenabschätzung

Es wurden alle Kosten innerhalb des Projektes von der Initialisierung bis hin zur Stilllegung versucht abzuschätzen. Eine genaue Erläuterung der Wirtschaftlichkeitsbetrachtung befindet sich im Anhang.[38]

Im Rahmen der Recherche hat sich gezeigt, dass pro Nutzer nur geringe Einnahmen zu erwarten sind. Es wird daher, allein um die regelmäßigen Infrastrukturkosten bewältigen zu können, eine große Benutzerbasis benötigt. Zieht man zusätzlich Personalkosten hinzu, sind nach unserer Berechnung über Zehn Millionen Benutzer pro Monat notwendig, um den Eventalizer mit den unterstellten Einnahmequellen wirtschaftlich betreiben zu können.

3.8.3 Anwendungsfalldiagramm

Mit Hilfe von Anwendungsfalldiagrammen (auch Use Case Diagramme genannt) lassen sich mögliche Benutzerinteraktionen mit einem Software System beschreiben. Dabei finden eine Abbildung möglicher Akteure, der möglichen Aktionen sowie eine Zuordnung beider statt. Die Entwicklung des Anwendungsfalldiagramms lässt sich in zwei Abschnitte gliedern.

Zur Zwischenpräsentation wurden alle Benutzeraktionen hierarchisch dargestellt. Es gab nur wenige Einstiegs-Anwendungsfälle, unter denen alle übrigen Benutzerinteraktionen gekapselt gewesen sind. Bei dieser Art der Darstellung wurde das Zusammenspiel zwischen Akteur und dem System verfälscht dargestellt.

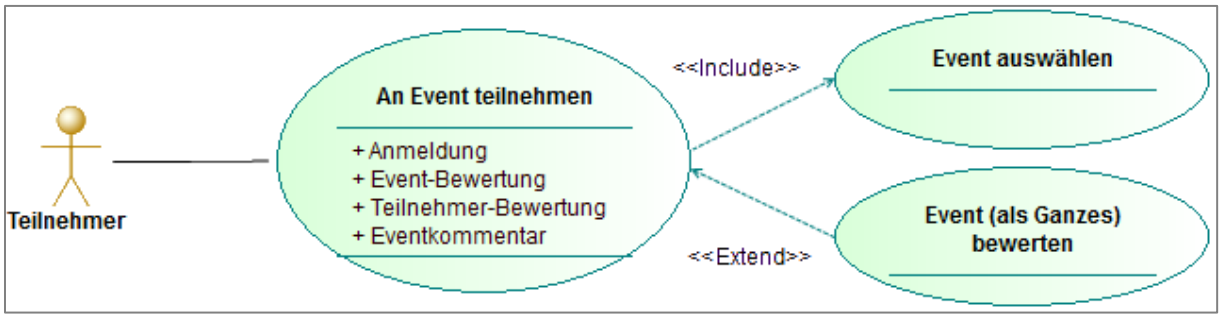

Abbildung 4 Anwendungsfall - "An Event teilnehmen" (ursprüngliche Version)[39]

Dieser Sachverhalt lässt sich anhand der Anwendungsfälle “An Event teilnehmen”, “Event auswählen” und “Event bewerten” veranschaulichen. Es ist nachvollziehbar, dass ein Event

[38] Siehe Artefakte auf CD „Kosten-Nutzenabschätzung.pdf“
[39] Eigene Darstellung.

ausgewählt werden muss, wenn man an diesem teilnehmen möchte. Die Bewertung eines Events kann aber nicht unmittelbar an die Teilnahme geknüpft werden. Aus diesem Grund wurden im zweiten Entwicklungsabschnitt des Anwendungsfalldiagramms die Beziehungen zwischen Akteur und Anwendungsfall überarbeitet.

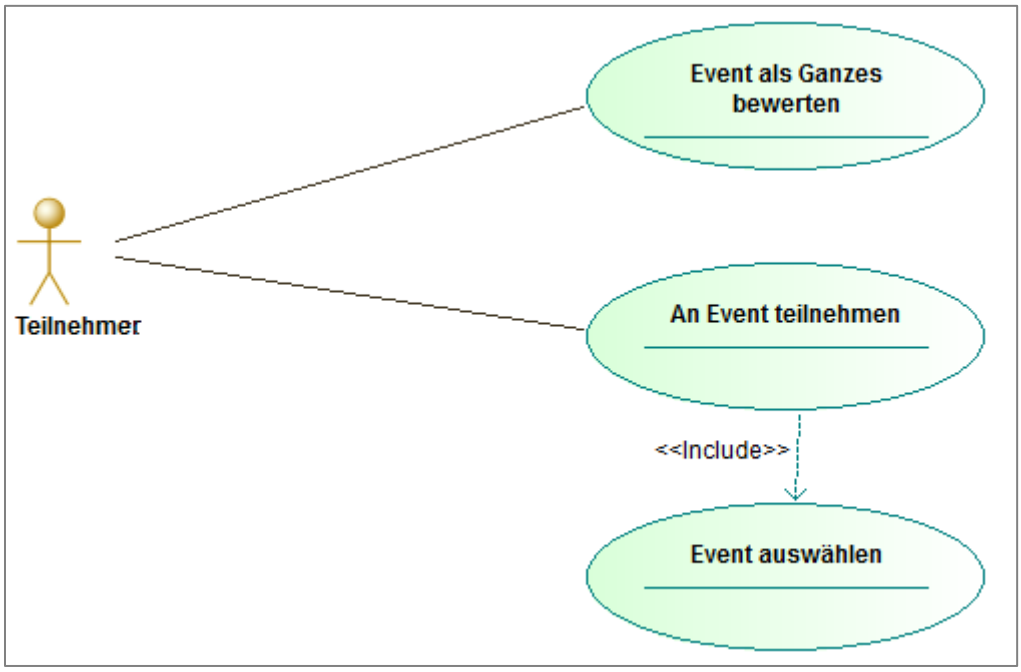

Abbildung 5 Anwendungsfall - "An Event teilenehmen" (überarbeitete Version)[40]

Die Include-Beziehung zwischen Event Teilnahme und Event Auswahl blieb bestehen. Mittelbar zusammenhängende Anwendungsfälle, wie hier die Bewertung eines Events und die Teilnahme an einem wurden direkt mit dem jeweiligen Akteur verbunden, beide Anwendungsfälle werden nicht zwingend gemeinsam aufgerufen.

Eine vergleichbare Entwicklung war bei der Hierarchie der Akteure zu verzeichnen. Zur Zwischenstandspräsentation wurden Anwendungsfälle entweder vom Akteur Teilnehmer, vom Akteur Organisator oder von beiden aufgerufen. Beide Akteure sind über eine Vererbungsbeziehung mit dem Akteur Benutzer verknüpft gewesen, der selbst mit keinem Anwendungsfall verknüpft gewesen ist.

Da der Organisator alle Anwendungsfälle aufrufen kann, die dem Teilnehmer zugeordnet sind, und einige Anwendungsfälle von jedem Benutzer aufgerufen werden können, wurden auch die Akteure überarbeitet.

[40] Eigene Darstellung.

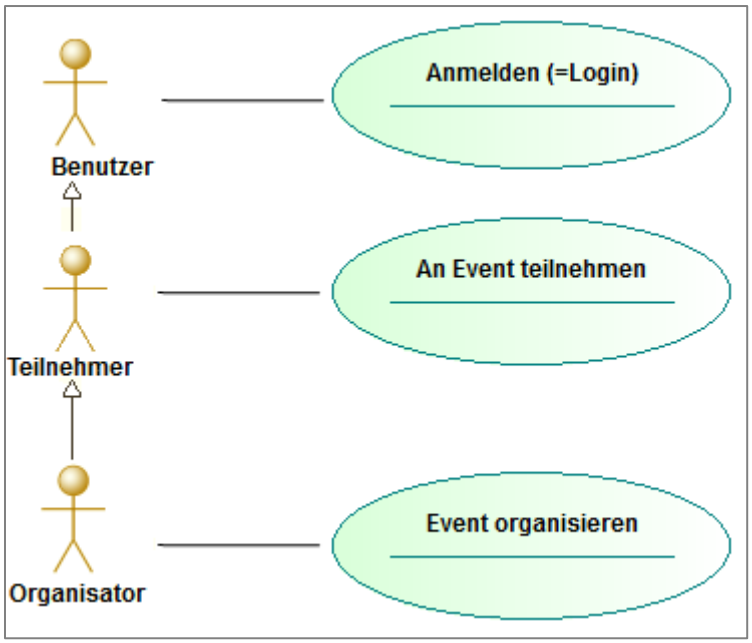

ABBILDUNG 6 ANWENDUNGSFALLDIAGRAMM - HIERARCHISCHE BEZIEHUNG DER AKTEURE[41]

Die Hierarchie der Akteure geht vom Allgemeinen, dem einfachen Benutzer mit rudimentären Rechten, ins spezielle, zum Organisator von Events. Diese Darstellung reduziert mehrfach Beziehungen zu Anwendungsfällen auf ein Minimum und hilft dadurch, das Berechtigungskonzept einfach zu gestalten

3.8.4 SZENARIEN

Die Szenario-Technik ist eine Methode in der strategischen Planung bei der mögliche zukünftige Entwicklungen festgelegt und die Auswirkungen auf das Unternehmen oder das Projekt beschrieben werden. Es werden mehrere alternative Vorstellungen der Zukunft entwickelt. Diese basieren auf unterschiedliche Ausprägungen von Einflussfaktoren in der Zukunft und mögliche Störereignisse. Die aufgestellten Szenarien besitzen unterschiedliche Eintrittswahrscheinlichkeiten. Mit Hilfe der Ergebnisse die sich aus der Auswertung der Szenarien ergibt, kann die aktuell Strategie überprüft und ggf. angepasst werden.[1][42]

Die Szenario-Technik wurde im Projekt Eventalizer nur rudimentär durchgeführt. Die Erstellung von mehreren möglichen Zukunftsbildern durch Variation in den Einflussfaktoren und das Auftreten von möglichen Störereignissen entfielen. Die Szenario-Technik modellierte das aus unserer Sicht wahrscheinlichste menschliche Verhalten bei den Wünschen und die Nut-

[41] Eigene Darstellung.
[42] Vgl. Schulze, F. (2010), S. 52f

zung der Plattform. Bei der Umsetzung haben wir User Stories in textueller Form verwendet. Die User Stories beschreiben Abfolge von Interaktionen bei der Mensch und Computer Interaktion.

Im Fokus des einen gewählten Szenarios lag auf den beiden grundlegenden Funktionalitäten der Plattform Eventalizer: Event anlegen und an einem Event teilnehmen. Die beiden User Stories waren nach dem Projekt Exposé die ersten schriftlich festgehaltenen Ausarbeitungen unseres Projekts. Im Team herrschte über die grundlegenden Funktionalitäten Einigkeit. Für die Umsetzung der Details hatte jeder seine eigene Vorstellung. Diese Vorstellungen wichen von Projektmitglied zu Projektmitglied ab. Wir waren uns z.B. einig, dass es Möglichkeiten gibt auf der Plattform Bewertungen abzugeben. Als die User Stories ausgearbeitet wurden, musste allerdings entschieden werden: Was kann bewertet werden, wer kann es bewerten und welche Bewertungsskala wird verwendet. Diese Fragen wurden gesammelt und anschließend in den wöchentlichen Jour fixes diskutiert bis ein gemeinsamer Konsens gefunden wurde.

Mit Hilfe der User Stories war es uns frühzeitig möglich ungeklärte Detailfragen wie die zum Bewertungsschema zu identifizieren und zu klären. Inkonsistenzen zwischen zukünftigen von verschiedenen Projektmitgliedern ausgearbeiteten Artefakten, wie das Lastenheft oder das Use Case Modell, konnten somit vermieden werden. Denn je später solche Inkonsistenzen entdeckt werden, desto schwieriger und zeitaufwändiger ist die Behebung solcher.

Der Inhalt der User Stories hat keinen verpflichtenden Charakter wie der Inhalt eines Lasten- oder Pflichtenheftes. Spätere Erkenntnisse oder geänderte Entscheidungen mussten somit nicht in den User-Stories nachgepflegt werden.

3.8.5 Begriffslexikon/Glossar

Das Glossar definiert die Begriffe, die für das Projekt von Bedeutung sind und wird kontinuierlich weiterentwickelt.[43] Die im Glossar aufgeführten projektspezifischen Begriffe wurden innerhalb der erstellten Artefakte während der Projektzeit verwendet. Das Glossar ist die Grundlage für die Mitarbeiter des Projekts zur Verständnis eines Begriffs. Diese Herange-

[43] Vgl. Kahlbrandt, B. (2001), S. 251f.

hensweise verhindert unterschiedlich subjektive Interpretationen, die später Auswirkungen bei der Modellierung des Systems haben könnten.

Die Erstellung eines Glossars hat uns vor allem beim Projektstart geholfen. Mit Hilfe des Dokuments konnte eine schriftliche Definition für wichtige projektspezifische Fachbegriffe hinterlegt werden. Diese konnte von allen Mitgliedern jederzeit abgerufen werden. Bei der Erstellung der einzelnen Definitionen konnten zudem ungeklärte Details identifiziert werden. Diese wurde später in den Jour fixen von allen Projektmitarbeitern gemeinsam geklärt.

Die aktuelle Definition von einem Event lautet folgendermaßen:

"Als Event wird eine Veranstaltung mit einem definierten Ziel oder einer festen Absicht bezeichnet. Das Event ist ein zeitlich begrenztes und vorab geplantes Ereignis, welches durch den Organisator eingestellt bzw. ausgelöst wird und in der dessen Verantwortung liegt. Ein Event wird durch seinen Titel und seine Beschreibung, sowie durch die Auswahl von Kategorie, Unterkategorie, Preis, minimale und maximale Anzahl von Teilnehmer- plätzen, Ort und Zeit näher spezifiziert.

An einem Event können sich andere Benutzer anmelden, d.h. teilnehmen. Sobald sich ein Benutzer für ein Event anmeldet erhält der Organisator eine private Nachricht mit den Teilnehmerdaten."[44]

Bei der ersten Definition eines Events gab es nicht die Bedingung, dass eine Person an der Plattform registriert sein muss um an einem Event teilzunehmen. Im Projektteam wurde diskutiert ob auch nicht registrierte Benutzer die Möglichkeit haben an einem Event teilzunehmen. Das Projektteam entschied sich dagegen. Wenn Personen sich nicht anmelden müssen, um an Events teilnehmen zu müssen, würde der wichtigste Grund fehlen sich auf der Plattform zu registrieren. Für eine soziale Plattform ist die Benutzerbasis und die gesammelten Daten das wichtigste Gut.

Die Entscheidung auch technische Begriffe im Glossar auszuführen ist in der Nachbetrachtung diskussionswürdig. Vor allem Begriffe wie "Java" oder "relationale Datenbank" sind in der Fachwelt eindeutig und brauchen keine projektspezifische Definition, da diese in diverser Literatur nachgeschlagen werden kann.

[44] Siehe Artefakte auf CD „Glossar.pdf", S. 2.

Im weiteren Verlauf des Projekts wurden neue Erkenntnisse oder Änderungen bei einzelnen Begriffen nicht immer im Glossar nachgezogen. Als Beispiel ist hier die Definition für den Begriff Registrierung anzuführen:

"Die Funktion Registrierung ist erforderlich um die Funktionen der Plattform Eventalizer zu nutzen. Bei der Registrierung müssen persönliche Angaben wie z.B. der Name, das Geburtsdatum, eine gültige E-Mail Adresse angegeben werden. Nach der Eingabe aller benötigten Daten muss der Benutzer noch sein Passwort für die Plattform festlegen. Nach dem Erfassen der Daten wird eine Bestätigungsmail an die vom Benutzer hinterlegte E-Mail Adresse gesendet. Mit einem Klick auf dem in der Mail enthaltenen Link bestätigt, der Benutzer seine Angaben und schaltet seinen Account auf Eventalizer frei.

Besitzt jemand bereits einen Facebook-Account kann diese Account über eine Schnittstelle direkt für die Plattform Eventalizer genutzt werden. Die oben angegebenen Schritte entfallen."[45]

Im späteren Projektverlauf wurde entschieden, dass eine eigene Anmeldung auf der Plattform Eventalizer erst in einer späteren Ausbaustufe eingeführt wird. Personen sollte sich über die Schnittstelle OAuth in Verbindung mit sozialen Netzwerken wie Facebook, Twitter oder Google+ anmelden können. Dieser Wissensstand hätte in den Glossar übertragen werden müssen.

Durch die nicht kontinuierliche Pflege des Glossars verlor dieser im späteren Projektverlauf seine Funktion als eindeutiges Artefakt übergreifendes Nachschlagewerk.

3.8.6 Lastenheft

Das Lastenheft wurde direkt zu Projektbeginn aufgesetzt, um allen Teammitgliedern einen geordneten Zugriff auf die Anforderungen an das neue Produkt zu geben. Es beschreibt aus Sicht des Anwenders, bzw. Auftraggebers, welche Funktionen und Eigenschaften wünschenswert für das neu zu erstellende System sind.[46] Im Folgenden wird selektiv auf einige Kapitel des Lastenhefts eingegangen, in denen resümierend Erkenntnisse gewonnen wurden, Verbesserungspotenziale erkennbar sind oder im Rahmen der Erstellung mehrfach grundlegend modifiziert wurden.

[45] Siehe Artefakte auf CD „Glossar.pdf", S. 3.
[46] Vgl. Kahlbrandt, B. (2001), S. 252.

3.8.6.1 Produktfunktionen

In diesem Kapitel des Lastenhefts wurden versucht alle Funktionen der Software textuell zu beschreiben. Diese wurden daraufhin den Kategorien Benutzer-, System- und Eventfunktionen zugeordnet. Da dieses Artefakt zu einem sehr frühen Zeitpunkt der Entwicklung angelegt wurde, findet sich hier die Quelle zu Informationen die an anderer Stelle, wie etwa dem Domänenklassenmodell genutzt wurden. Die Anforderungen beschränken sich jedoch nicht ausschließlich auf die zu erstellende Software Eventalizier, sondern können auch bewusst die Infrastruktur in die Abdeckung einiger Anforderungen mit einbeziehen (z. B. LF260 Statistikfunktionalität[47]).

Bei den Benutzerfunktionen wurden die Bereiche An- und Abmeldung, persönliches Profil, inkl. persönliche Konfiguration, als auch Kommunikation betrachtet. Als Beispiel der zuletzt genannten Punkt sei hier die Kommunikationsmöglichkeit über private Nachrichten genannt. Diese Art eines integrierten Nachrichtendienstes wird bei vielen Internetportalen geboten. Vorteile liegen darin, dass Medienbrüche (wie es bei Nutzung von E-Mail-Diensten der Fall wäre) verhindert werden und eine stärkere Bindung erfolgt. Der Nutzer wird beispielsweise angeschrieben, per E-Mail informiert und schaut sich daraufhin auf der Seite die Nachricht an.

Die Akteure der Eventfunktionen sind Organisatoren und Eventteilnehmer. Es wird zum einen die Organisation eines Events (z.B. Anlage) als auch die Teilnahme an einem Event inklusive der Bewertung betrachtet. So soll jeder Nutzer im Nachgang die Möglichkeit haben das Event und die Organisation zu bewerten. Hier wird implizit erhofft durch viele Bewertungen eine möglichst genaue Qualitätsbewertung erstellen zu können. So können Nutzer sich über den Organisator und dessen erstellte Events informieren, bevor sie sich anmelden.

Resümierend kann beim Lastenheft bzw. bei der Erstellung der Produktfunktionen angemerkt werden, dass innerhalb des Projektteams versucht wurde, dieses Lastenheft erst einmal möglichst vollständig und final zu erstellen, um dann erst mit der weiteren Softwarespezifizierung fortzufahren. Das gewählte Vorgehensmodell sieht aber eher eine inkrementelle bzw. iterative Softwareentwicklung vor, bei der immer wieder Änderungen am Lastenheft, wie beispielsweise das Hinzufügen weiterer Anforderungen, gemacht werden.

[47] Siehe Artefakte auf CD „LH_Eventalizer.pdf“, Kap. 2.1.3 „Administrationsfunktionen“, S. 8.

Gerade im Bereich der Produktfunktionen hätte dieses Vorgehen strikter umgesetzt werden müssen.

3.8.6.2 Anwendungsfalldiagramm

Es wurde ein gesamtes Anwendungsfalldiagramm modelliert. Dieses Anwendungsfalldiagramm enthält alle möglichen Anwendungsfälle für die Internetplattform Eventalizer, wobei die Abhängigkeit zwischen diesen Anwendungsfällen, beispielsweise die zeitliche Abfolge dieser einzelnen Anwendungsfälle, nicht abgebildet wurden.

Diese Abhängigkeiten hätten im Rahmen einer Geschäftsprozessmodellierung, beispielsweise mit Hilfe der Ereignisgesteuerten Prozesskette (EPK) oder der Business Process Model and Notation (BPMN) aufgezeigt und abgebildet werden können. Aufgrund des zeitlichen Verzugs des Projektes wurde auf diesen Aspekt bei der Anforderungsermittlung bzw. Softwarespezifizierung allerdings verzichtet.

Resümierend wäre es durchaus lohnend gewesen, mit Hilfe der EPK bzw. der BPMN Modelle für die Abhängigkeiten zwischen den Anwendungsfällen zu erstellen. Besonders in der Praxis sind solche Modelle sehr wertvoll und für die Anforderungsermittlung bzw. Softwarespezifizierung fast unabdingbar.

3.8.6.3 Domänenklassendiagramm

Mit Hilfe eines Domänenklassendiagramms kann man Softwareentwürfe aus Persistenzsicht strukturieren. Dabei handelt es sich um eine Darstellung von Entitäten, den Beziehungen dazwischen sowie der Attribute. Die Erstellung dieses Diagramms ist sinnvollerweise zwischen der Beschreibung des Funktionsumfangs einer Software und der Datenbankmodellierung angesiedelt.

Der erste Entwurf des Domänenklassendiagramms wurde auf Basis der zu dem Zeitpunkt festgelegten Produktfunktionen gemeinsam entwickelt. In darauf anschließenden Diskussionen wurden die Produktfunktionen gemeinsam mit dem Domänenklassendiagramm weiterentwickelt. Dabei wurden zum einen die Attribute für die Klassen bestimmt. Zum anderen wurden die Beziehungen zwischen den einzelnen Klassen überarbeitet oder bestätigt. Als besonders kompliziert hat sich die Abgrenzung der Klassen Event-Teilnahme und Event-Organisation voneinander dargestellt. Während die Produktfunktionen unterstellt haben, dass ein Organisator grundsätzlich an einem Event auch teilnimmt, hat sich im Domänen-

klassendiagramm die Verknüpfung des Organisators mit den weiteren Klassen des Event-Teilnehmers als ungeschickt herausgestellt. Während Teilnehmer sowohl das Event, als auch andere Teilnehmer bewerten können, soll der Organisator nur die Teilnehmer bewerten können. Hieraus ableitend wurden die Produktfunktionen so angepasst bzw. interpretiert, dass die Organisatoin und Teilnahme unabhängig voneinander sind.

Das Domänenklassendiagramm wurde im Rahmen der gemeinsamen Entwicklung zeitweise kompliziert und unübersichtlich. Am Ende des Prozesses stand jedoch ein übersichtliches klar strukturiertes Diagramm, dass gut für die Entwicklung des Datenbankschemas herangezogen werden konnte.

3.8.6.4 Qualitätsanforderungen

In diesem Unterkapitel des Lastenheftes wurde auf die Qualitätsanforderungen, d.h. den nicht-funktionalen Anforderungen der Internetplattform Eventalizer eingegangen. Diese Anforderungen unterteilten sich dabei zum einen in die äußere und innere Qualität und zu anderen in die Gebrauchstauglichkeit. Während die äußere und innere Qualität die nicht-funktionalen Anforderungen zur Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Wartbarkeit und Portabilität umfasst, sind unter der Gebrauchstauglichkeit die nicht-funktionalen Anforderungen zur Effektivität, Produktivität, Sicherheit und Zufriedenheit zu finden. Bei den Qualitätsanforderungen wurden vor allem die Anforderungen zur Zuverlässigkeit und Benutzbarkeit fokussiert. Diese Anforderungen sollten einerseits eine stabile, robuste und damit zuverlässige Internetplattform garantieren, sowie andererseits eine unkomplizierte, intuitive Benutzung über jeden aktuellen Browser ermöglichen.

Resümierend wurden die Qualitätsanforderungen zu weich und damit nicht wirklich mess- oder prüfbar formuliert. In der Praxis sollten diese Anforderungen durch den Auftraggeber konkreter formuliert werden, damit das erstellte System bzw. Produkt auch gegen diese im Lastenheft definierten Anforderungen getestet und abgenommen werden kann.

3.8.6.5 Abnahmekriterien

Die Abnahmetestfälle lehnen sich stark an die definierten Produktfunktionen und deren Kategorisierung an. Es wurden größtenteils Muss-Testfälle definiert, da es bei dieser Neuentwicklung viele Basisfunktionen gibt, ohne deren einwandfreie Funktion die ganze Software

nicht brauchbar ist. Sie definieren, welche Tests für eine erfolgreiche Abnahme erfüllt sein müssen.

Kann-Testfälle sind in der Kategorie Benutzerfunktionen z. B. das Führen von Freundes- oder Blockierlisten. Hierbei handelt es sich um systemunkritische Funktionen, welche auch per (Wartungs-)Update nachgeliefert werden können.

Zur Abnahme gehören jedoch nicht nur die Abnahmetestfälle, sondern auch die Bereitstellung der Software sowie eine Demonstration.

3.8.7 Pflichtenheft

Nach dem ersten Entwurf eines Lastenheftes mit ausreichender Qualität wurde damit begonnen, das Pflichtenheft zu erstellen. Das Pflichtenheft beschreibt die aus Sicht des Auftragnehmers, in welcher Art und Weise die im Lastenheft definierten Anforderungen des Auftraggebers umgesetzt und realisiert werden sollen. In den RUP-Artefakten wird das Pflichtenheft auch als „Software Requirement Specification“ betitelt. Im Folgenden wird daher analog zum Lastenheft selektiv auf einige Kapitel des Pflichtenheftes eingegangen, bei denen resümierend Erkenntnisse und Verbesserungspotenziale erkennbar sind.

3.8.7.1 Produktfunktionen

Im Pflichtenheft wurden die im Lastenheft skizzierten Produktfunktionen detailliert beschrieben. Die Unterteilung in Benutzer- und Eventfunktionen wurde beibehalten. Administratorfunktionen wurden jedoch nicht weiter betrachtet, da sich gezeigt hat, dass gerade Administratorfunktionen nur eine besonders geringe Schnittmenge zu den übrigen Funktionen haben.

Die Funktionen wurden detailliert inklusive der vorhandenen Felder und dem Verhalten der Felder beschrieben. Während im Lastenheft Masken mit Hilfe einfacher Mock-ups schematisiert wurden, wurden Mock-ups im Pflichtenheft analog zur Darstellung im Browser erstellt. Die auf der nächsten Seite folgende Abbildung zeigt beispielhaft einen Mock-up für die “Erstellung eines Events” (Funktionale Anforderung F60):

ABBILDUNG 7 MOCK-UP - "ERSTELLUNG EINES EVENTS" (FUNKTIONALE ANFORDERUNG F60)

3.8.7.2 ENTITÄTSKLASSENDIAGRAMM

Grundlage für die Entwicklung des Entitätsklassendiagramms, welches bei uns einem Datenbankmodell entspricht und im Folgenden auch so bezeichnet und als solches verwendet wird, war das Domänenklassenmodell. Beim ersten Entwurf des Datenbankschemas wurde ein besonderer Fokus auf das Normalisierungskonzept gelegt. Die Verwendung der dritten Normalform sollte verhindern, dass Daten bei der Erstellung redundant gespeichert werden und die Daten im Betrieb bei Änderungen konsistent bleiben. Im Anschluss wurde an wenigen ausgewählten Tabellen im Hinblick auf eine Performanceverbesserung eine Denormalisierung vorgenommen. So wurde das Attribut *anzahlTeilnehmer* in der Tabelle Event hinzugefügt. Die Anzahl der Teilnehmer ließe sich auch über die Tabelle Teilnehmer berechnen, in dem nur die Anzahl der Datensätze gezählt werden, die die gesuchte *idEvent* aufweisen. Da die Suche Events mit freien Plätzen oder die Detailansicht eines Events einige der häufigsten Abfragen sein wird, erfolgt mit der Auslagerung des Wertes in ein Attribut eine Performancesteigerung. Bei jeder Anmeldung oder Absage an einem Event muss dafür nun ein Update auf das Attribut *anzahlTeilnehmer* erfolgen.

Als eines der Hauptdesignelemente haben wir uns darauf verständigt soweit es sinnvoll ist Surrogatschlüssel in den Tabellen zu nutzen. Es wurden numerisch fortlaufende künstliche Schlüssel gewählt, da dieses am einfachsten und sehr gebräuchlich ist.[48] Im Kontext der privaten Nachrichten wurde allerdings auf den Einsatz eines Surrogatschlüssels verzichtet, da hier eine Kardinalität von n:m mit dem Benutzer vorliegt. Diese wird durch eine zusätzliche Tabelle (*PrivMsgAn*) umgesetzt, welche die Primärschlüssel der beiden Tabellen Benutzer und *PrivMsg* als Fremdschlüssel enthält. Hierdurch wird die n:m-Beziehung in zwei 1:n-Beziehungen aufgelöst.

Im Domänenklassenmodell hat der Benutzer die beiden Attribute *Freundesliste* und *Blockierliste*. Diese wurden zunächst als zwei eigenständige Tabellen in das Datenbankmodell übernommen.

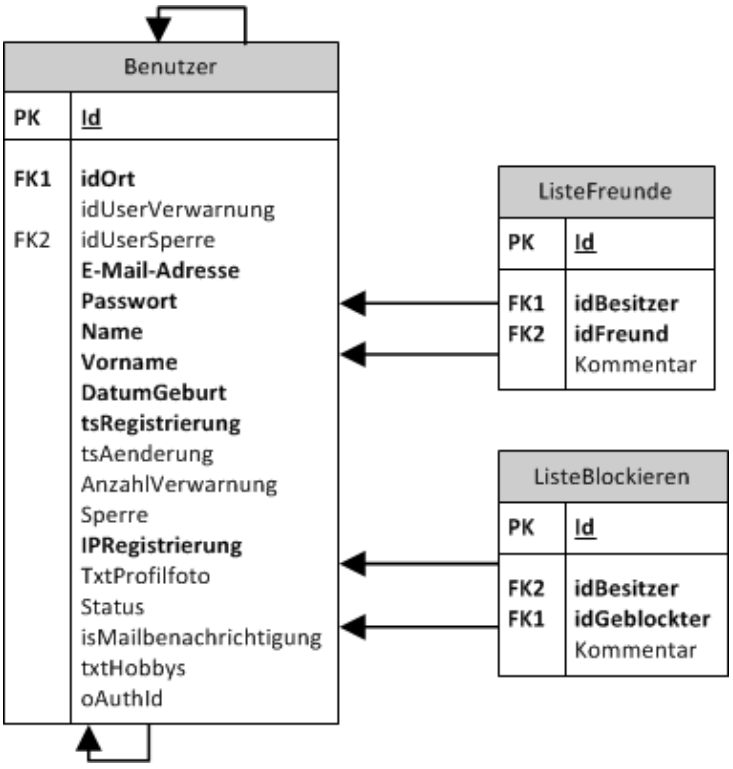

ABBILDUNG 8 DATENBANKSCHEMA - FREUNDES-/BLOCKIERLISTE (URSPRÜNGLICHE VERSION)[49]

Nach kurzer Zeit kam jedoch die Frage auf, was die beiden Tabellen unterscheidet, wo also die Berechtigung für zwei separate Tabellen liegt.

[48] Vgl. Brücher, C.; Jüdes, F.; Kollmann, W. (2011), S. 187.
[49] Eigene Darstellung.

Beide Tabellen waren exakt gleich aufgebaut, als Unterschied hieß ein Fremdschlüssel jedoch *idBlockierter*, bzw. *idFreund.* Die Bezeichnung der Tabellen unterschied sich ebenso, sodass lediglich die Metadaten für den Unterschied sorgten.

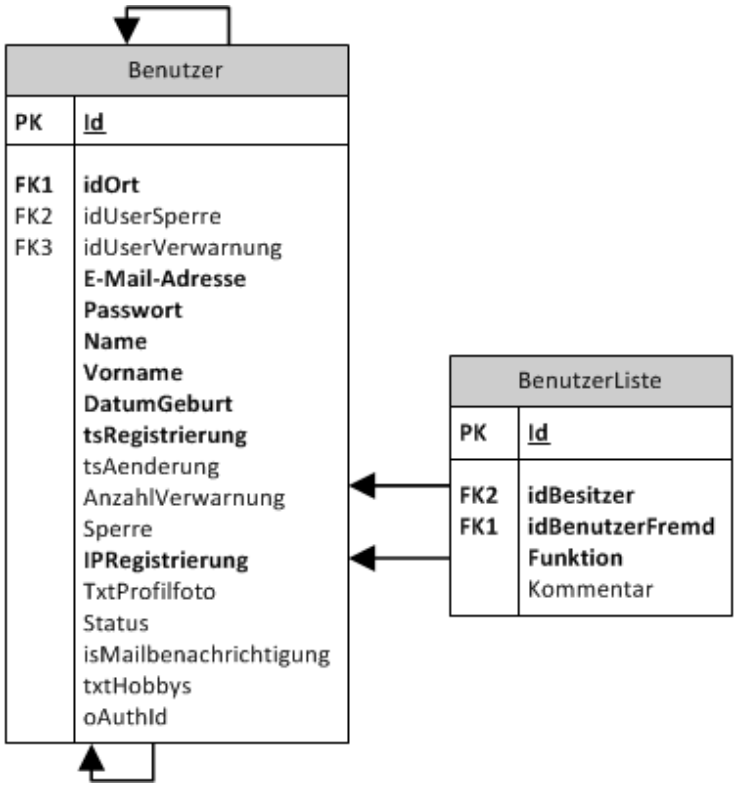

ABBILDUNG 9 DATENBANKSCHEMA - FREUNDES-/BLOCKIERLISTE (ÜBERARBEITETE VERSION)[50]

Dieser Unterschied wurde daher in ein eigens Feld namens *Funktion* ausgelagert, mit welchem spezifiziert werden kann, ob es sich bei dem aufgenommen Benutzer um einen Freund oder um einen blockierten Nutzer handelt. Somit konnte eine Tabelle entfallen. Ein weiterer Vorteil besteht in der Erweiterbarkeit (z. B. bei mehreren Freundeslisten). Die Felder *idBlockierter*, bzw. *idFreund* wurden so neutral wie möglich in *idBenutzerFremd* umbenannt.

Eine weitere Besonderheit des Datenbankentwurfs ist die Modellierung der Kategorien. Aus den Anforderungen im Lastenheft (s. Kapitel 2.3 „Produktdaten“) ist bekannt, dass ein Event eine Kategorie und eine Unterkategorie besitzen soll.

Zunächst wurden daher die beiden Tabellen *Kategorie* und *Unterkategorie* angelegt.

[50] Eigene Darstellung.

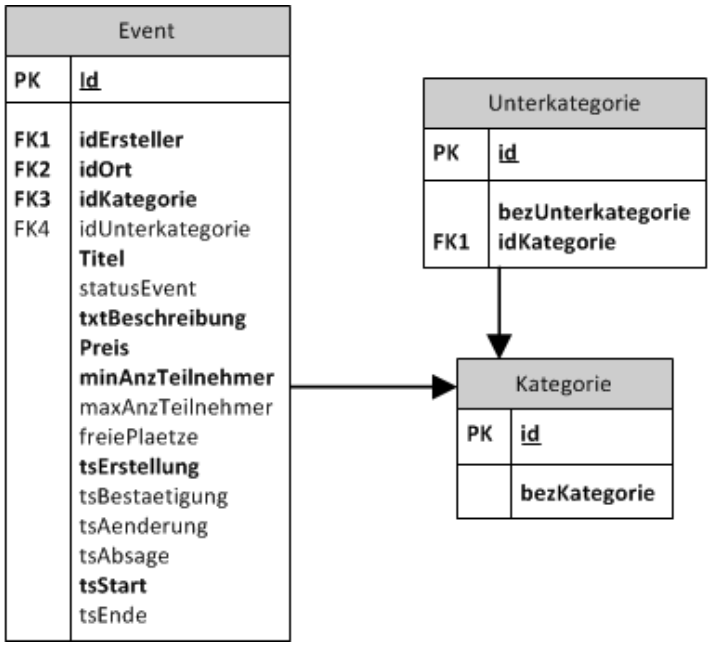

ABBILDUNG 10 DATENBANKSCHEMA - EVENT-KATEGORIE (URSPRÜNGLICHE VERSION)[51]

In der Tabelle *Unterkategorie* war der künstliche Primärschlüssel der Tabelle *Kategorie* eingebunden, welcher den Bezug zur Oberkategorie herstellen sollte. Auch hier stellte sich die Frage nach dem Nutzen zweier Tabellen mit ähnlichen Inhalten. Die finale Modellierung sieht nun eine Tabelle *Kategorie* mit einem rekursiven Fremdschlüssel vor, welcher die Oberkategorie darstellt.

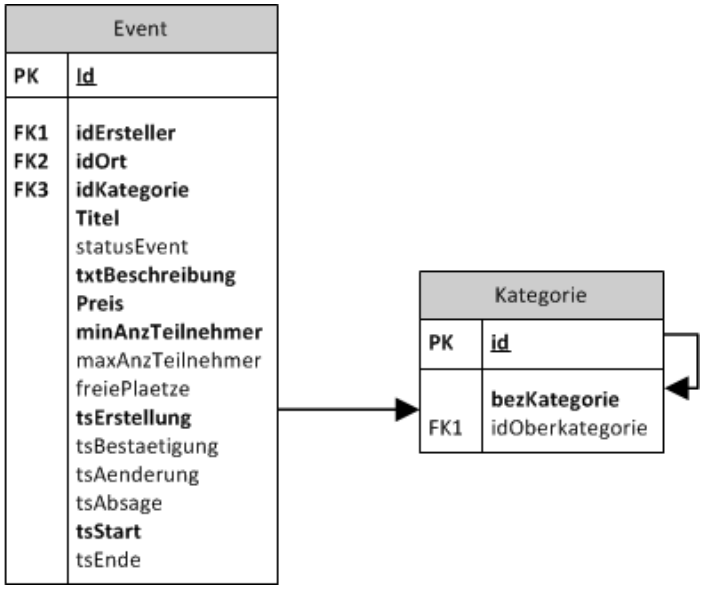

ABBILDUNG 11 DATENBANKSCHEMA - EVENT-KATEGORIE (ÜBERARBEITETE VERSION)

Auch hier ist der Vorteil, dass neben dem Wegfall einer Tabelle nun nicht nur zwei Ebenen realisiert werden können, sondern unendlich viele.

[51] Eigene Darstellung.

3.8.7.3 Aktivitätsdiagramm

Aktivitätsdiagramme haben die Aufgabe den Ablauf von Prozessen sprach- und technologie-unabhängig aufzuzeigen. Die ausgeführten Aktionen können von verschiedenen Systemen wie einem Benutzer oder einem Softwareprogramm ausgeführt werden.[52]

Die Identifikation der wichtigsten Aktivitäten im Projekt Eventalizer erfolgte mit Hilfe der Szenarien-Technik und der Erstellung eines Anwendungsfall Diagramms. Hierdurch wurden die Aktivitäten Anmelden, Event anlegen und Event teilnehmen als zentrale Bausteine der Plattform Eventalizer ermittelt. Diese drei Aktivitäten bilden die wesentlichsten sowie im späteren Betrieb am häufigsten aufgerufenen Funkionen der Plattform. Die Modellierung von weiteren Aktivitäten war aufgrund des sehr engen Zeitplans nicht möglich.

Bei der Modellierung der Aktivitätsdiagramme wurde der Fokus auf den Kontrollfluss gelegt. Die Datenflüsse einzelner Objekte wurden aufgrund ihrer hohen Veränderbarkeit zu diesem Zeitpunkt des Projekts nicht näher betrachtet.

Die einzelnen Aktionen innerhalb der Aktivitätsdiagramme wurden einem Verantwortlichem zugeteilt. In den aufgeführten Diagrammen ist dies der Benutzer, der Aktionen ausführt, Datenfelder ausfüllt oder Entscheidungen trifft oder die Software für die Plattform Eventalizer. Diese ist für die Darstellung, die Speicherung der Daten, sowie die Reaktion auf die Aktionen des Benutzers zuständig. Der Verantwortungsbereich “Eventalizer” hätte auch beispielsweise in den genannten Teilsystemen wie Datenbank, Darstellung und App-Server weiter unterteilt werden können.

Im Diagramm “Anmeldung” kam mit “Fremdsoftware” ein dritter Verantwortungsbereich hinzu. Die Anmeldung auf der Plattform Eventalizer geschieht über die Schnittstelle OAuth in Verbindung mit ausgewählten bereits bestehenden sozialen Plattformen wie Facebook, Google+ oder Twitter. Die in diesem Verantwortungsbereich aufgeführten Aktionen sind nur beispielhaft aufgeführt, da keine Kontrolle oder Einblick in die Fremdsoftware besteht und nur die Schnittstelle zur Übergabe und Empfang der Daten an die Fremdsoftware definiert ist.

[52] Vgl. Vogel-Heuser, B. (2009), S. 68.

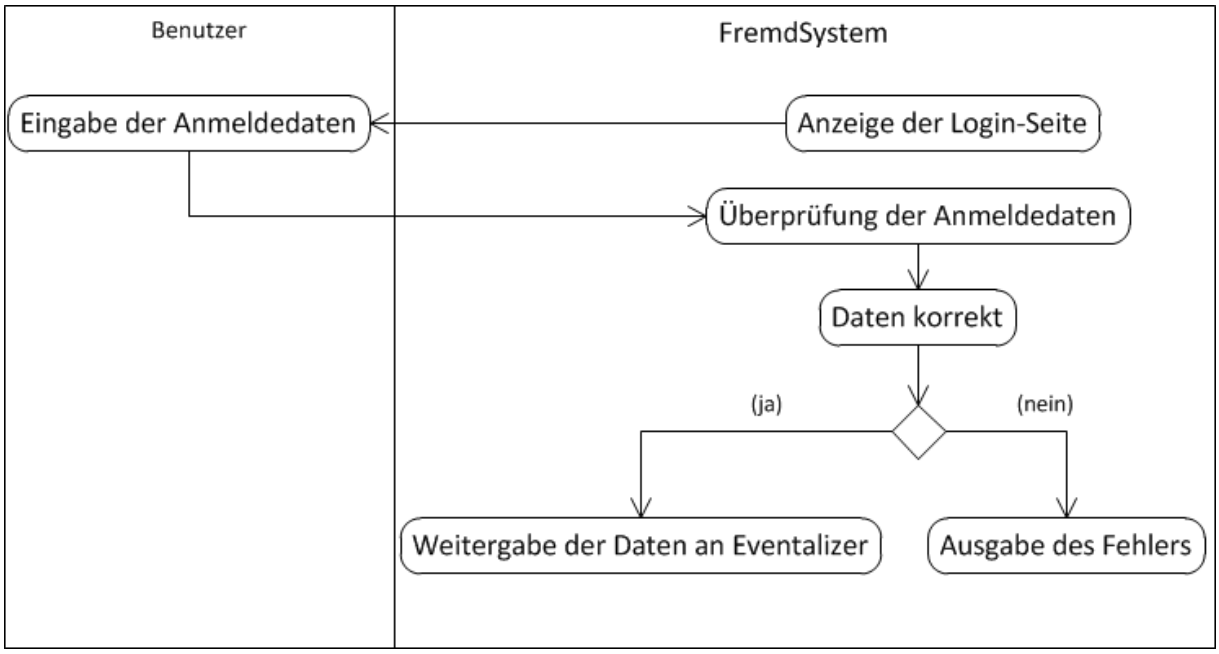

ABBILDUNG 12 AKTIVITÄTSDIAGRAMM - ANMELDUNG VIA FREMDSYSTEM[53]

Zur Übersicht und leichteren Verständlichkeit des Aktivitätsdiagramms „Anmeldung“ wurden die Aktionen aufgenommen.

In einer früheren Version der Aktivitätsdiagramme waren einzelnen Aktionen modelliert die es dem Benutzer ermöglichten das Diagramm zu verlassen. Wenn dies konsequent modelliert wird, muss der Benutzer bei fast jeder Aktion auch die Möglichkeit haben, die Aktivität abzubrechen. Eine solche Modellierung würde das Diagramm unübersichtlich machen. Die Möglichkeit eine Aktivität zu verlassen ist bei einer heutigen Internet-Anwendung eine selbstverständliche Standardanforderung.

Bei der Erstellung der Aktivitätsdiagramme wurde darauf geachtet, dass die aufgeführten Aktionen in einem zeitlich abgeschlossenen Rahmen stattfinden. Welche Aktionen der Benutzer nach einer möglichen Anmeldung durchführt ist zum einem z.B. für das Aktivitätsdiagramm Anmeldung unerheblich zudem kann nicht mit Sicherheit bestimmt werden, ob der Benutzer überhaupt noch eine Aktion durchführt.

[53] Eigene Darstellung.

3.8.7.4 Sequenzdiagramm

Der Kommunikationsfluss zwischen Benutzer, dem eigenen System sowie des OAuth-Partners wurde im nachfolgenden Sequenzdiagramm dargestellt.

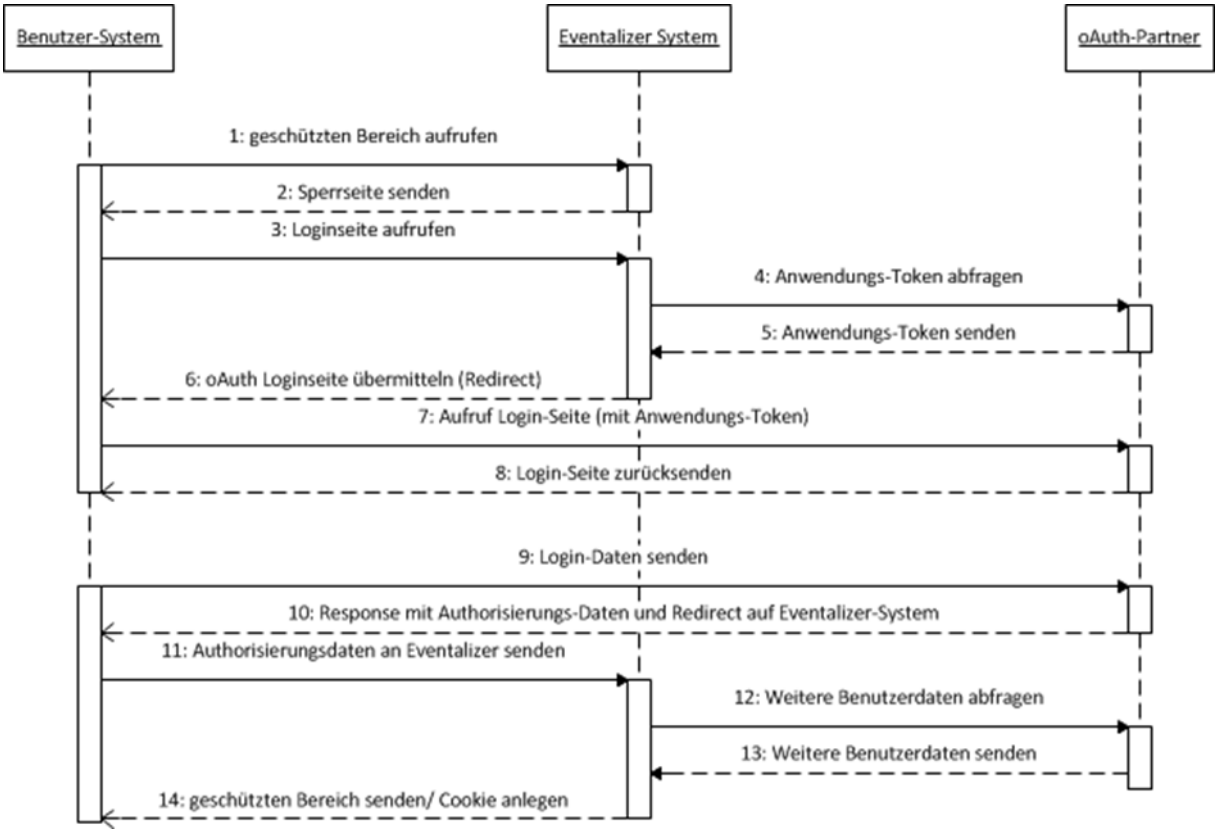

Abbildung 13 Sequenzdiagramm - Login via OAuth[54]

Dabei ist der Kommunikationsfluss durch die Verwendung des OAuth-Protokolls größtenteils vorgeben. Das Sequenzdiagramm wurde dabei vornehmlich zu eigenen Dokumentation und Veranschaulichung erstellt. Der Inhalt der Nachrichten beschränkt sich auf geheime Schlüssel die vorher zwischen der Anwendung, dem Benutzer und dem OAuth-Partner vereinbart wurden und die zur gegenseitigen Identifikation dienen ohne personengebundene Daten zu übermitteln und die Übermittlung von eben jenen Personendaten nach erfolgreicher Kommunikation. Die Kommunikation erfolgt dabei über das HTTP-Protokoll und damit synchron. Dieses wurde zuerst im Sequenzdiagramm fälschlich als asynchrone Kommunikation dargestellt und im Projektverlauf (und auch im Pflichtenheft) korrigiert.

Für den Benutzer des Eventalizers wird zur Laufzeit ein Schlüssel generiert, womit er sich beim OAuth-Partner und dem Eventalizer identifiziert. Der Eventalizer selbst hat jeweils ei-

[54] Eigene Darstellung, siehe Artefakte auf CD „PH_Eventalizer.pdf“, Kap. 2.5. „Sequenzdiagramm“, S. 17.

nen festen geheimen und öffentlichen Schlüssel vom OAuth-Partner bekommen in dem die Anwendung beim OAuth-Partner registriert wurde. Dies erfordert einen Benutzer-Account eines Entwicklers beim OAuth-Partner, welches am Beispiel vom Facebook zu Komplikationen führt, da Facebook einen ausreichen gefüllten Benutzer-Account erwartet, d.h. dass beispielsweise ein Facebook-Dummy-Account nicht verwendet werden konnte.

3.8.7.5 Abnahmekriterien

Die Abnahmekriterien des Pflichtenheftes versuchen die Abnahmekriterien des Lastenheftes durch konkrete Testfälle abzudecken. Diese Testfälle sind sehr viel feingranularer, als die Abnahmekriterien im Lastenheft. Es wurde darüber hinaus darauf geachtet, dass das Umfeld definiert ist. Jeder Testfall könnte anders ausfallen, wenn sich das System in unterschiedlichen Zuständen befindet würde.

Die Abnahmekriterien sind in Kann- und Muss-Testfälle eingeteilt, um zu definieren, auf welche Testfälle besonderes Augenmerk gelegt werden muss. 90% der Testfälle müssen erfolgreich durchgeführt werden, jedoch darf kein Muss-Testfall negativ abgeschlossen sein.

Im Nachhinein konnten nicht alle ursprünglich vorgenommen Funktionalitäten implementiert werden, so dass die Abnahmetests nicht durchgeführt wurden, da ein vollständig realisierter Prototyp noch nicht vorhanden war. Aus Projektsicht wäre daher die am Anfang definierten Abnahmetests nicht bestanden worden bzw. das konnte nicht im vorgegebenen Zeitraum beendet werden. Die Verzögerungen ergeben sich nach eigener Meinung, wie oben bereits beschrieben, zum größten Teil aus dem nicht eingespielten Projektteam, welches zu viel Zeit mit niedrigpriorisierten Sachverhalten verbracht hatte und damit zu spät auf Fehler und Verzögerungen reagieren konnte.

In der Nachbetrachtung wurde als eine Art “lessons learned” festgestellt, dass ein schon von Anfang an entwickelter, wenn auch noch wenig funktionsfähiger Prototyp, schon sehr zeitnah gegen vorhandene Abnahmetestfälle geprüft hätte werden können. Durch diese Testgetriebenen Entwicklungsansatz hätten Programmfehler, fehlende Funktionalitäten, etc. frühzeitig aufgedeckt werden können. Dies würde, insbesondere in der Praxis, nachträgliche und damit teurere Nach-Programmierungen von Funktionalitäten bzw. dem Ausbau von Fehlern, vorbeugen.

3.8.8 Anwendungsarchitektur

Durch die Einbindung von Spring Roo wurde die Architektur der Eventalizer-Anwendung bereits stark vorgeben. Es handelt sich um eine typische 4-Schichten-Webarchitektur mit einem Model-View-Controller Pattern mit Präsentations-, Steuerungs-, Geschäftslogik- und Datenbankzugriffsschicht.

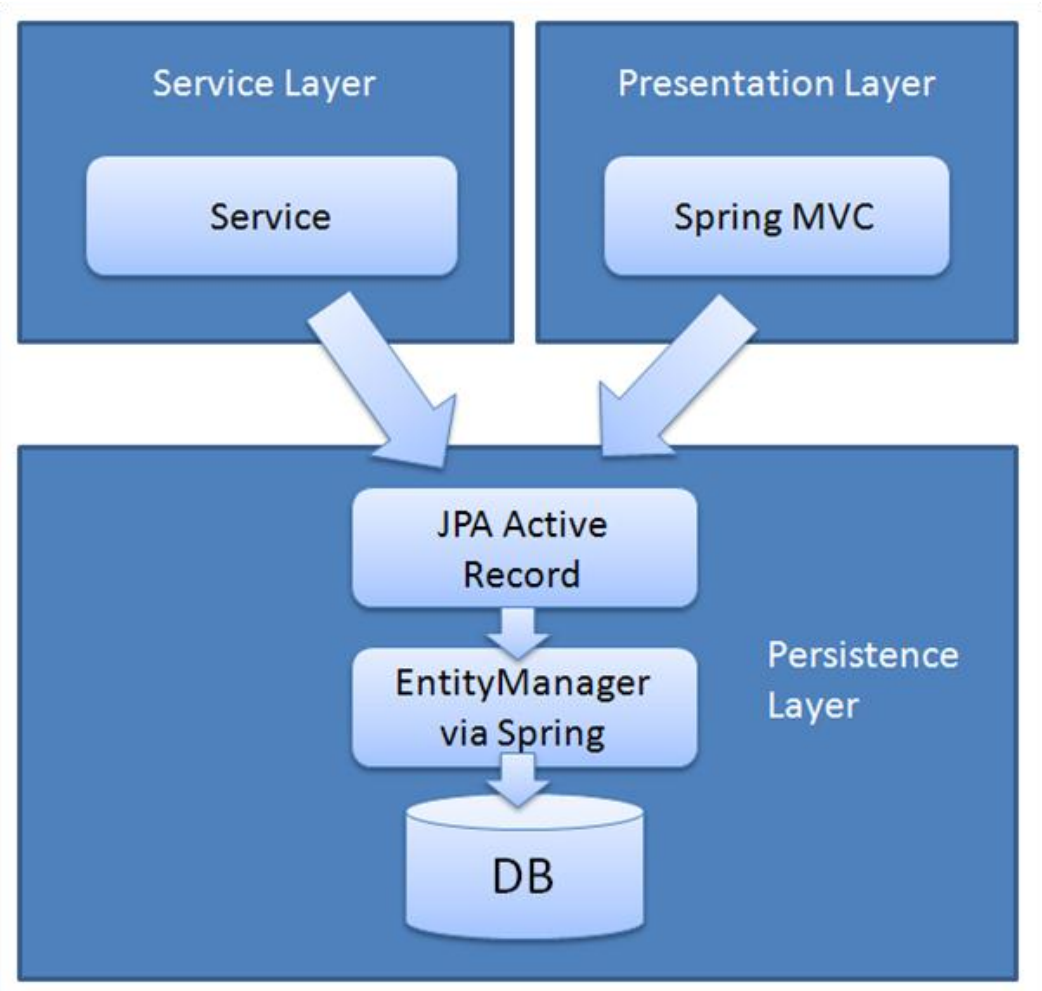

Abbildung 14 Spring Roo Webarchitektur für Evantelizer[55]

Spring Roo trennt allerdings weniger stark zwischen Geschäftslogik und Datenbankzugriff. Dieser geschieht bei den meisten Webanwendung durch die Verwendung eines Data-Access-Objects (DAO), welches die Operationen zur Datenbank bereitstellt. In Spring Roo werden stattdessen die CRUD-Methoden vom Spring Framework direkt in die Model-Klassen gene-

[55] Eigene Darstellung basierend auf o.V. (2012a).

riert. Daher werden die Model-Klassen und der Datenbankzugriff in Spring Roo-Dokumentationen als „Entity Layer“[56] zusammengefasst.

Die Architektur und damit die Nutzung des Spring Roo Framework war sicherlich geeignet für die Umsetzung des Projektes. Sicherlich sind Rapid-Development-Frameworks kein Allheilmittel in der Programmierung die auf magische Weise generieren, was der Entwickler sich wünscht. Der generierte Code muss anschließend vom Entwickler angepasst werden. Der Zeitdauer der Implementierung wird dadurch nicht wesentlich verkürzt. Jedoch passt es zur iterativen Vorgehensweise eher vorgenerierten Code anzupassen, als zunächst ein komplettes Anwendungsgerüst selber zu schreiben. Spring Roo verringert dadurch die Zeit, die für einzelne Iterationszyklen benötigt wird, so dass nachträgliche Änderungen schneller eingefügt werden können bzw. Probleme schneller erkannt werden. Trotzdem gab es insbesondere bei fehlendem Know-how der Projektmitarbeiter durch die erstmalige Nutzung des Frameworks und einiger Schlüsseltechnologien (Apache Maven) Probleme bei der Umsetzung. So kam es vermehrt zu Problemen funktionierende Workspaces auf unterschiedlichen PCs einzurichten. Schwierigkeiten gab es insbesondere bei den vom Apache Maven nachgeladenen Jar-Dateien. Dies führte teilweise dazu, dass zeitweise bestimmte Projektmitglieder von der Entwicklung ausgeschlossen wurden. Das Deployment und die Vorgehensweise von Apache Maven im Gegensatz zu einem einfachen Ant-Build einer Anwendung ist zwar durchaus beeindruckend. Allerdings haben die persönlichen negativen Erfahrungen, die sich nicht ausschließlich durch ungeschickte Bedienung erklären lassen, zu einer ambivalenten Einstellung zu Maven geführt. Diese konnten jedoch größtenteils behoben werden und waren daher auch eine interessante persönliche Bereicherung für die berufliche Praxis.

[56] Vgl. o.V. (2012a).

4 PROJEKTABSCHLUSS

In diesem Kapitel erfolgt eine zusammenfassende Reflexion des gesamten Projekts und einiger Themenübergreifender Problemstellungen. Der Bericht endet mit einem Ausblick auf die potentielle Weiterentwicklung des Projekt „Eventalizer“.

4.1 ZUSAMMENFASSUNG

In der heutigen Arbeitswelt und im Bereich der Softwareentwicklung steigt die Nachfrage nach hoch spezialisierten Fachkräften. Die Spezialisten werden nur in einzelnen und zwar ihrem Spezialgebietet entsprechenden Phasen eingesetzt. Mit Hilfe des Projekts war es uns möglich einen Gesamteindruck von den verschieden Stufen, über die Idee, den Auftrag, die Konzeption und auch die Realisierung zu bekommen. Dies hat uns einen Einblick in die Zusammenhänge der einzelnen Phasen und die verschiedenen Anforderungen und Blickwinkel, die dort vorherrschen, ermöglicht. Diese unterschiedlichen Betrachtungswinkel zu verstehen, ist ein wichtiger geschaffener Mehrwert in der Zusammenarbeit und dem Verstehen von Mitarbeitern aus verschiedenen Bereichen eines Unternehmens und wird uns in der Praxis helfen. Für viele von uns war es eine der wenigen Gelegenheiten bei einem Projekt in jeder Phase eingebunden und für verschiedene Aufgaben (mit-)verantwortlich gewesen zu sein.

Die Entscheidung, dass es kein bindendes Rollenkonzept im Projektteam gab, ist in der Nachbetrachtung dennoch diskussionswürdig. Trotz der Vorteile hätten wir zwei feste Rollen im Projekt Eventalizer vergeben sollen. Dies ist zu einem der Kunde und zum anderen der Projektleiter. Durch das Fehlen dieser beiden Rollen hatten wir eine sehr flache Hierarchie in unserem Projektteam. Dies hatte zur Folge, dass in den Jour fixes über jedes Detail diskutiert wurde. Ein Projektleiter oder ein Kunde hätten festgefahrene Diskussionen abbrechen und eine bindende Entscheidung treffen können.

Der geleistete Aufwand der einzelnen Projektmitglieder war durch das nicht vorhanden Rollenkonzept unterschiedlich. Genauso wie auch im beruflichen Alltag engagierten sich einzelne Mitglieder in jeder Projektphase und übernahmen selbstständig oder freiwillig Aufgaben. Andere Projektmitglieder pflegten eine passivere Haltung und warteten auf die Zuweisung von Aufgaben. Im Nachhinein half uns diese Erfahren aber auch dabei zu lernen, dass zu ei-

nem jedem Projekt unterschiedliche Typen von Projektmitgliedern bzw. Mitarbeitern zwingendermaßen notwendig sind, um das Projekt erfolgreich zu gestalten bzw. abzuschließen.

Leider wurden auch nicht immer alle Aufgaben termingerecht erfüllt, so dass es zu Verschiebungen im Projektplan kam. Da es nicht wie in der Arbeitswelt einen Projektleiter bzw. Vorgesetzten oder einen Kunden gab, der mit Konsequenzen drohen könnte, hätte vielleicht als "Lösungsansatz" eine Projektkasse helfen können. Bei verspäteten Abgaben von Aufgaben hätte ein kleiner monetärer Obolus in diese eingezahlt werden müssen.

Während der Konzeptionsphase wurde viel Zeit darauf angewendet, Funktionen des Softwareprodukts inklusive vieler Sonderfälle zu beschreiben. Daraus resultierte, dass mehr Zeit als veranschlagt für die Konzept aufgewendet wurde und die Implementierung nur in Teilen erfolgen konnte. Dieser Prozess veranschaulicht, wie sich der Blick auf das Projektergebnis erst während der Projektbearbeitung entwickelt hat. Der Anspruch, eine vollständige Software, die alle Eventualitäten abdeckt, hat sich als zu hoch herausgestellt. Für einen gleichmäßigeren und geradlinigeren Projektfluss wäre hier hilfreich gewesen, von Anfang an eine klare, gemeinschaftliche Vorstellung des Endergebnisses zu haben.

Ebenso kam es auch bei der Implementierung zu Problemen. Eine Heterogene Systemlandschaft mit einer jeweils unterschiedlichen Konfiguration hat dazu geführt, dass überproportional viel Zeit in die Konfiguration der Entwicklungssysteme investiert wurde. Die Vorgehensweise anderer Gruppen, die Entwicklung in einer jeweils identischen virtuellen Maschine durchzuführen hätte geholfen, die Zeit für die Entwicklung zielgerichtet aufzuwenden. Durch diese Probleme musste ebenfalls in der Implementierung der Funktionsumfang des Endproduktes erneut reduziert werden, ein Schwerpunkt wurde dabei auf technische Aspekte gelegt.

Nichtsdestotrotz ist das Projekt, obwohl im Projektverlauf einige Umwege in Kauf genommen werden mussten, ein absoluter Erfolg gewesen. Insbesondere die Diskussionen in den regelmäßigen Projektmeetings haben für jedes Projektmitglied einen Lernerfolg erzeugt. Es konnten wichtige Erfahrungen auch für das Berufsleben gesammelt werden, die dabei helfen werden, vergleichbare Fehler im größeren Umfeld zu erkennen und zu umgehen.

4.2 Ausblick

Die Webanwendung Eventalizer bietet mit der Umsetzung der während des Projekts festgelegten Anforderungen solide Grundfunktionen und einige darüber hinaus gehende Komfortfunktionen (z.B. alternative Login-Möglichkeiten).

Während der Ausarbeitung sind noch einige Funktionen besprochen worden, aber aufgrund des beschränkten Projektumfangs zurückgestellt worden. Es wäre daher ohne weiteres möglich ein direktes Nachfolgeprojekt zu starten, um diese Anforderungen umzusetzen.

Ein allgegenwärtiges Thema dieser Tage ist der zunehmende Medienkonsum über mobile Geräte, wie etwa Smartphones oder Tablets. Der Eventalizer in seiner gegenwärtigen Ausprägung ist ausschließlich für den Aufruf von einem PC entwickelt und bietet keine Darstellungsform, welche für diese mobilen Endgeräte angepasst ist. Eine solche neue Oberfläche würde den Bedienkomfort erhöhen und die Nutzer könnten auf mobilen Geräten intuitiver mit der Software interagieren. Da diese Endgeräte auch oft GPS-Sensoren beinhalten, bietet es sich an Standortbezogene Dienste zu integrieren. Beispielsweise lässt sich ein Besucher Events in seiner Nähe anzeigen, ohne den Standort eintippen zu müssen.

Im Hinblick auf mögliche Datenschutzanforderungen wäre es sinnvoll Privatsphäreneinstellungen anzubieten. Hierbei hat der Nutzer die Möglichkeit zu wählen, welche Informationen öffentlich zugänglich sein sollen. In der derzeitigen Form wird dies über eine globale Standardeinstellungen gelöst, eine weitere Individualisierung wäre jedoch ein Schritt weiter in die Richtung, dem Nutzer selbst die Kontrolle über seine Daten zu geben.

Neben dem Ausblick auf Funktionserweiterungen muss jedoch auch die Frage gestellt werden, wie die Realisierungschancen des Projekts sind. Die durch dieses Projekt geleistete Vorarbeit in Planung und Konzeption verspricht eine schnelle Implementierung. Jedoch funktioniert eine solche Plattform nur, wenn die “kritische Masse” an Nutzern gewonnen werden. Außerdem sind Wartungsaufwand und Betriebskosten nicht zu verachten.[57] Dem Projekt wird daher insgesamt eine niedrige Chance eingeräumt ernsthaft realisiert zu werden.

[57] Siehe Artefakte auf CD „Wirtschaftlichkeitsbetrachtung.pdf“.

VI LITERATURVERZEICHNIS

Brücher, C.; Jüdes, F.; Kollmann, W. (2011):
SQL Thinking - Vom Problem zum SQL-Statement, 1. Aufl., Heidelberg 2011.

Cooper, A.; Reimann, R. et al. (2010):
Cooper, A.; Reimann, R., Cronin, D.: About Face - Interface und Interaction Design. 1. Aufl., Heidelberg 2010.

Kahlbrandt, B. (2001):
Software-Engineering mit der Unified Modeling Language,2. Aufl., Heidelberg 2001.

Schatten, A.; Biffel, S. et al. (2010):
Schatten, A.; Biffel, S.; Demolsky, M.; Gotischa-Franta, E.; Östreicher, Th.; Winkler, D.: Best Practice Software-Engineering. 1. Aufl., Heidelberg 2010.

Schulze, F. (2010):
KMU im Wandel: Mehrwert im mittelständischen Unternehmen durch Implementierung eines Beschaffungscontrollings, 1. Aufl., Hamburg 2010.

Vogel-Heuser, B. (2009):
Automation & Embedded Systems - Effizienzsteigerung im Engineering, 1. Aufl., Kassel 2009.

VII URL-VERZEICHNIS

Kaak, J. (2012):
Strategische Partner für den Start, http://www.mittelstandswiki.de/wissen/Venture_Capital, Überprüft am 28.05.2012.

o.V. (2012):
SpringSource Tool Suite - The Best Development Tool for Enterprise Java, http://www.springsource.com/developer/sts; Überprüft am 07.06.2012.

o.V. (2012a):
Spring Framework – Chapter 3. Application Architecture, http://static.springsource.org/spring-roo/reference/html/architecture.html, Überprüft am 07.06.2012.

o.V. (2012b):
Spring Framework – Appendix C. Project Background, http://static.springsource.org/spring-roo/reference/html/background.html#background-mission, Überprüft am 07.06.2012.

o.V. (2012c):
Spring Framework – Chapter 6. Removing Roo, http://static.springsource.org/spring-roo/reference/html/removing.html, Überprüft am 07.06.2012.

Pronschinske, M. (2010)
Spring Roo 1.0 - a RAD tool for Java, http://java.dzone.com/news/spring-roo-10-rad-tool-java, Überprüft am 07.06.2012.

Schwaber, K. (2012):
ScrumButs and Modifying Scrum, http://www.scrum.org/scrumbut, Überprüft am 04.06.2012.

VIII EIGENSTÄNDIGKEITSERKLÄRUNG

“Hiermit versichern wir an Eides statt, dass diese Arbeit selbständig und ohne unzulässige fremde Hilfe erstellt worden ist, keine anderen Quellen und Hilfsmittel als die angegebenen benutzt und die Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht haben. Das gleiche gilt auch für eingefügte Zeichnungen, Karten Skizzen und Darstellungen.”

Münster, den 11.06.2012.

Matthias Beer	Alexander Benölken	Martin Garrels
Felix Schulze Mönking	Felix Wessel	Patrick Wiebeler