Technology
Arts Sciences
TH Köln

Prozessassessement & Fazit

DTSharing

Prozessassessment & Fazit des Projekts DTSharing für das Modul “Entwicklung interaktiver Systeme”

Betreuer

Prof. Dr. Gerhard Hartmann
Prof. Dr. Kristian Fischer
Ngoc-Anh Dang
Jorge Pereira

Studierende

Thomas Friesen	thomas.friesen@smail.th-koeln.de	11092095
Johannes Kutsch	johannes.kutsch@smail.th-koeln.de	11090517

Einleitung

Im folgenden wird der Projektverlauf des Projektes DTSharing beschrieben und kritisch reflektiert. Das Dokument ist in die drei größten Meilensteine “Konzept”, “Dokumentation” und “Implementation” gegliedert und endet in einem Fazit, in dem der Zielerreichungsgrad diskutiert wird.

Meilenstein 1: Konzept

Im erstem Meilenstein wurde das Projektteam vor die Aufgabe gestellt ein Problem zu Identifizieren und ein Konzept zur Lösung dieses Problems, durch ein System mit verteilter Anwendungslogik, zu erstellen.

Die Projektgruppe wurde während der ersten Einführungsveranstalltung gebildet, die beiden Teammitglieder kannten sich nur flüchtig “vom sehen”. Trotz diesem Umstand stand das Thema des Projektes noch am selben Tag nach einer längeren Brainstorming Session fest.

In den ersten Wochen gab es einige Schwierigkeiten, hauptsächlich aufgrund von Defiziten bei den WBA- und MCI-Kentnissen beider Gruppenmitglieder. Durch eine gute Beratung seitens der Betreuer und eine Aufarbeitung der WBA- und MCI-Inhalte konnte jedoch trotzdem produktiv gearbeitet werden.

Die Artefakte wurden größtenteils gemeinsam in den Räumlichkeiten der TH erarbeitet. Schnell stellte sich heraus das die Erarbeitung der Artefakte kein linearer Prozess ist und Artefakte ständig überarbeitet werden müssen. Um nicht in Zeitnot zu geraten wurden in den Projektplan Puffer eingebaut, welche hauptsächlich zum iterieren von Artefakten allerdings auch zum aufholen von zeitlichen Defiziten genutzt werden konnten.

Die größte Schwierigkeit während des ersten Meilensteines stellte die Wahl des MCI-Vorgehensmodelles dar. Die Wahl viel regelmäßig auf unterschiedliche Vorgehensmodelle, jedoch konnte nach einigen Feedbackterminen und häufigem umwerfen der bisherigen Ansätze ein für das Projekt geeignetes Vorgehensmodell identifiziert werden.

Die Entwicklung des Rapid Prototypen wurde für die letzte Woche des Meilenstein 1 angesetzt und sollte hauptsächlich der Umsetzung der Proof of Concepts dienen. Daher wurde wenig Wert auf die Optik gelegt und nur die Funktionalitäten standen im

Vordergrund. Die Entwicklung dieser verlief im großen und ganzen ziemlich rund, da beide Entwickler schon vor dieser mit der Recherche begonnen haben.

Meilenstein 2: Dokumentation

Während des zweiten Meilensteines wurde mit dem Konzept als Grundlage eine Dokumentation erarbeitet. Diese Dokumentation unterteilt sich in einen MCI und einen WBA Teil. Zusätzlich wurden während dieser Zeit die restlichen PoC's durchgeführt. Da die gemeinsame Arbeit in den Räumlichkeiten der TH während des ersten Meilensteines zufriedenstellend war wurde sich dazu entschieden die Dokumentation ebenfalls gemeinsam in der TH zu erstellen.

Das Team entschied sich dafür als erstes den MCI Teil zu erarbeiten und daraufhin den WBA Teil zu bearbeiten, da dieser von den Erkenntnissen die während der MCI gewonnen wurden abhängt. Die restlichen PoC's sollten anschließend durchgeführt werden, da erwartet wurde das sich diese Aufgrund neuer Erkentnisse ändern würden.

Die Erarbeitung des MCI Teiles verlief ohne größere Schwierigkeiten. Der Usability Engineering Lifecycle konnte in der Vorgesehen Zeit ohne größere Verzögerungen durchgeführt werden. Wenn es notwendig erschien wurden durch das Team einige Änderungen an dem Lifecycle vorgenommen, wie etwa die Erstellung von HTA's. Nachträglich lässt sich sagen, dass das Team bereits zu früh mit zu detaillierten Prototypen gearbeitet hat, wodurch jedoch eine detaillierte Ausarbeitung des UI's vereinfacht wurde.

Die Erarbeitung des WBA Teiles verlief nicht so Reibungslos wie die des MCI Teiles. Dies lag Größtenteils daran das die benötigte Zeit stark unterschätzt wurde und somit zu wenig Zeit für diesen Teil in dem Projektplan vorgesehen war. Ein großer Fokus lag während dieser Zeit auf eine Darstellung der Anwendungslogik durch Pseudocode. Dies wurde als sehr wichtig eingeschätzt, da aufgrund der GTFS-Daten ein komplexer Algorithmus entwickelt werden musste der Fahrten zwischen zwei Bahnhöfen ermittelt. Außerdem musste ein System für das erweiterte Matching entwickelt werden. Um mehr Klarheit über die weitere Anwendungslogik zu erhalten wurde diese Ebenfalls in Pseudocode festgehalten.

Meilenstein 3: Implementation

Um den geplanten Workload bis zur Abgabe umzusetzen, hat sich die Gruppe im dritten Meilenstein dazu entschlossen die Arbeiten am Server und Client aufzuteilen, um so möglichst effizient Arbeiten zu können. Somit wurde Zeit gespart, da die Teammitglieder sich erstmal nur in einen Bereich einarbeiten mussten und somit zu Experten für ihren Teil wurden. Beide Teammitglieder konnten so außerdem zu Hause arbeiten und die Vorteile eines Desktoprechners und mehrerer Bildschirme genießen, wodurch deutlich effektiver programmiert werden konnte.

Während der Implementation wurde schnell deutlich dass einige Änderungen an den in der Dokumentation spezifizierten URI's vorgenommen werden musste, da entweder ein größerer Payload benötigt wurde, oder die URI's durch eine Umstrukturierung logischer wurden. Eine weitere große Änderung wurde an der Asynchronen Kommunikation vorgenommen. Ursprünglich war es geplant, den Client bei Chatnachrichten oder Suchagent über von ihm abonnierte Topics anzusprechen, jedoch wurde sich dafür entschieden den Client direkt über sein FCM-Token anzusprechen, da für den Chat immer nur der Chatpartner benachrichtigt werden sollte und der Suchagent so viele Parameter berücksichtigen musste, dass das direkte Ansprechen des Clients als sinnvoller erschien.

Das größte Problem während der Implementation des Servers war es einen performanten Fahrtenfindungsalgorithmus zu entwickeln. Der in der Dokumentation entwickelte Algorithmus wurde mit einer durchschnittlichen Laufzeit von etwa 10 Sekunden auf einem "leistungsstarken" Rechner als zu langsam eingeschätzt. Durch eine Umstrukturierung des Algorithmus konnte die durchschnittliche Laufzeit auf etwa 1.5 Sekunden reduziert werden. Ein weiteres Problem stellte das Einlesen der GTFS-Daten in die Datenbank dar. Das für den Rapid-Prototypen benutzte Script brauchte dafür mehrere Stunden. Abhilfe schaffte das Node Module "node-gtfs", welches ein solches Script bereitstellte. Dieses war zur Zeit des Rapid Prototypen nicht brauchbar, da der Speicher voll lief. Dies wurde durch ein Update behoben. Auf die Nutzung weiterer Funktionalitäten dieses Modules wurde verzichtet, da mit ihnen nicht so flexibel wie mit eigenem Code gearbeitet werden konnte und somit eine längere Laufzeit für den Fahrtenfindungsalgorithmus erreicht würde.

Ein weiteres Problem hat sich nach der Implementation von FCM im Teil des Chats ergeben. Ursprünglich war es nicht geplant Nachrichten eine eigene URI zu geben. Neue Nachrichten sollten über eine sogenannte sequence bezogen werden, welche einer inkrementelle Nummerierung der Nachrichten entspricht. Schreiben nun beide Teilnehmer zeitgleich, tritt jedoch eine Race Condition auf und einer der Teilnehmer erhält die von ihm verfasste Nachricht. Dies konnte durch die Implementation einer

URI für jede Nachricht und der im Payload der Notification mitgegebenen message_id behoben werden.

AutoComplete und Umkreissuche wurden, im Gegensatz zum Rapid Prototypen, komplett auf den Client umgelagert, was wiederum eine Benutzung dieser Features auch ohne eine intakte Internetverbindung begünstigt. Desweiteren können die Daten der Haltestellen als so statisch angesehen werden, dass diese nicht jedes mal erneut vom Server angefordert werden müssen. Diese werden bei der ersten Verwendung der Applikation vom Server bezogen und versioniert in der Lokalen Datenbank gesichert.

Es wurde während der Implementation eng mit Stakeholdern zusammengearbeitet. Diese Zusammenarbeit hat mehrere Probleme aufgezeigt, dass größte Problem war, dass nicht deutlich genug war an welchen Haltestellen der Reisepartner dazu- und aussteigt. Dieses Problem wurde durch einen neuen Screen, die detaillierte Fahrtenansicht, behoben. Er stellt die wichtigsten Informationen einer Fahrt schematisch dar.

Fazit

Im folgendem Fazit wird der Zielerreichungsgrad der Implementation evaluiert und diskutiert inwieweit die geplanten Ziele umgesetzt/erreicht wurden.

Die zu Beginn des Projektes verfassten tatkischen und operativen Ziele konnten alle Eingehalten werden. Dadurch das schon früh eng mit Stakeholdern zusammengearbeitet wurde entstand eine Dokumentation die eine effektive Lösung des Problems beschreibt.

Da die Implementierung der Applikation in einem kleinen Team, bestehend aus zwei Entwicklern, in einem Zeitfenster von etwa 4 Wochen als nicht realisierbar eingeschätzt wurde mussten einige Kompromisse hingehen der Funktionalitäten, die Implementiert werden, gemacht werden. Die Implementierung von Funktionalitäten die kein Alleinstellungsmerkmal darstellten, wie etwa die Ansicht des Chatverlaufs ohne aktive Internetverbindung, wurde niedriger priorisiert als Funktionalitäten die als essentiell für die Nutzung der Anwendung eingeschätzt wurden.

Der entstandene vertikale Prototyp ist durchaus für die Lösung des identifizierten Problems zu gebrauchen, allerdings könnte durch weitere Funktionalitäten eine höhere Gebrauchstauglichkeit erzielt werden. Zu diesen Funktionalitäten gehören unter anderem:

- Eintragen von Gruppenfahrten
- automatisches Eintragen von Fahrten die regelmäßig stattfinden
- die Ansicht des Chatverlaufs ohne aktive Internetverbindung

Inwieweit die strategischen Ziele erfüllt wurden kann sich erst entgültig zeigen nachdem das Projekt mit einer größeren Stakeholdergruppe getestet wurde. Das Entwicklerteam ist allerdings aufgrund der während der Implementierung gewonnen Eindrücke auch hier zuversichtlich das diese Ziele erfüllt wurden.