Technology
Arts Sciences
TH Köln

Prozessassessement & Fazit des Projekts *TravelCompats* für das Modul *Entwicklung interaktiver Systeme*

Betreuer

Prof. Dr. Gerhard Hartmann

Prof. Dr. Kristian Fischer

B.Sc. Robert Gabriel

Studierende

Nico Ferdinand	nico.ferdinand@smail.fh-koeln.de	11103027
Simon Porten	simon.porten@smail.fh-koeln.de	11103278

Einleitung

Dieses Dokument beschäftigt sich mit dem Projektverlauf des EIS-Projektes *TravelCompats* und reflektiert diesen kritisch. Es wird chronologisch die Arbeit zu den einzelnen Meilenstein reflektiert. Beginnend mit dem ersten Meilenstein hinweg zum letzten Meilenstein.

Meilenstein 1

Der erste Meilenstein beschäftigte sich mit der Identifizierung von Nutzerproblemen und der Konzeption einer Lösung dieser Probleme. Dazu sollte ein Projektplan erstellt werden, welcher die Aufgabenfragmente über die Meilensteine bis zum Beenden des Projekts auflisten sollte. Doch bevor sich dem gewidmet wurde, musste ein Expose für die Dozenten erstellt werden, welches aus den vier Punkten *Nutzungsproblem*, *Zielsetzung*, *verteilte Anwendungslogik* und *wirtschaftliche und gesellschaftliche Aspekte* bestehen sollte. Erst, wenn diese Punkte geklärt werden konnten und die Dozenten die Projektidee für in Ordnung befunden, konnte die Arbeit beginnen. Das Nutzungsproblem und wirtschaftliche sowie gesellschaftliche Aspekte waren schon vor dem EIS-Modul bekannt. Doch die Zielsetzung und vor allem die Anwendungslogik veränderten sich auch nach dem Konzept noch leicht. Denn erst während der praktischen Arbeit wurden Ressourcen wie Zeit und Aufwand realistisch einschätzbar. So reichte eine Verifikationsmethode nicht aus, um die Voraussetzungen der Anwendungslogik zu erfüllen.
Allgemein wurde zu Beginn der Arbeit an dem Projekt deutlich, dass die Erarbeitung der Fragmente keine lineare Arbeit ist, sondern immer evaluiert werden muss.

Auch der Projektplan war schwierig zu erstellen und besonders ihn fortlaufend auszufüllen. So war zur Zeit der Konzeption unbekannt, welche Schritte bis zum dritten Meilenstein erfüllt werden müssen. Diese Schritte mussten immer wieder bei Meetings mit den Dozenten erfragt werden, was dafür sorgte, dass der Projektplan seinen Sinn gar nicht mehr erfüllte und immer erst nach der Arbeit

als Kontrollplan genutzt wurde. Vielmehr dienten bei Entwicklung des Konzepts die Folien der Dozenten als Projektplan. Eine Vorlage als Musterprojektplan wäre für die Anfänge der Arbeit sinnvoll gewesen.
Die größte Schwierigkeit während des 1. Meilensteins war die Wahl eines Vorgehensmodells und sich daran zu halten. Dieses Projekt ist für die Studenten das erste Mal gewesen, dass nach einem Modell vorgegangen wurde. Und da die Erfahrung fehlte, wurde einfach ein Modell gewählt, das man in den Vorlesungen kennen gelernt hatte.

Im Rapid Prototyp wurden keine design-technischen Entscheidungen getroffen, sondern lediglich die wichtigsten Teilimplementation der PoCs integriert. Die Tatsache, dass die Voraussetzungen für dieses Modul bestimmen, dass der Client auf Java und der Server auf Node.js basieren soll, erforderte extrem viel Aufwand. Noch nie zuvor wurde in unserem Studium über die programmierte Kommunikation zwischen Node.js- und Java-Anwendungen diskutiert. So musste sich diese Fähigkeit in einem Selbststudium zu Hause selbst beigebracht werden. Ebenso erwies sich die erstmalige Nutzung von Tools, wie Android Studio, als komplex und forderte viel Zeit, die leider nicht in die Umsetzung des eigentlichen Projekts einfließen konnte. Wünschenswert wäre es gewesen, dass die vorherigen Semester des Studienganges, diese Dinge lehren.

Meilenstein 2:

Das Konzept des ersten Meilensteins dient beim zweiten Meilenstein als Grundlage der Dokumentation, welche erstellt und abgegeben werden musste. Die Dokumentation unterteilt sich in MCI- und WBA-Inhalte. Des Weiteren werden die PoCs weiter erarbeitet und nach der Dokumentation zum Code-Audit durchgeführt und implementiert. Die Anforderungsanalyse war ebenso ein sehr großer Teil der Dokumentation. Dort wurde eine Umfrage erstellt, User Profiles in einem Szenario aufgelistet und zwei Interviews durchgeführt, um die gesammelten Informationen über die Benutzergruppen zu validieren. Nach mehreren Meetings wurde das Architektur-Modell überarbeitet und an asynchrone Handlungen angepasst.
Der MCI-Teil wurde mit Hilfe von Literatur über das gewählte Vorgehensmodell erarbeitet. Der WBA-Teil wurde vorerst mit Pseudocodes dargestellt. Außerdem wurden vor der Implementierung mehrere Prototypen designed und ein Style-Guide erstellt, der bei dieser Aufgabe helfen sollte.

Die Implementierung beinhaltet die wichtigsten Punkte der PoCs, welche in einem Code-Audit vorgestellt wurden. Aufgrund der zeitintensiven Programmierung, wurde die Atmosphäre der CompyMarks bis zum Code-Audit nur aus Audio-Analysen und GPS-Daten definiert. Später könnten weitere Attribute wie App-Nutzung der Benutzer, Lage des Ortes, Audio des Ortes, Wetter, Freizeit-Art der Freizeitanlagen, Sportarten der Sport-Anlagen, Art und Weise der Nutzung einer Einrichtung, Art und Weise eines Geschäftes, Tourismus-Mehrwert, historischer Hintergrund, Bezeichnung der vorhandenen Natur und umliegende Routen hinzugefügt werden. Dies muss aber erst in Form einer Machbarkeitsstudie bis zur Projektabgabe erörtert werden. Zu Unterscheiden sind dabei Attribute die automatisch von anderen Systemen ausgelesen werden können und Attribute die von Nutzern eingegeben und übermittelt werden können.
Im Laufe des zweiten Meilensteins entstand eine Rohfassung der App, die zum Testen immer wieder auf ein ausgeliehenes Android Gerät aufgespielt wurde.

Meilenstein 3:

Nach dem Code-Audit wurde deutlich, dass das Objekt *Atmosphäre* aus weiteren Attributen bestehen muss, um die angezeigte Atmosphäre realistischer darzustellen. Bis zu diesem Zeitpunkt bestand das Objekt Atmosphäre lediglich aus ausgewerteten Audio-Dateien und Ortsangaben mit Hilfe von GPS-Informationen. Grund dafür war die Zeitknappheit bis zum Code-Audit und die fortgeschrittenen Programmierkenntnisse, welche nötig, aber noch nicht komplett erreicht sind.
Ein Brainstorming erzielte das Ergebnis, das aufzeigt, welche weiteren Attribute sehr gut geeignet sind. Die folgenden Informationen können von OpenStreetMap mit Hilfe von REST ermittelt werden. Zu diesem Zeitpunkt ist allerdings noch nicht geklärt, ob diese Attribute genauso in den Code implementiert werden - weitere Testphasen müssen die Machbarkeit verdeutlichen.

- **App-Nutzung.** An welchen Orten werden welche Apps auffallend oft benutzt? Wo sind Smartphones oder AR-Apps (siehe China) vielleicht sogar verboten? Nutzer die gerne bestimmte Apps nutzen, können so an bestimmten Orten Gleichgesinnte treffen - z.B. Myfitnesspal.
- **Lage.** Ist ein CompyMark in einer ländlichen, städtischen, bergischen, etc. Lage? Befindet sich das Etablissement an einer befahrenen Straße? Der Schlüssel “**landuse**” von OSM könnte sich dafür als nützlich erweisen. Er zeigt wo sich Rasenflächen befinden “landuse = grass” oder wo sich ein Industriegebiet befindet “landuse = industrial”
  Hinzu kommt der Schlüssel “**highway**” der anzeigt welche Straße vorhanden ist. Bsp.: “highway = pedestrian” ist ein Fußgängerplatz.
- **Audio.** Die durchschnittliche Lautstärke eines Ortes soll berechnet und angezeigt werden.
- **Wetter.** Von Nutzern sollen GPS-Informationen bezogen werden. Mit dessen Hilfe soll das Wetter angezeigt werden.

- **Freizeit-Art:** Der Schlüssel “leisure” ist ein Objekt-Attribut auf der OSM. Dieser zeigt an, welche Art von Freizeitgestaltung eine Lokalität ist. Z.B. “dance” ein Ort wo getanzt wird. “gaming_centre” eine Spielhalle für jedes Alter., etc.
- **Sport-Art:** Ist eine Einrichtung für sportliche Aktivitäten erbaut kann der Schlüssel “sport” detailliert in eine Beschreibung eingehen. “sport = darts”.
- **Nutzung:** Der Schlüssel “amenity” auf OSM besagt wie ein Gebäude oder eine Sache genutzt wird. Es kann beschreiben, dass ein Gebäude ein Restaurant ist “amenity = restaurant”. Weitere Beispiele: “amenity = college”, “amenity = drinking_water” oder “amenity = bicycle_parking”.
- **Geschäfte:** Der Schlüssel “shop” zeigt welche Art von Geschäft eine Einrichtung ist. Z.B.: “shop = bakery”, “shop = ice_cream”, etc.
- **Tourismus:** Der Schlüssel “toursim” zeigt an welche Art von Tourismus-Unterhaltung eine Einrichtung bietet. Z.B.: “tourism = artwork” oder “tourism = camp_site”, “tourism = attraction”.
- **Historischer Hintergrund:** Der Schlüssel “historic” zeigt an ob eine Sache, eine Einrichtung oder ein Gelände einen Historischen Hintergrund besitzt und eine Besichtigung wert ist. Bsp: “historic = castle” oder “historic = battlefield”.
- **Natur-Bezeichnung:** Der Schlüssel “natural” lässt die natürliche Umgebung erkennen. Er zeigt wo sich z.B. ein Wald befindet “natural = wood”, oder wo sich ein Strand befindet “natural = beach”, usw.
- **Routen:** Der Schlüssel “route” zeigt fortlaufende Routen. Also z.B. Zugrouten, Laufrouten oder Reitwege und vieles mehr. “route = horse”.

Bei Testarbeiten an Client und Server stellte es sich als sehr **schwierig** heraus, die oben aufgelisteten Ressourcen über die OSM-API abzufragen. Grund dafür ist, dass die Informationen nicht als JSON Object zu beziehen sind. Diese Tatsache könnte man umgehen, wenn der Client die Daten verbogen bezieht und in ein JSON Object umwandelt. Diese Alternative wäre allerdings äußerst unperformant und würde konkret bedeuten, dass ein Client im Hintergrund die

OSM Dienste laufen lässt, um diese zu sammeln, an den Server zu parsen, damit der Server diese zum Arbeiten verwenden kann und dann schließlich die Informationen an den Client zurücksendet. Da in der kurzen Implementationszeit so viele Schwierigkeiten mit Open Street Map zu Tage kommen, muss spontan auf Google Places zurückgegriffen werden. Dort sind ebenfalls sehr viele Informationen erhältlich. Lediglich das Premium Modell, welches im Gegenzug Geld verlangt und teilweise veraltete Datenbanken (Restaurants die nicht mehr existieren werden in Google Places noch angezeigt) sind Probleme beim API-Zugriff. Daher werden zum dritten Meilenstein Datenbanken zu Simulationszwecken selbst erstellt und mit Informationen gefüttert, die zur Berechnung einer Atmosphäre nötig sind. Bis zum jetzigen Zeitpunkt werden folgende Attribute benutzt um Empfehlungen auszugeben:

- **Geruch** eines Ortes.
- **Geschmack** des Essens z.B.
- **Freundlichkeit** der Menschen und des Personals
- **Geräusche** stammen aus Netatmos und eigenen Aufnahmen, um zu prüfen wann Richtwerte überstiegen werden.
- **Gesprochene Sprachen** an einem Ort und eines Landes werden berücksichtigt.
- **Wetter**. Informationen stammen von Open Weather Map.
- **Familientauglichkeit**
- **Textuelle Bewertungen** von anderen Nutzern über gelistete Orte in Google Places werden bei Empfehlungen mit angezeigt. Ein Keyword-Filter wird hierbei eingesetzt, um aus einem Fließtext wertvolle Informationen zu sammeln.

Diese Attribute werden in dezimalen Werten beschrieben - von 0 bis 5. 0 ist dabei ein Platzhalter, falls dieses Attribut gar nicht vertreten ist. 1 steht für den stärksten positiven Einfluss.

Geplant war im Konzept, dass Daten von anderen Apps gesammelt werden. Als Beispiel sollte MyFitnessPal genutzt werden. Schwierig an diesem Vorgehen ist

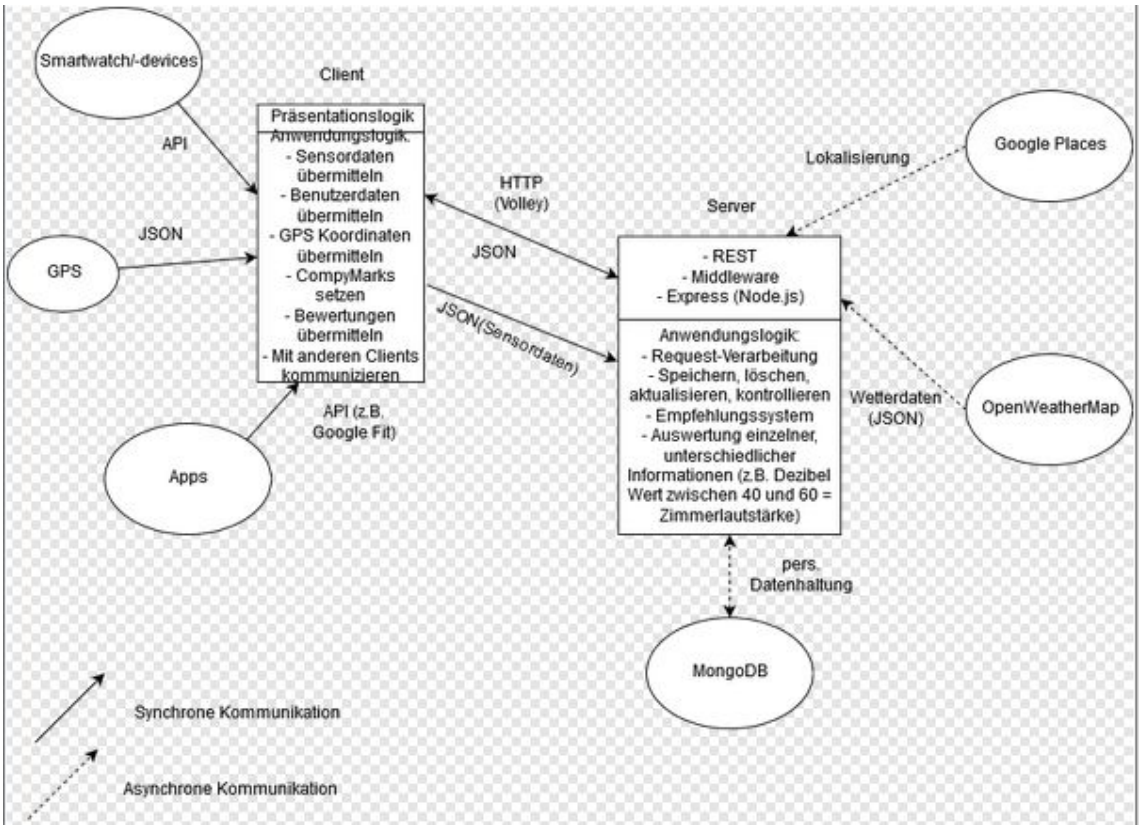
nun, dass dafür ein API Schlüssel bezogen werden muss, der bis zum Ende des Projekts nicht mehr zu erhalten ist, da die Vergabe sehr lange dauert. Alternativ wird nun die Fitness Applikation/ API von Google benutzt. Der Zugriff darauf ist möglich, setzt allerdings voraus, dass ein Client die passende Applikation auf seinem Gerät installiert hat.

```
LocationAtmosphere:{
  Geruch: [0-5],
  Geschmack: [0-5],
  Freundlichkeit: [0-5],
  Aktivitaeten: [(App-Analyse:) Pokemon Go, Fitness, etc.],
  Umgebungslautstärke: [0-5],
  Sprachen: [Deutsch, Englisch, Spanisch, Italienisch, Türkisch, ...]
  Wetter: ["Gebäude","Wetter betroffen"],
  Familientauglichkeit: [0-5],
  Sauberkeit: [0-5],
  Barrierefrei: [(Unterstützende Mittel)],
  Google Bewertung: [0-5],
}
```

Pseudocode Atmosphäre

Des Weiteren kam es immer öfter zu Schwierigkeiten, da in der Konzeptphase zu wenig Zeit in die Implementierung floss. Einige wenige Tage/Wochen mehr, hätten den letzten Meilenstein deutlich erleichtert. Anfangs war geplant, dass ein universal Werkzeug erstellt wird, wo Benutzer Empfehlungen für Reiseziele, Orte, Aktivitäten u.v.m. erhalten. Nach und nach wurde das Spektrum der Empfehlung verkleinert, um realisiert werden zu können. In der zweiten Phase wurde beschrieben, dass ein Radius von 50km gewählt wird, um den Benutzer, um Empfehlungen auszusprechen. Allerdings bemerkten wir in der dritten Phase, dass selbst 500m um die Mensa TH Köln (in Gummersbach) 17 Restaurants liegen. Deshalb wurde die Realisierung weiter verkleinert, um auch Testdaten für die 17 Restaurants zu erstellen. Während der Erstellung der Testdaten kam es erneut zu Schwierigkeiten, da man nicht je Restaurant nur eine Bewertung braucht, sondern möglichst “viele”, um “echte” Empfehlungen abzugeben. Deshalb haben wir uns auf 5 Restaurants um die Mensa TH Köln (inkl. der Mensa) spezialisiert und jeweils mindestens 5 Testdaten hinzugefügt, ausgewertet und verarbeitet.

Wie in der Code Implementation besprochen wurde findet der Datenaustausch zwischen Client und Server über HTTP synchron statt und nicht asynchron wie zunächst angenommen. Grund für die Annahme war bei den Tests die Beobachtung, dass der Server immer noch gesendete Informationen des Clients bearbeitete und zurücksenden wollte, obwohl die Client nicht mehr verbunden war. Lediglich die Informationen, die über die APIs an den Server laufen (Google Places und OWM) asynchron.



Architekturdiagramm Stand 10.07.2017

Fazit

Ziel des Fazits ist es den Zielerreichungsgrad offen zu legen und zu diskutieren. Wie weit wurden die anfangs definierten Ziele erreicht und Implementierungsideen umgesetzt?

Die operativen Ziele wurden schon zu Beginn der Arbeit aufgenommen und bearbeitet. Der Zielerreichungsgrad füllt in Anbetracht der operativen Ziele zunehmend.
Bei Erfüllung der taktischen Ziele werden Probleme während der Arbeitszeit ersichtlich. So wurde der Projektplan zu den späteren Momenten des Projekts nicht immer eingehalten, bzw. angepasst, da Projektphasen spontan verändert wurden. Grund dafür sind zum Beispiel strenge Zeitgrenzen und fehlende Expertise im Umgang mit Tools und Programmiersprachen. Außerdem war zu Beginn unklar wie ein Projektplan zu erstellen und führen ist. Auch die Evaluationsschritte des Vorgehensmodell verbrauchten mehr Zeit, als anfangs eingeplant wurde. Im Gegensatz dazu wurde stetig an PoCs gearbeitet, diese umgesetzt und getestet. Sobald sich ein Risiko erfüllte wurde mit Hilfe der Fallback-Cases eine Alternative gefunden, um das Problem zu beheben. Auch der Style Guide half den Prototypen möglichst effizient zu erstellen.
Die bis Dato erfüllten Ziele, verdeutlichen den Weg zu den strategischen Zielen. So wurde stetig die Analyse- und Empfehlungsfunktion verbessert. Diese besagt, dass sensorische Daten, Daten aus App-Schnittstellen (nun z.B. Google Fitness App) und Daten von Webservices erfasst werden und in die analysierende Anwendungslogik mit einfließen. Ebenso lief die Teilimplementierung von CompyMarks gut ab.
Leider fehlt das Gesamtpaket, welches das System ausmachen würde. Ein soziales Netzwerk als digitalen Sammelplatz, eine Separierung zwischen normalen und wirtschaftlichen Nutzers und ein einzigartiges Empfehlungssystem, das sich von der Konkurrenz abheben würde. Dieses Vorhaben stellt sich als EIS-Projekt zwar sehr gut dar, ist aber für die Fähigkeiten von nur zwei Teammitgliedern zu überwältigend. So mussten folglich alle weniger relevanten Alleinstellungsmerkmale und PoCs ignoriert, bzw. weniger priorisiert werden.

Die Zusammenarbeit mit Stakeholdern lief sehr gut ab und ermöglichte durch Umfragen und Interviews eine solide Datenvalidierung. Durch einen vertikalen Prototypen konnte eine höhere Gebrauchstauglichkeit erzielt werden.

Inwieweit das restliche Vorhaben und die offenen Ziele erfüllt werden können, kann erst nach dem Projekt an einer umfangreichen Stakeholdergruppe getestet werden.