Selbstreflexion

# Überfülltes Projekt

Wir haben uns viel zu viel vorgenommen und auch nicht auf die Hinweise der Referenten reagiert, dass dies wahrscheinlich der Fall ist. Wir hätten das Projekt kleiner halten sollen, mit Möglichkeiten zu Erweiterungen.

# Zeitplan

Aufgrund des überfüllten Projektes wurde für die jeweiligen Teilaufgaben im Zeitplan viel zu wenig Zeit eingeplant. Jedoch selbst als wir nach der hälfte der Projektzeit den Zeitplan angepasst haben um das Problem zu lösen planten wir immer noch zu wenig Zeit ein, so dass der Zeitplan später erneut angepasst werden musste. Wir hatten also nur teilweise aus den Fehlern gelernt. Ein Grund dafür war auch, dass es sich schlecht anfühlte Programmteile welche man eigentlich umsetzen wollte herauszustreichen.

# Performantes Programmieren

Wir hatten in unserer Programmierkarriere noch nie mit dem Blick auf die Performance des Programms programmiert. Somit war es schwer für uns performantes Programmieren umzusetzen und wir bauten viele unnötige Schleifen usw. ein. Dies führte dazu, dass wir das Performance-Ziel nicht erreichen konnten und viel Zeit in die Programmoptimierung investieren mussten, welche nicht eingeplant war. Wir hätten uns im vorherein besser informieren sollen und unsere Erwartungen herunterschrauben sollen. Mit unserer Erfahrung konnten wir nicht erwarten die im Paper angegebenen Zeiten zu erreichen.

# Programmcode Kommentieren

Ein Fehler von uns war, dass wir den Code nicht direkt kommentiert haben. Als wir darauf hingewiesen wurden haben wir den damaligen Zustand des Codes kommentiert. Jedoch kommentierten wir zukünftigen Code nicht direkt und mussten so am Schluss einen Grossteil nachkommentieren. Dies führte zu einigen Programmteilen welche wir zeitaufwendig neu interpretieren mussten, da wir den sinn des Teils nicht mehr wussten. Für zukünftige Projekte willen wir spezifischer kommentieren. Kommentare für Funktionen mit offensichtlichen Aufgaben sind die Kommentare nur eine Fleissarbeit und können weggelassen werden. Dafür müssen komplexe Funktionen besser kommentiert werden.

# Vorgreifendes Programmieren

Oftmals wichen wir von unserem Plan ab und programmierten Funktionen die erst viel später geplant waren. Dies führte auch dazu, dass wir Funktionen programmierten welche schlussendlich gar nicht gebraucht wurden. Ein Beispiele dafür ist, dass der JourneyToTripPlanConverter eine komplette Implementation für verschiedene Formen der Fusswege sowie deren Geometrie besitzt, obwohl diese schlussendlich aus dem Scope gefallen sind. Ein weiteres Beispiel ist unsere Implementierung der ServiceDates von spezifischen Trips welche im TimeTableBuilder berechnet werden, jedoch im Algorithmus nicht beachtet werden. Dies führte dazu, dass der Programmcode einige tote Programmteile besitzt und wir einige Zeit verschwendeten.

# Überstürztes Programmieren

Wir programmierten oft nach besten Wissen und Gewissen ins Blaue hinein ohne uns vollständig zu informieren. Dies führte dazu, dass wir einige unkonventionelle Lösungsvarianten wählten welche später Aufgrund von Funktionsschwierigkeiten revidiert werden mussten. Ein Beispiel dafür ist unsere Implementation der Serialisierung. Wir serialisierten unser TimeTable-Objekt mithilfe von Jackson2 in mehreren Text-Files. Dies führte zu Problemen bei der Deserialisierung, da einige Referenzen nicht mehr vorhanden waren. Später änderten wir auf Hinweis der Referenten hin die Serialisierung, so dass das komplette TimeTable-Objekt einfach in einem SerializedObjectFile gespeichert wurde. Dies ist der Standard und stellte sich als viel einfacher heraus als alles was wir versucht haben.