Softwareunterstützung für ein Museum

Visualisierungen: Wegrekonstruktionen

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2009

Autoren:	Stefan Züger, Tobias Zürcher
Betreuer:	Prof. Dr. Lothar Müller
Projektpartner:	Albis Technologies Zürich, Naturama Aarau

Abstract

Ausgangslage

In diversen Vorgängerarbeiten wurden bereits verschiedene Softwareunterstützungssysteme für Museen entwickelt, mit deren Hilfe die zurückgelegten Wege der Besucher analysiert werden können. Die Positionsdaten der Besucher werden dabei mittels RFID erfasst und in einer zentralen Datenbank abgespeichert. Die bestehenden Anwendungen wiesen jedoch noch einige Mängel und fehlende Visualisierungen auf, was zugleich die Aufgabenstellung für diese Studienarbeit lieferte.

Resultate

Auf der Basis der bereits umgesetzten Arbeiten wurden in dieser Studienarbeit alle bestehenden Wegvisualisierungen komplett neu implementiert und die vorhandenen Mängel beseitigt. Dies ermöglicht animierte Bewegungsanalysen für mehrere Personen oder ganze Gruppen durchzuführen. Des Weiteren wurde die fehlende Unterstützung von mehreren Stockwerken hinzugefügt, so dass nun Wegrekonstruktionen über beliebig viele Etagen durchgeführt und visualisiert werden können. Neben den funktionalen Erweiterungen an der Software ist der ganze Code und der Wegalgorithmus neu entworfen und implementiert worden. Dadurch konnte beispielsweise ein A*-Wegalgorithmus realisiert werden, der bereits kleinere Wege über 100 Mal schneller berechnen kann als die frühere Version.

Technologien

Die ganze Software wurde wie alle Vorgängerarbeiten in Java geschrieben, wobei das GUI in Swing und alle Visualisierungen mit der Java 2D API umgesetzt wurden. Die Daten der Museen und aller Besuchererfassungen werden in einer MySQL Datenbank abgespeichert.

Inhaltsverzeichnis

Teil 1: Allgemeine Informationen

1 Dokumenteninformationen

1.1 Zweck

Dieser technische Bericht beinhaltet oder referenziert die komplette Dokumentation, die in der Studienarbeit *„Softwareunterstützung für ein Museum - Visualisierungen: Wegrekonstruktionen“* im Herbstsemester 2009 während 14 Wochen an der Hochschule für Technik in Rapperswil erstellt wurde.

1.2 Gültigkeitsbereich

Dieses Dokument hat Gültigkeit während der gesamten Projektdauer (14.09.2009 – 18.12.2009) und wird während dieser Zeit laufend aktualisiert. (siehe Kapitel *1.4 Änderungsgeschichte und Reviews*).

1.3 Übersicht der Teile

Die vorliegende Arbeit wird in mehrere Teile gegliedert, worüber die nachfolgende Tabelle eine kurze Übersicht liefert.

Teil	Titel	Beschreibung	Seite
1	Allgemeine Informationen	Allgemeine Informationen zum gesamten Dokument sowie die Änderungsgeschichte	7
2	Management Summary	Kurze Zusammenfassung über die vorliegende Arbeit	10
3	Projekt Management	Organisatorisches zum gesamten Projekt	13
4	Sprints	Dokumentation der Ergebnisse der einzelnen Iterationen des Projekts	29
5	Schlussfolgerungen	Abschliessende Schlussfolgerungen und Zusammenfassung über das gesamte Projekt	86
6	Persönliche Berichte	Persönliche Erfahrungsberichte und Reflexionen der Autoren zur gesamten Studienarbeit	88
7	Anhang	Aufgabenstellung im Original, Glossar, Verzeichnisse und Erklärung über die eigenständige Arbeit	97

Tabelle 1: Übersicht der Teile des Berichts

1.4 Änderungsgeschichte und Reviews

Revision

Version	Status	Datum	Beschreibung/Änderung	Autor
0.1.0	In Bearbeitung	17.09.2009	Formatvorlage erstellt	TZ
0.1.1	In Bearbeitung	22.09.2009	Kapitel Management Summary hinzugefügt	SZ
0.1.2	In Bearbeitung	23.09.2009	Kapitel Qualitätsmassnahmen hinzugefügt	SZ
0.1.3	In Bearbeitung	23.09.2009	Qualitätsmassnahmen überarbeitet	TZ
0.1.4	In Bearbeitung	26.09.2009	Kapitel Projekt-Management überarbeitet	SZ
0.1.5	In Bearbeitung	06.10.2009	Analyse Sprint 1 dokumentiert	TZ
0.1.6	In Bearbeitung	07.10.2009	Implementation Sprint 1 dokumentiert	TZ
0.1.7	In Bearbeitung	11.10.2009	Analyse Sprint 1 dokumentiert	TZ
0.2.0	In Bearbeitung	12.10.2009	Implementation Sprint 1 dokumentiert	TZ
0.2.1	In Bearbeitung	13.10.2009	Analyse und Design zur US 25 dokumentiert	SZ
0.2.2	In Bearbeitung	15.10.2009	Implementierung, Systemtest und Fazit der User Story 25 dokumentiert	TZ
0.2.3	In Bearbeitung	18.10.2009	Kapitel Analyse und Design US 26 hinzugefügt	SZ
0.2.4	In Bearbeitung	20.10.2009	Implementierung und Tests der US 26	SZ
0.2.5	In Bearbeitung	21.10.2009	Fazit US 26 und Sprint 2 Retrospektive	TZ
0.2.6	In Bearbeitung	26.10.2009	Externes Design US 23 dokumentiert	SZ
0.2.7	In Bearbeitung	29.10.2009	Implementierung US 23 dokumentiert	TZ
0.2.8	In Bearbeitung	29.10.2009	Systemtest und Fazit US 23 dokumentiert	TZ
0.2.9	In Bearbeitung	01.11.2009	Analyse und Design US 56 dokumentiert	SZ
0.2.10	In Bearbeitung	04.11.2009	Dokumentation US 27	TZ
0.3.0	In Bearbeitung	05.11.2009	Sprint 3 Retrospektive hinzugefügt	SZ
0.3.1	In Bearbeitung	06.11.2009	Zwischenbericht geschrieben	TZ
0.3.2	In Bearbeitung	08.11.2009	Zwischenbericht geschrieben	SZ
0.3.3	In Bearbeitung	10.11.2009	Analyse US 28 dokumentiert	TZ
0.3.4	In Bearbeitung	14.11.2009	Implementierung US 28 dokumentiert	SZ
0.3.5	In Bearbeitung	14.11.2009	Kapitel Test und Fazit der US 28 hinzugefügt	TZ
0.3.6	In Bearbeitung	17.11.2009	Analyse der US 29 dokumentiert	TZ
0.4.0	In Bearbeitung	19.11.2009	Sprint 4 Retrospektive hinzugefügt	SZ

0.4.1	In Bearbeitung	26.11.2009	Implementierung der TreeTable und Farbauswahl dokumentiert	SZ
0.4.2	In Bearbeitung	26.11.2009	Implementierung des Besucher Auswahl Dialogs plus Kapitel Tests dokumentiert	TZ
0.4.3	In Bearbeitung	01.12.2009	Dokumentation Analyse / Design zur US 30	SZ
0.4.4	In Bearbeitung	03.12.2009	Dokumentation der Implementierung zur US 30	SZ
0.5.0	In Bearbeitung	03.12.2009	Fazit US 30 und Sprint 5 Retrospektive hinzugefügt	TZ
0.5.1	In Bearbeitung	13.12.2009	Dokumentation der erweiterten Analyse und Implementierung der US 30	SZ
0.5.2	In Bearbeitung	13.12.2009	Systemtests und Fazit US 30 dokumentiert	TZ
0.6.0	In Bearbeitung	15.12.2009	Sprint 6 Retrospektive dokumentiert	SZ
0.6.1	In Bearbeitung	16.12.2009	Kapitel Zeitplanung und Zeitauswertungen hinzugefügt	TZ
0.6.2	In Bearbeitung	16.12.2009	Beurteilung der Resultate und Ausblick dokumentiert	SZ
0.6.3	In Bearbeitung	17.12.2009	Kapitel Zusammenfassung geschrieben	TZ
0.6.4	In Bearbeitung	17.12.2009	Scrum und XP Reflexionen geschrieben	SZ
0.6.5	In Bearbeitung	17.12.2009	Glossar hinzugefügt	TZ
0.6.6	In Bearbeitung	17.12.2009	Persönlicher Schlussbericht geschrieben	SZ
0.6.7	In Bearbeitung	17.12.2009	Persönliche Schlussbericht geschrieben	TZ
1.0	Veröffentlicht	18.12.2009	Dokument veröffentlicht	SZ/TZ

Tabelle 2: Änderungsgeschichte

Review

Datum	Kommentar	Revisor
08.10.2009	Review der Sprint 1 Dokumentation	TZ
21.10.2009	Review der Sprint 2 Dokumentation	SZ
05.11.2009	Review der Sprint 3 Dokumentation	TZ
19.11.2009	Review der Sprint 4 Dokumentation	SZ
03.12.2009	Review der Sprint 5 Dokumentation	TZ
14.12.2009	Review der Sprint 6 Dokumentation	SZ
17.12.2009	Review des gesamten Dokuments vor Druck	TZ/SZ

Tabelle 3: Reviews

Teil 2: Management Summary

2 Management Summary

2.1 Ausgangslage

In mehreren bereits vergangenen Studien- und Bachelorarbeiten wurden Prototypen von Softwareunterstützungen für Museen entwickelt. Mit dessen Hilfe lassen sich detaillierte Analysen über das Besucherverhalten machen. Welche Museumsräume werden am meisten besucht? Was sehen sich die Besucher wie lange an? Welchen Weg nehmen die Besucher durch das Museum? All diese Fragen lassen sich mithilfe der erstellten Prototypen basierend auf einer RFID Lokalisierung beantworten. Im Frühjahrssemester 2009 wurde ein solches Softwaresystem in einem Feldtest im Naturama in Aarau eingesetzt und ausgiebig getestet.

naturama
das neue Aargauer Naturmuseum

Abbildung 1: Naturama in Aarau

Die bereits erstellten Prototypen und vor allem der Feldtest im Naturama haben einige Mängel der bestehenden bzw. fehelenden Visualisierungen aufgezeigt.

Fehlende Visualisierungen sind:

- Bewegungsanalyse für mehrere Personen

Mängel bei den bestehenden Visualisierungen sind:

- Animation bei der Bewegungsverfolgung (Voraussetzung für neue Visualisierung)
- Realistischere Empfangsbereiche
- Bessere Unterstützung für mehrere Stockwerke mit Übergängen zwischen diesen

Für alle Visualisierungen liegen umfangreiche Daten aus dem Feldtest im Naturama vor. Während diesem Feldtest wurden Bewegungsanalysen von über 500 Personen aufgezeichnet.

Anbei ist ein Screenshot der bestehenden Visualisierungssoftware, welche in einer vergangenen Bachelorarbeit bereits umgesetzt wurde

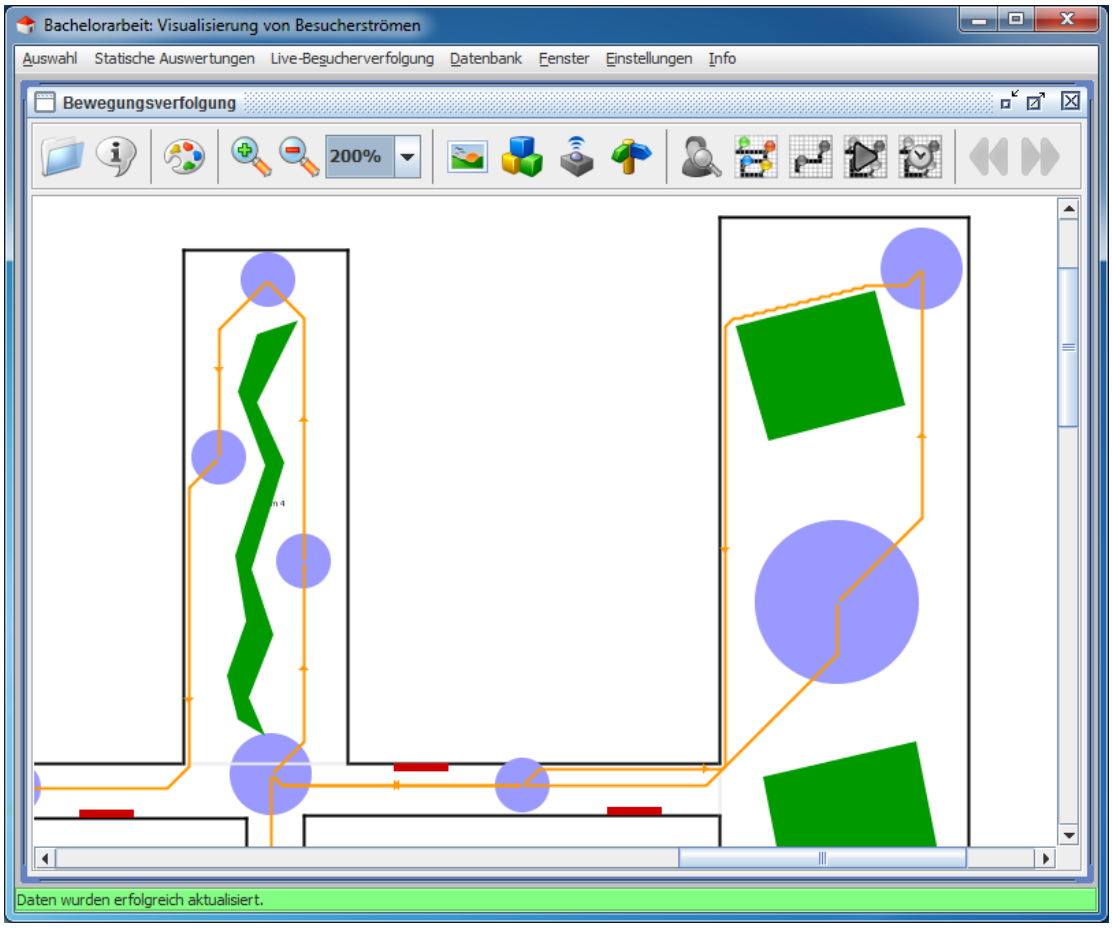

Abbildung 2: Screenshot der bestehenden Visualisierungssoftware

2.2 Aufgabenstellung

Basierend auf all den vergangenen Arbeiten hat sich Prof. Dr. Lothar Müller zum Ziel gesetzt, innerhalb eines Semesters aus all den vorhandenen Prototypen und Analysen ein marktreifes Endprodukt zu erstellen. Das Projekt wird durch zwei Studienarbeiten, einer Masterarbeit und Mitarbeit eines IFS Assistenten realisiert.

Das vorliegende Teilprojekt realisiert die Visualisierungen zur Wegrekonstruktion in verschiedensten Darstellungsarten. Dabei handelt es sich beispielsweise um folgende Aufgaben:

- Weg einer Person in verschiedenen Darstellungen visualisieren.
- Wege mehrerer Personen bzw. Gruppen in verschiedenen Darstellungen visualisieren.

Die konkret zu realisierenden Funktionen werden im Laufe des Projekts iterativ festgelegt und jeweils in den Projektsitzungen abgesprochen.

Eine Kopie der originalen Aufgabenstellung ist im Anhang im Kapitel *21 Aufgabenstellung*.

2.3 Vorgehen

Das Projekt wurde in mehrere Teilprojekte aufgeteilt. Das Team setzte sich aus der Masterarbeit von Thomas Kälin als Scrum Master, zwei Studienarbeitsteams à je zwei Entwickler und Michael Rüegg auch als Entwickler zusammen. Das gesamte Projekt wurde mithilfe des Vorgehensmodells Scrum umgesetzt (siehe Kapitel *4.1 Scrum*).

Der Product Owner (Prof. Dr. Lothar Müller) definierte dabei die Anforderungen, welche im Product Backlog festgehalten wurden. Beim Sprint Planning Meeting bestimmten die Entwickler regelmässig, wie viele User Storys aus dem Product Backlog im folgenden Sprint (jeweils 2 Wochen) erledigt werden können. Am Ende jedes Sprints wurde dem Product Owner eine nutzbare Software präsentiert, zu der er Feedback abgab. Während der Entwicklungsphase trafen sich alle Entwickler beim täglichen Daily Stand-Up Meeting, bei dem innert 15 Minuten jeder über seine Tasks berichtete und allfällige Probleme ansprach. Das gesamte Projekt umfasste 6 Sprints.

2.4 Ergebnisse

Das Ergebnis sind die Wegvisualisierungen innerhalb der Visualisierungs Software, in der zwischen drei verschiedenen Modi gewechselt werden kann:

Kompletter Weg
Der ganze Weg eines Besuchers wird gezeichnet.

Wegetappen
Es kann durch die einzelnen Wegetappen navigiert werden.

Animierte Wege
Der Aufenthalt von Besuchern und Gruppen wird animiert dargestellt.

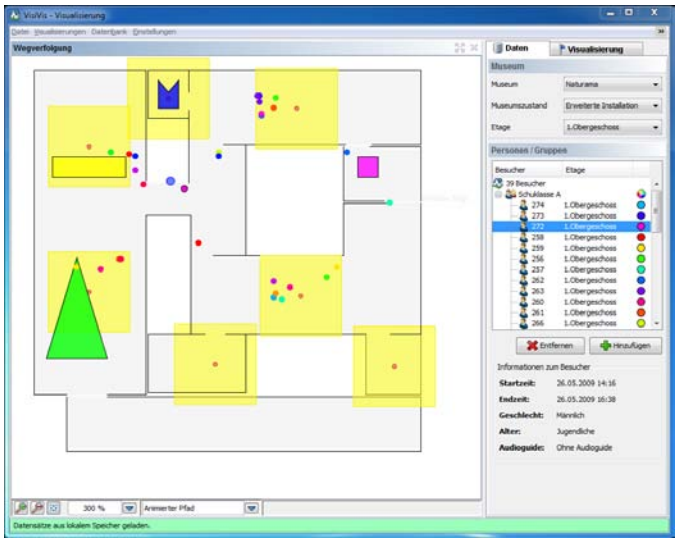

Abbildung 3: Screenshot der Wegvisualisierung

Des Weiteren wurde der Wegfindungsalgorithmus optimiert, so dass nun realistischere Wege berechnet werden. Im Animationsmodus werden unterschiedliche Wege berechnet, so dass zum Beispiel eine Schulklasse animiert dargestellt werden kann und deren Aufenthaltsorte als „Punktwolken“ verfolgt werden können. Zusätzlich wurde die Performance des Wegberechnungsalgorithmus stark verbessert, wobei dieser bereits kleinere Wege über 100 Mal schneller berechnen kann.

2.5 Ausblick

Die implementierte Software soll in einem weiteren Feldtest nochmals ausgiebig getestet und erweitert werden, wofür bereits eine weitere Bachelorarbeit definiert wurde. Die Firma Albis Technologies, von welcher die RFID Hardware bezogen wurde, kündete kürzlich eine neue Technologie an, bei der mithilfe von Triangulation der RFID Signalen die Position eines Besuchers genau bestimmt werden kann.

Teil 3: Projekt Management

3 Projektorganisation

3.1 Autoren

Diese Semesterarbeit ist lediglich ein Teil des Projektes VisiVis. Folgend die beiden Autoren dieser Semesterarbeit:

Abbildung 4: Stefan Züger

Stefan Züger
szueger@hsr.ch

Abbildung 5: Tobias Zürcher

Tobias Zürcher
tzuerche@hsr.ch

3.2 Projektteam

Das komplette Projekt VisiVis umfasst eine Masterarbeit und zwei Semesterarbeiten.

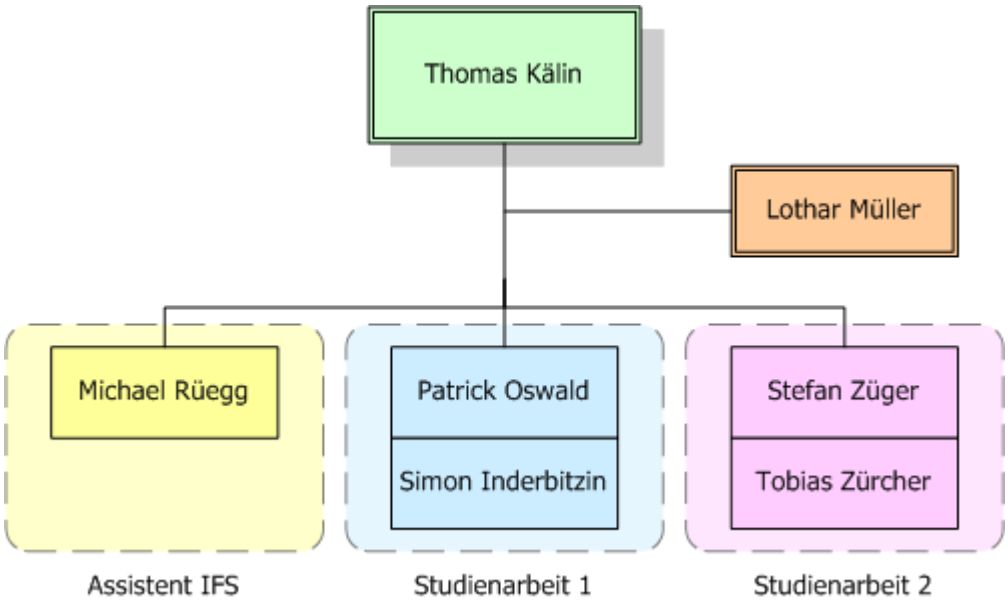

Abbildung 6: Organigramm des kompletten Projektteam

Thomas Kälin (tkaelin@hsr.ch) übernimmt mit seiner Masterarbeit die Projektleitung und ist somit der Scrum-Master. Genauere Details zur Funktion eines Scrum Masters sind im Kapitel *4.1 Scrum* dokumentiert.

Prof. Dr. Lothar Müller (lmueller@hsr.ch) definierte die Aufgabenstellung und übernimmt die Rolle als Product Owner. Er betreut alle drei Projektarbeiten.

Michael Rüegg ist Assistent beim Institut für Software (IFS) und unterstützt Thomas Kälin bei der Entwicklung.

Patrick Oswald und Simon Inderbitzin arbeiten an einer eigenen Semesterarbeit, bei der es primär um die Visualisierung verschiedener Statistiken geht.

3.3 Sitzungstermine

Folgende Planungsmeetings sind bereits zu Beginn des Projekts festgelegt worden:

Datum	Zeit	Raum	Kommentar
15.09.2009	10:00-12:00	1.111	Schulung des Teams / Kickoff
24.09.2009	13:00-16:30	6.111	Erstellung Product Backlog, Sprint Planning für Sprint #1
08.10.2009	13:00-16:30	6.111	Sprint Review für Sprint #1, anschliessend Sprint Planning für Sprint #2
21.10.2009	13:00-16:30	1.111	Sprint Review für Sprint #2, anschliessend Sprint Planning für Sprint #3
05.11.2009	13:00-16:30	6.111	Sprint Review für Sprint #3, anschliessend Sprint Planning für Sprint #4
19.11.2009	13:00-16:30	6.111	Sprint Review für Sprint #4, anschliessend Sprint Planning für Sprint #5
03.12.2009	13:00-16:30	6.111	Sprint Review für Sprint #5, anschliessend Sprint Planning für Sprint #6
17.12.2009	13:00-14:30	6.111	Sprint Review für Sprint #6

Tabelle 4: Sitzungstermine für Planungsmeetings

Neben den Planungsmeetings gibt es in der Entwicklungsphase sogenannte Scrum Daily Meetings (siehe Kapitel *4.1.2 Scrum Tätigkeiten*). Diese Meetings sind in jeder Woche wie folgt geplant:

Tag	Zeit	Ort	Wer?
Montag	09:00	IFS Pausenraum	Kälin, Rüegg
Dienstag	09:00	IFS Pausenraum	Kälin, Oswald, Inderbitzin, Züger, Zürcher
Mittwoch	13:00	IFS Pausenraum	Kälin, Rüegg, Oswald, Inderbitzin, Züger, Zürcher
Donnerstag	13:00	IFS Pausenraum	Kälin, Oswald, Inderbitzin, Züger, Zürcher

Tabelle 5: Termine für Daily Scrum Meetings

3.4 Sitzungsprotokolle

Die Sitzungsprotokolle zu allen relevanten Meetings befinden sich im Ordner `11 Sitzungsprotokolle`.

4 Vorgehensmodelle

Folgendes Kapitel liefert eine kurze Übersicht über die Vorgehensmodelle Scrum und XP. Die Kapitel über Scrum sind eine Kurzzusammenfassung der detaillierten Ausführungen der Webseite Scrum-Master.de *[URL10]*.

4.1 Scrum

Scrum (engl. das Gedränge) ist ein Vorgehensmodell, welches beim Entwickeln von Software im Rahmen agiler Softwareentwicklung hilfreich ist. Die Teammitglieder organisieren sich weitgehend selbst und bestimmen auch die eingesetzten Software-Entwicklungswerkzeuge und –Methode. Grundsätzlich verfolgt Scrum das Ziel, möglichst schnell eine ausführbare Software zu erhalten.

Scrum enthält Empfehlungen für Rollen, Tätigkeiten (Meetings) und Artefakte, welche in den folgenden Unterkapiteln kurz beschrieben werden.

4.1.1 Scrum Rollen

Rolle	Beschreibung
Product Owner	• Definiert Anforderungen als „User Stories“ • Vergibt Prioritäten (→ wählt Features aus)
Scrum Master	• Leitet Entwicklung und Meetings • Kontrolliert Qualität und Fortschritt
Scrum Team	• Optimalerweise 5-7 Personen • Interdisziplinär • Schätzt Entwicklungsaufwand • Organisiert sich selbständig

Tabelle 6: Scrum Rollen

4.1.2 Scrum Tätigkeiten

Tätigkeit	Beschreibung
Sprint Planning	Jeder Sprint beginnt mit dem Sprint Planning, bei dem das Arbeitspaket des Scrum-Teams für den kommenden Sprint (Sprint Backlog) geschnürt wird. Vorgehen: 1. Product Owner überarbeitet Product Backlog 2. Team schätzt Aufwand für User Stories 3. Team teilt User Stories in Tasks auf → Sprint Backlog

Daily Meeting	Das kurz (15 Minuten) zu haltende Daily Meeting findet – wie der Name schon sagt – an jedem Arbeitstag im Stehen statt und dient dem Team dazu, sich abzustimmen und gegenseitig zu informieren. Der Scrum Master nimmt teil, notiert sich genannte Impediments und greift nur moderierend ein, wenn es unbedingt nötig ist. Der Scrum Owner kann nach Möglichkeit teilnehmen, um auf dem neusten Stand zu bleiben und bei Bedarf Fragen zu beantworten. Das Scrum Team berichtet sich gegenseitig über folgendes: • Was habe ich seit dem letzten Daily Scrum getan? • Was plane ich, bis zum nächsten Daily Scrum zu tun? • Was hat mich bei der Arbeit behindert (Impediments)?
Sprint Review	Am Ende jedes Sprints präsentiert das Team dem Product Owner und allen interessierten Stakeholders das Ergebnis seiner Arbeit live am System und sammelt Feedback ein. Wichtige Punkte: • Das Team selbst präsentiert live am System, was es innerhalb des Sprints erreicht hat. • Keine Dummies, kein Power Point! Nur fertige Produktfunktionalität darf vorgeführt werden. • Auf Basis des Gezeigten entscheidet später der Product Owner, ob das Inkrement produktiv gesetzt oder weiter entwickelt werden soll. Diese Möglichkeit hat er nach jedem Sprint.
Sprint Retrospective	Das Sprint Retrospective folgt im Anschluss an das Sprint Review. Das Team diskutiert rückblickend auf den soeben zu Ende gegangenen Sprint und überlegt sich was weshalb gut oder schlecht gelaufen ist, und wie man den nächsten Sprint produktiver und/oder angenehmer gestalten kann.

Tabelle 7: Scrum Tätigkeiten

4.1.3 Scrum Artefakte

Artefakt	Beschreibung
Product Backlog	Das Product Backlog enthält alle Features des zu entwickelnden Produkts in Form von User Storys. Vor jedem Sprint werden die einzelnen User Storys neu bewertet und priorisiert.
Sprint Backlog	Das Sprintbacklog enthält alle aus dem Product Backlog abgeleiteten Aufgaben, welche in einem Sprint bearbeitet werden. Eine Aufgabe sollte nicht länger als 16 Stunden dauern. Bei der Planung werden nur so viele Aufgaben geplant wie einem Team an Kapazität zur Verfügung steht.
Sprint Burndown Chart	Der Burndown Chart stellt den Fortschritt des aktuellen Sprints dar. Die Kurve sollte am Ende des Sprints auf 0 sein. Die Tendenz der Kurve soll zeigen, ob der geschätzte Aufwand bis Ende Sprint erledigt werden kann.

Tabelle 8: Scrum Artefakte

Folgende Abbildung zeigt, wie die erwähnten Scrum Tätigkeiten und Artefakte zusammenhängen:

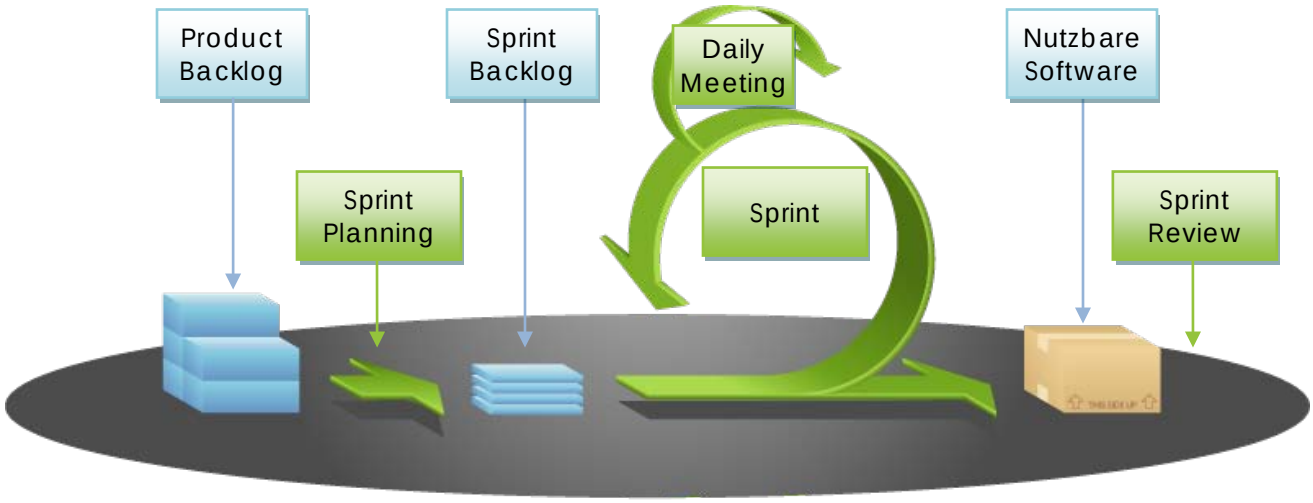

Abbildung 7: Scrum Ablauf (überarbeitetes Bild von Mountain Goat Software *[URL11]*)

4.2 Extreme Programming

Scrum gibt keine Empfehlung über Entwicklungstätigkeiten ab. Aus diesem Grund wird zusätzlich zu Scrum das Extreme Programming praktiziert.

Extreme Programming definiert Werte, welche bei der Softwareentwicklung von zentraler Bedeutung sind. Aus diesen Werten werden Prinzipien abgeleitet, welche berücksichtigt werden sollen. Leider sind diese Prinzipien nicht eindeutig definiert und tragen darum - abhängig vom Autor – unterschiedliche Bezeichnungen. *[BECK00] [URL02]*

4.2.1 Im Projekt berücksichtigte Extreme Programming Praktiken

In der folgenden Tabelle werden die berücksichtigten Extreme Programming Praktiken aufgelistet und beschrieben:

Praktiken	Beschreibung
Planning Game	Scrum Team schätzt den Aufwand der von Product Owner vorgegebenen User Stories mittels Planning Poker.
Small Releases	Einen Sprint (=Iteration) dauert zwei Wochen.
Simple Design	Das Design wächst inkrementell.
Refactoring	Refactoring wird ständig von den Entwicklern betrieben.
Testing	Unittests zu jedem Code (falls möglich), welche automatisch von Hudson beim Check-in ausgeführt werden.
Pair Programming	Pair Programming soll von den Teams ausprobiert werden und bei Erfolg regelmässig angewendet werden.
Collective Ownership	Die Scrum Mitglieder dürfen jeden Code ändern.
Continous Integration	Beim Einchecken erstellt Hudson einen Build und führt die Unittests aus. Des Weiteren werden täglich Builds erzeugt.
Ten-Minute Build	Die Build-Dauer wird von Hudson überwacht und in einem Diagramm dargestellt.
Coding Standards	Es gelten die Sun Codierungsrichtlinien. *[URL03]*

Tabelle 9: Berücksichtigte XP Praktiken

5 Risikomanagement

Das folgende Kapitel dient der Analyse der potentiellen Risiken und deren Vorbeugung. Es wird analysiert, welche Auswirkungen ein Eintreten eines Risikos mit sich bringt und mit welchen Massnahmen diese Risiken eingedämmt werden können.

5.1 Risiko Liste

ID	Bezeichnung	Risikobeschreibung	Auswirkungen	EW	SP [h]	K [h]
R01	A* Algorithmus	Der A* Algorithmus lässt sich nicht so einfach umsetzen oder ist komplizierter als erwartet.	Einzelne Features eines Sprints können evtl. nicht rechtzeitig fertiggestellt werden. Zeitplanung verzögert sich. Evtl. muss sogar nach einer Alternative zum A* gesucht werden.	10%	10	1
R02	Mehrere Etagen	Das Zeichnen der Wege über mehrere Etagen ist viel aufwändiger als erwartet.	Einzelne Features eines Sprints können evtl. nicht rechtzeitig fertiggestellt werden. Zeitplanung verzögert sich. Falls es bei der Erfassung von Transferpunkten auch noch Probleme gibt, muss evtl. sogar ganz auf dieses Feature verzichtet werden.	20%	15	3
R03	Mehrere Personen	Das Zeichnen der Wege von mehreren Personen bereitet Probleme oder kann nicht so schön visualisiert werden wie gewünscht. Vielleicht stellt sich auch heraus, dass der A* Algorithmus für diesen Fall nicht mehr geeignet ist.	Das Suchen nach anderen Algorithmen nimmt zusätzliche Zeit in Anspruch. Einzelne Features eines Sprints können evtl. nicht rechtzeitig fertiggestellt werden oder man muss sich mit nicht ganz optimalen Visualisierungen zufrieden geben. Zeitplanung verzögert sich.	20%	20	4
R04	Know-How Verteilung	Know-how für spezialisierten Bereich beschränkt sich auf eine einzelne Person.	Bei einem speziellen Problem bleibt alles an der Know-how tragenden Person hangen. Arbeit kann nicht effizient verteilt werden.	20 %	20	4
R05	SVN Ausfall	Team SVN fällt aus oder erleidet einen Datenverlust.	Neu geschriebener Code kann nicht mehr gesichert und Dateikonflikte nicht mehr synchronisiert werden.	5 %	20	1

R06	Einarbeitung in Code	Die Einarbeitung in den bereits bestehenden Code dauert länger als geplant.	Andere wichtigere Arbeiten verzögern sich, durch den erhöhten Zeitaufwand bei der Einarbeitung.	20%	10	2
R07	Bugs	Kurz vor Ende eines Sprints wird ein grösserer Bug gefunden.	Das Feature kann evtl. im geplanten Sprint nicht geliefert werden. Bestehende Bugs müssen in folgenden Sprints behoben werden.	20%	10	2
R08	Krankheit / Unfall	Ein Projektmitglied wird krank bzw. verunfällt und fällt für mehrere Tage aus.	Einzelne Arbeitspakete können nicht rechtzeitig fertig gestellt werden oder eine andere Person muss die Arbeit übernehmen, was zusätzliche Einarbeitungszeit beansprucht.	5 %	40	2
R09	Uneinigkeit	Meinungsverschiedenheiten innerhalb des Teams führen zu einem grösseren Streit.	Stimmung innerhalb des Teams ist nicht mehr gut und die Motivation der einzelnen Mitglieder sinkt, was sich direkt auf die Qualität der Arbeit auswirken wird.	10 %	20	2
R10	Doppelte Arbeit	Arbeit wird durch zwei Projektmitglieder doppelt ausgeführt.	Kostbare Arbeitszeit, welche anderweitig effizienter hätte eingesetzt werden können, geht verloren.	20%	10	2
R11	Datenverlust	Ein Projekt Artefakt oder Teile des Source Codes werden unwiderruflich gelöscht.	Bereits getätigte Arbeit geht verloren und muss nochmals komplett von Vorne neu gemacht werden. Kostbare Zeit geht verloren.	5 %	20	1
R12	Hardware-Defekt	Notebook eines Projektmitglieds fällt aus.	Das Notebook muss repariert oder ersetzt werden, was zur Folge hat, dass die komplette Konfiguration nochmals gemacht werden muss.	5 %	20	1
Total einzuplanende Reserven in Arbeitsstunden [h]						**25**

Tabelle 10: Risiko Management Liste

Legende:
EW Eintrittswahrscheinlichkeit
SP Schadenspotential (in Arbeitsstunden h)
R Einzuplanende Reserve (EW * SP, in Arbeitsstunden h)

5.2 Massnahmen zur Risikovermeidung

Um die Risiken unter Kontrolle behalten und so gut wie möglich zu eliminieren, haben wir folgende allgemeine Massnahmen definiert:

- An jedem *Daily Scrum* erläutert jeder Entwickler seinen aktuellen Entwicklungsstand und welche Probleme ihn gerade beschäftigen. Entsprechende Massnahmen können direkt mit dem Scrum Master besprochen werden. Somit werden zumindest die entwicklungsspezifischen Risiken abgedeckt.
- Es werden regelmässige Reviews von Programmcodes und Dokumenten durchgeführt. Jeder Code und jedes Dokument muss mindestens einmal von einer anderen Person überprüft und wenn nötig angepasst werden.

Zusätzlich zeigt die folgende Tabelle, welche konkreten Massnahmen im Bezug auf die einzelnen Risiken vorgenommen werden. Falls möglich, wird zusätzlich angegeben, bis zu welchem Datum die getroffenen Massnahmen getroffen werden müssen.

	Bezeichnung	Massnahmen	Datum
R01	A* Algorithmus	Der A* Algorithmus wird bereits im ersten Sprint detailliert analysiert und direkt umgesetzt. Bei Problemen kann zusätzlich auf die Skripte vom vergangenen Modul Prog2 zurückgegriffen werden, da dort dieser Algorithmus auch bereits behandelt wurde.	09.10.2009 Ende Sprint 1
R02	Mehrere Etagen	Damit die Anforderung von mehreren Etagen auch wirklich umgesetzt werden kann, muss man pro Museum die einzelnen Transferpunkte pro Etage erfassen können. Dieses Feature muss von Thomas Kälin oder Michael Rüegg angeboten werden. Bei Problemen wird direkt mit ihnen zwei Kontakt aufgenommen.	06.11.2009 Ende Sprint 3
R03	Mehrere Personen	In vergangenen Arbeiten wurde schon mehrmals analysiert, wie man mehrere Personen in einer Visualisierung darstellen könnte. Diese Analysen müssen genauestens studiert werden. Bei Problemen kann auch bei Thomas Kälin Hilfe eingeholt werden.	06.11.2009 Ende Sprint 3
R04	Know-how Verteilung	Durch Code-Reviews und Daily Scrum Sitzungen probieren wir, das Know-how möglichst verteilt zu halten. Zusätzlich möchten wir bei einer anspruchsvollen Programmieraufgabe einmal beispielhaft das Pair Programming ausprobieren, was auch dazu führt, dass sich das Know-how verteilt.	-

R05	SVN Ausfall	Lokale Arbeitskopie ist immer aktuell zu halten. Zur zusätzlichen Absicherung wird jeweils über Nacht der komplette Stand des SVN Repository auf einen dedizierten Server kopiert. Sofern das Ereignis eintritt, weichen wir auf einen anderen SVN Server aus und verwenden temporär andere Kommunikationskanäle zum Datenaustausch. (E-Mail, CD, USB-Stick...)	-
R06	Einarbeitung in Code	Möglichst früh mit dem Analysieren des bestehenden Codes beginnen. Bei Unklarheiten einen Termin zur Besprechung mit Thomas Kälin festlegen.	09.10.2009 Ende Sprint 1
R07	Bugs	Jeder Entwickler beginnt frühzeitig mit dem Testen seiner einzelnen Komponenten und schreibt wo sinnvoll eigene Unittests. Durch Code Reviews probieren wir zusätzlich das Risiko eines unentdeckten Fehlers zu minimieren.	-
R08	Krankheit / Unfall	Jedes Projektmitglied ist angehalten, seinen Entwicklungsstand laufend ins SVN zu committen und gleichzeitig zu jedem Commit einen sinnvollen Kommentar zu erfassen. Damit der Code von allen einfach verstanden werden kann, muss der Code an komplizierten Stellen mit sinnvollen Kommentaren ergänzt werden.	-
R09	Uneinigkeit	Wir pflegen eine offene Kommunikationskultur und bringen wenn immer möglich konstruktive Verbesserungsvorschläge ein. Eskalationsursachen werden frühzeitig analysiert und versucht zu verhindern. Allenfalls wird der Betreuer oder der Scrum Master hinzugezogen, um bei Uneinigkeiten die definitive Entscheidung zu fällen.	-
R10	Doppelte Arbeit	Genaue Planung, Aufteilung und Abgrenzung der einzelnen Arbeitspakete.	-
R11	Datenverlust	Jedes Teammitglied muss ständig eine aktuelle Kopie des SVN Repository auf seinem Computer halten. So minimieren wir das Risiko, dass ein wichtiges Artefakt komplett verloren geht.	-
R12	Hardware-Defekt	Der Student arbeitet solange sein eigenes Notebook nicht funktioniert an einem HSR Rechner weiter. Da alles in einem zentralen Repository gespeichert ist, kann er relativ schnell wieder weiterarbeiten.	-

Tabelle 11: Massnahmen zur Risikovermeidung

5.3 Risiko Zusammenfassung

Wie in der Risiko Liste im Kapitel *5.1 Risiko Liste* ermittelt wurde, beträgt das gewichtete Schadenspotential aller Risiken für das Projekt *25 Arbeitsstunden*!

Bei einer wöchentlichen Arbeitszeit von 17 Stunden pro Person, entspricht dies etwa einer Arbeitswoche. Auf die 14 Projektwochen verteilt ist dieser Anteil genügend gering, dass diese Reservezeit durch einen Wochenendeinsatz kompensiert werden könnte.

6 Meilensteine

Wie die folgende Abbildung zeigt, beginnt die Semesterarbeit am 14.09.2009 und dauert bis zum 18.12.2009. Da das Ziel eines Scrum Sprints stets eine lauffähige Software am Ende des Sprints ist, wurde jedes Sprintende als separater Meilenstein markiert. Am Ende jedes Sprints soll eine neue Version der bestehenden Software entstehen, was als solche gekennzeichnet wurde.

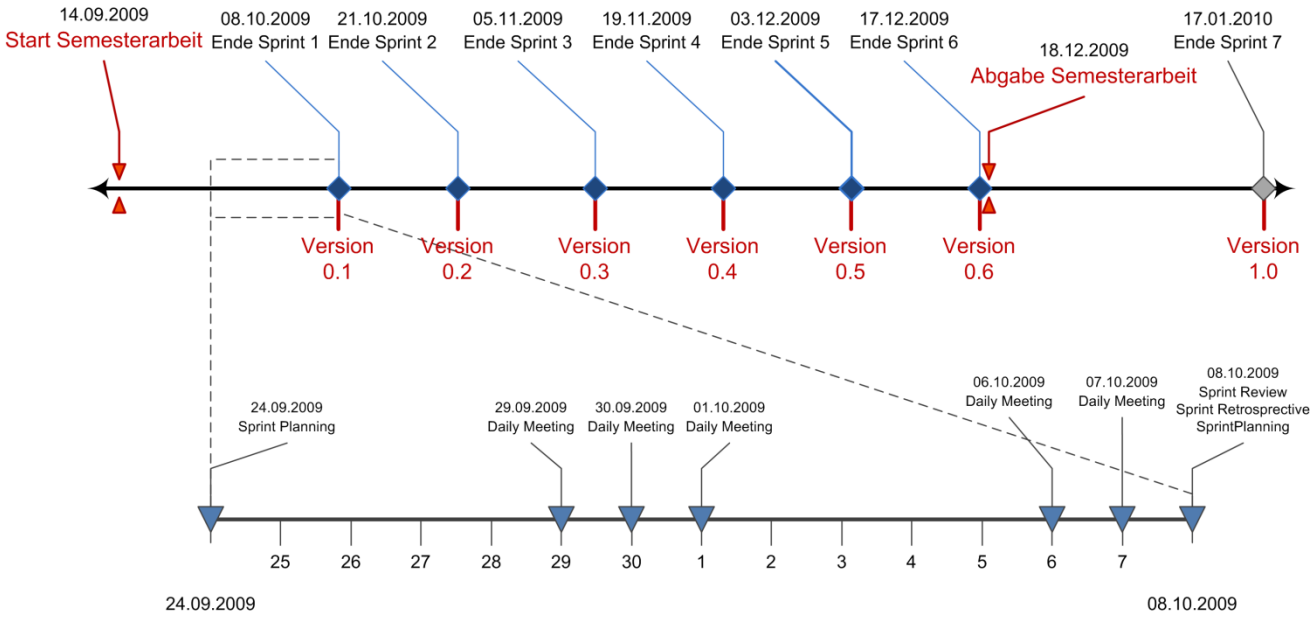

Abbildung 8: Meilensteine

7 Zeitplanung

Gemäss Vorgabe müssen pro ECTS Punkt 30 Arbeitsstunden geleistet werden. Die Semesterarbeit wird mit 8 ECTS Punkten belohnt, somit müssen 240 Stunden aufgewendet werden.

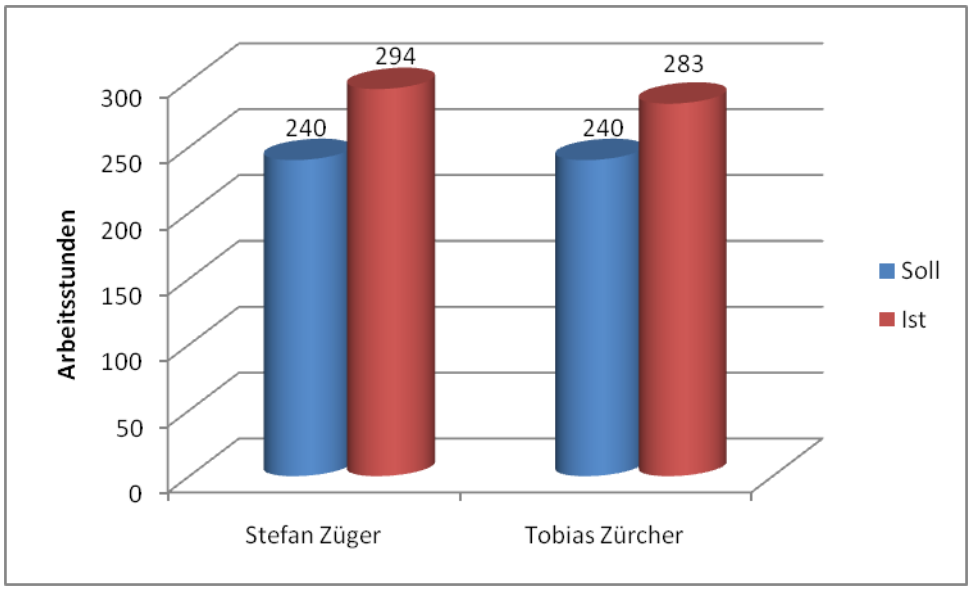

Abbildung 9: Arbeitsstundentotal

Da für die Semesterarbeit 14 Wochen zur Verfügung stehen, muss jeder Student im Durchschnitt17 Stunden pro Woche arbeiten.

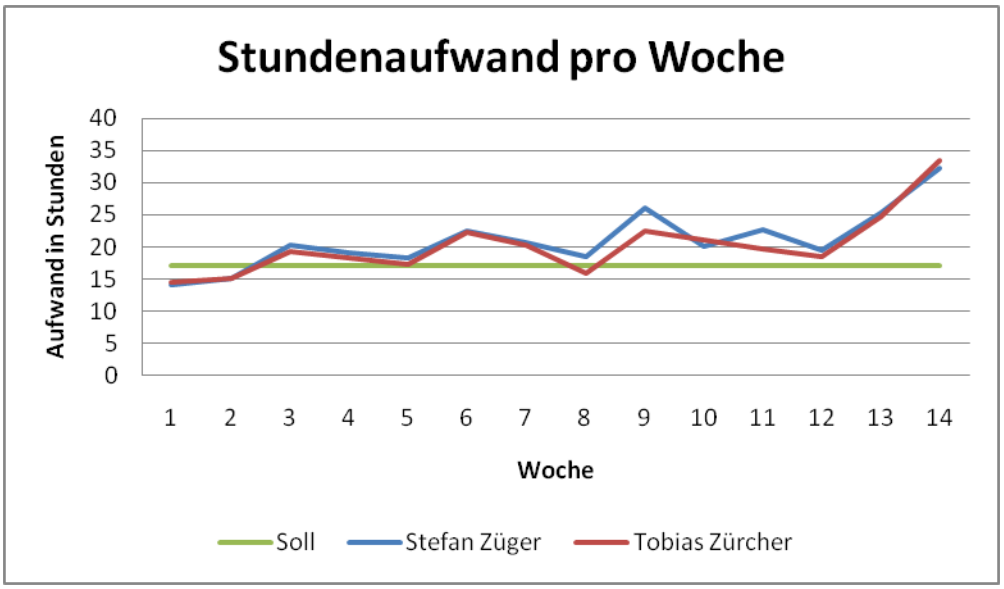

Abbildung 10: Stundenaufwand pro Woche

7.1 Arbeitsaufteilung

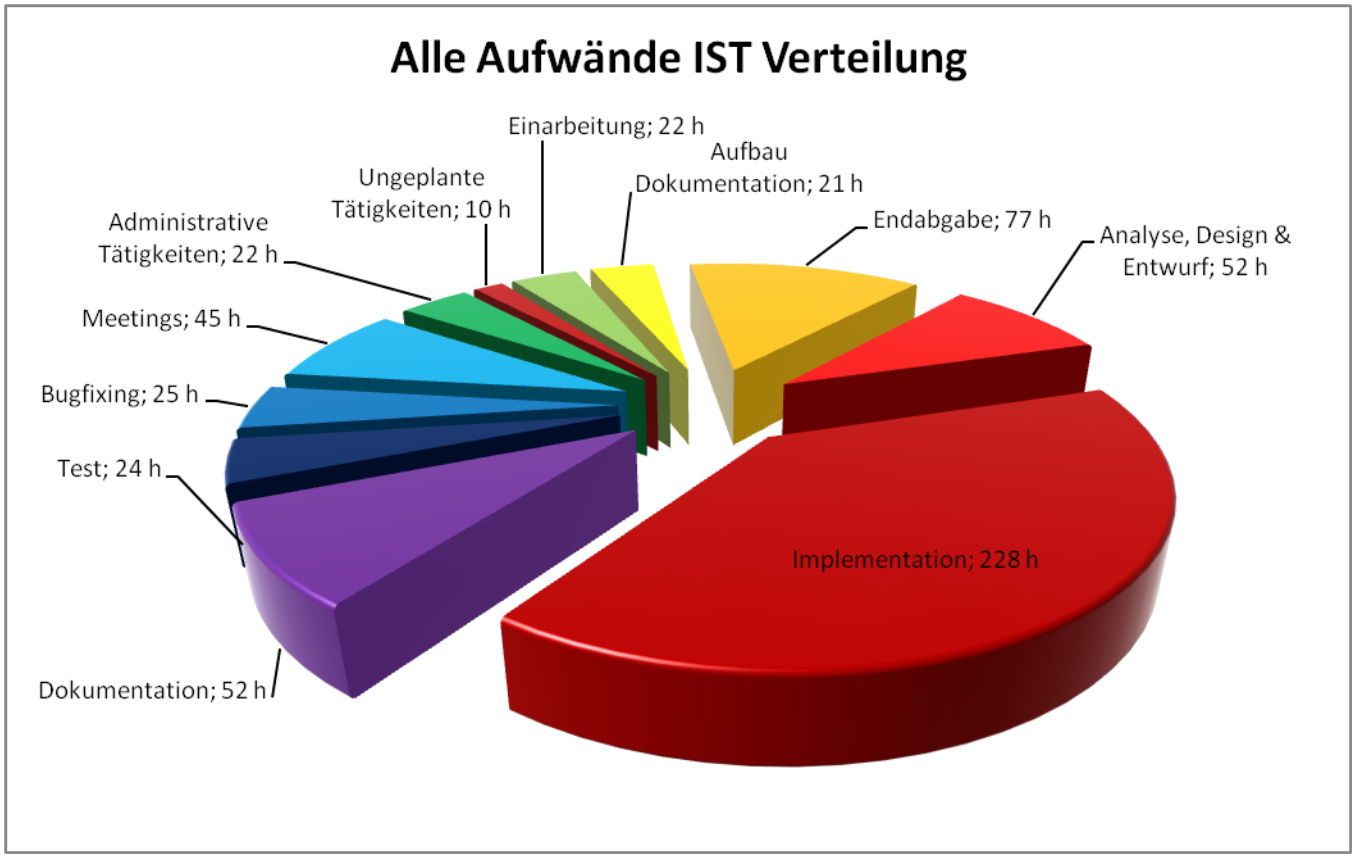

Abbildung 11: Arbeitsaufteilung

7.2 Aufwandvergleich der User Storys

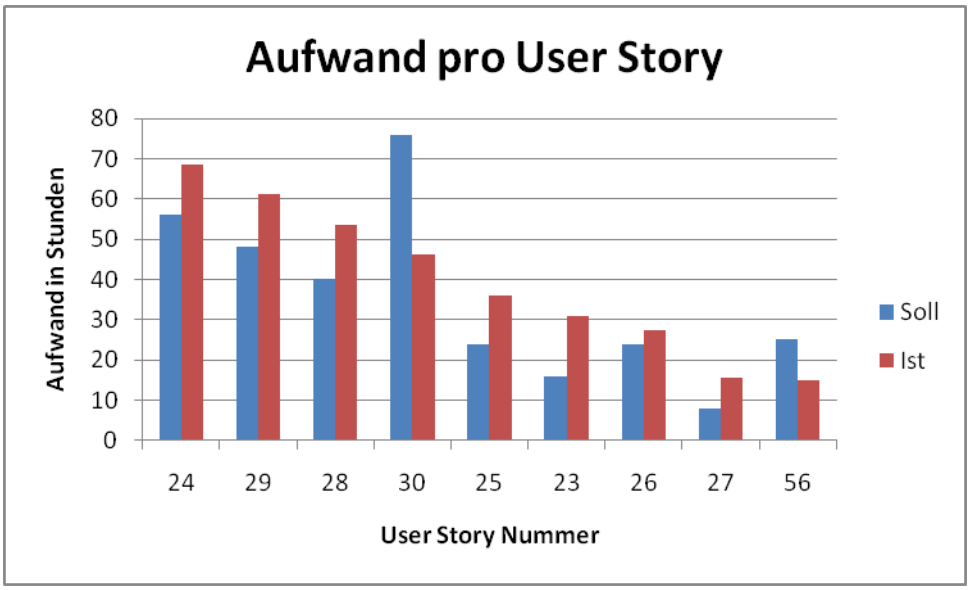

Abbildung 12: Aufwand pro User Story

Nr	User Story
24	Weg einer Person als Linie
29	Auswahl mehrere Einzelpersonen
28	Wege über mehrere Etage darstellen
30	Wege mehrerer Personen: animierte Punkte
25	Weg einer Person in Teilstrecken
23	Weg einer Person animiert
26	Zeiten zum Weg einer Person
27	Geschwindigkeit bei animierten Weg veränderbar
56	Auswahl einer Person

Tabelle 12: Liste der User Storys

8 Qualitätsmassnahmen

Im folgenden Teil werden alle qualitätssichernden Massnahmen detailliert aufgeführt. Das Ziel dieses Kapitels liegt darin, schon zum Beginn des Projektes Massnahmen zu treffen, welche ein qualitativ hochstehendes Endprodukt garantieren.

8.1 Teammeetings

Wie es der Scrum Prozess vorschlägt, halten wir mit dem ganzen Scrum Team und dem Scrum Master sogenannte „Daily Scrum“ Meetings. Diese kurzen Standup-Meetings dienen dem Statusaustausch innerhalb des ganzen Teams. Jeder Entwickler erwähnt woran er gerade arbeitet und an welchen Hindernissen er ansteht. So kann sichergestellt werden, dass jeder den Überblick über den Gesamtfortschritt behalten kann und dass bei allfälligen Problemen direkt Massnahmen getroffen werden können um das Problem zu beheben.

Alle zwei Wochen, d.h. zu Beginn jedes Sprints, findet zusätzlich ein Sprint Planning Meeting statt. An diesem Meeting nehmen der Product Owner, der Scrum Master und das ganze Scrum Team teil. Es wird zuerst ein Review des vergangenen Sprints durchgeführt, bevor dann der nächste Sprint im Detail geplant wird (siehe Kapitel *3.3 Sitzungstermine*).

8.2 Codierungsrichtlinien

Wie es der Extreme Programming (XP) Prozess vorschlägt, wurde für das Gesamtprojekt eine einheitliche Codierungsrichtlinie festgelegt. Es sollen sich alle an die Java Richtlinien von Sun *[URL03]* halten.

Um zu überprüfen ob die Codierungsrichtlinien tatsächlich eingehalten werden, kommt die Software Checkstyle *[URL04]* zum Einsatz. Diese führt automatische Checks durch und listet direkt alles auf, was nicht den Richtlinien entspricht.

8.3 Reviews

8.3.1 Code-Reviews

Während des gesamten Projektes werden laufend Code-Reviews von den komplexeren Klassen durchgeführt.

Der Revisor überprüft folgende Punkte und verbessert wo möglich direkt im Code:

- Verständlichkeit des Codes (evtl. durch Kommentare ergänzen)
- Einhalten der Codierungsrichtlinien unter Mithilfe von Checkstyle
- Korrektheit des Codes
- Auftreten von „Code Smells“
- Effizienz von komplexen Algorithmen

8.3.2 Dokumenten-Reviews

Jedes geschriebene Dokument muss mindestens einmal von der anderen Person durchgeschaut und wenn nötig verbessert werden. Der Revisor trägt sich in die „Revision History“ des entsprechenden Dokuments mit Datum und Kommentar ein.

Der Revisor überprüft:

- Rechtschreibung / Grammatik
- Einhalten des vorgegebenen Dokumenten Templates
- Konsistenz der verschiedenen Texte
- Vollständigkeit des Dokuments

8.4 Testing

8.4.1 Unittests

Jeder Entwickler ist aufgefordert in seinem Source-Code fortlaufend Unittests zu schreiben und gegen diese Tests zu entwickeln. Unittests werden mittels JUnit umgesetzt und sollen überall zur Anwendung kommen, wo es möglich ist, solche zu schreiben. Bevor ein Entwickler seinen neuen Source Code ins SVN eincheckt, müssen alle seine Unittests erfolgreich absolviert worden sein.

8.4.2 FindBugs

Um das Risiko eines unentdeckten Bugs zu minimieren, wird die Software FindBugs *[URL05]* zur fortlaufenden Analyse des Source Codes eingesetzt. FindBugs verwendet diverse statische Analysen um den Source Code nach potentiellen Fehlerquellen abzusuchen.

Abbildung 13: Logo FindBugs

8.4.3 Cobertura

Im Zusammenhang mit den JUnit Tests wird zusätzlich Cobertura *[URL06]* eingesetzt um die allgemeine Code Coverage zu analysieren. Mit Hilfe von Cobertura kann eine detaillierte Auswertung erstellt werden, die aufzeigt, wie viel Prozent des gesamten Java Code durch Unittests abgedeckt wird.

8.5 Build Server

Für das gesamte VisiVis Projekt wurde von Thomas Kälin einen Hudson Build Server *[URL07]* aufgesetzt, mit dessen Hilfe die Conitnuous Integration der Software sichergestellt wird. Der Hudson Server führt regelmässig folgende Tasks aus:

Abbildung 14: Logo Hudson

Task	Zeitpunkt der Ausführung	Beschreibung
VisiVis – JUnit	Täglich um 23:40	Führt alle im Projekt vorhandenen JUnit Tests aus. Zusätzlich wird ein Cobertura bericht erstellt, welcher anzeigt, wie viel Prozent des Codes durch Unittests abgedeckt wird.
VisiVis – FindBugs	Täglich um 23:45	Startet die Ausführung von FindBugs, welches durch statische Codeanalysen potentielle Fehlerquellen aufdeckt und daraus einen ausführlichen Bericht erstellt.
VisiVis – Checkstyle und Metrics	Täglich um 23:50	Berechnet für das gesamte Projekt diverse Code Metriken und überprüft zusätzlich mittels Checkstyle, ob die Codierungsrichtlinien eingehalten wurden.
VisiVis – Build	Jeden Sonntag um 23:55	Kompiliert den aktuell eingecheckten Projektstand und erstellt daraus ein ausführbares Java Programm.

Tabelle 13: Automatisierte Tasks auf dem Build Server

Teil 4: Sprints

9 Sprint-Organisation

Der vorliegende Teil 4 der Dokumentation enthält chronologisch alle abgearbeiteten Sprints des Projekts. Dabei wird jeder Sprint für sich alleine bearbeitet und dokumentiert. Am Ende jedes Sprints wird eine Retrospektive durchgeführt, welche über die positiven, wie auch negativen Aspekte des Sprints reflektiert.

Jeder Sprint verläuft terminlich nach demselben Schema, beginnt jeweils an einem Donnerstag und dauert zwei Wochen. Die folgende Tabelle zeigt den terminlichen Ablauf eines einzelnen Sprints:

	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
W1				Sprint Planning	
W2		Daily Meeting 8h Arbeitszeit	Daily Meeting 4h Arbeitszeit	Daily Meeting 4h Arbeitszeit	
W3		Daily Meeting 8h Arbeitszeit	Daily Meeting 4h Arbeitszeit	Sprint Review Sprint Retrospective Sprint Planning	

Tabelle 14: Grobübersicht eines einzelnen Sprints

9.1 Dauer der einzelnen Sprints

Sprint	Start	Ende
1	24.09.2009	08.10.2009
2	08.10.2009	21.10.2009
3	21.10.2009	05.11.2009
4	05.11.2009	19.11.2009
5	19.11.2009	03.12.2009
6	03.12.2009	17.12.2009

Tabelle 15: Dauer der Sprints

10 Sprint 1

10.1 Sprint Tasks

ID	Beschreibung	Soll	Ist
24-1	Weg einer Person als Linie: Einarbeitung Code (inkl. Caching)	13.0	4.0
24-2	Weg einer Person als Linie: A*-Analyse (bestehender Code)	8.0	14.0
24-3	Weg einer Person als Linie: Skizzen / Entwurf	3.0	2.0
24-4	Weg einer Person als Linie: Implementierung (Unittests)	16.0	32.5
24-5	Weg einer Person als Linie: Doku (speziell Unterschiede zu früher), Erfahrungen	8.0	12.0
24-6	Weg einer Person als Linie: Systemtests (Datensatz suchen)	8.0	4.0
56-1	Auswahl einer Person: Entwurf / Skizze	4.0	0.0
56-2	Auswahl einer Person: Implementierung	5.0	3.0
		65.0	**71.5**

Tabelle 16: Tasks zum Sprint 1

10.2 User Story 24: Weg einer Person als Linie

Die Aufgabe dieser User Story besteht darin, den Weg eines einzelnen Besuchers als Linie darzustellen. Die Darstellung des Weges beschränkt sich auf eine einzelne Etage und muss nicht animiert oder in Teilstrecken dargestellt werden. Zusätzlich zum Weg soll auf jeder Teilstrecke (immer zwischen zwei RFID-Reader) einen Pfeil in Laufrichtung des Besuchers angezeigt werden. Die Person, deren Weg dargestellt wird, muss in dieser User Story noch nicht über ein GUI ausgewählt werden können.

10.2.1 Analyse A* Algorithmus

Das VisiVis System basiert auf einer zellenbasierten Besuchererfassung, was bedeutet, dass die Besucher nur immer pro Zelle und auf den Punkt genau lokalisiert werden. Deshalb müssen die zurückgelegten Wege berechnet werden. P.E. Hart veröffentlichte im Juli 1968 einen Artikel über einen Wegfindungsalgorithmus, den er A* nannte. *[HART69]*

Thomas Kälin hat in seiner Studienarbeit den A* Algorithmus analysiert und in Java umgesetzt. Seine Implementierung zeigt, dass mit diesem Algorithmus zuverlässig und in einem akzeptablen Zeitraum einen Weg berechnet werden kann. *[KÄLIN07]*

Der berechnete Weg ist jedoch sehr eckig und wirkt daher künstlich. Eine erste Idee war die Wege mit Interpolation abzurunden. Eine sehr simple Methode wäre zum Beispiel nur jeden zweiten Punkt zu verwenden. Weitere Ideen und Details zur Implementation der Abrundung des Weges folgt in Kapitel *10.2.2 Implementierung*. Weil der Algorithmus immer den kürzesten Weg sucht, sind die Wege oftmals nahe entlang den Wänden, was sehr künstlich erscheint.

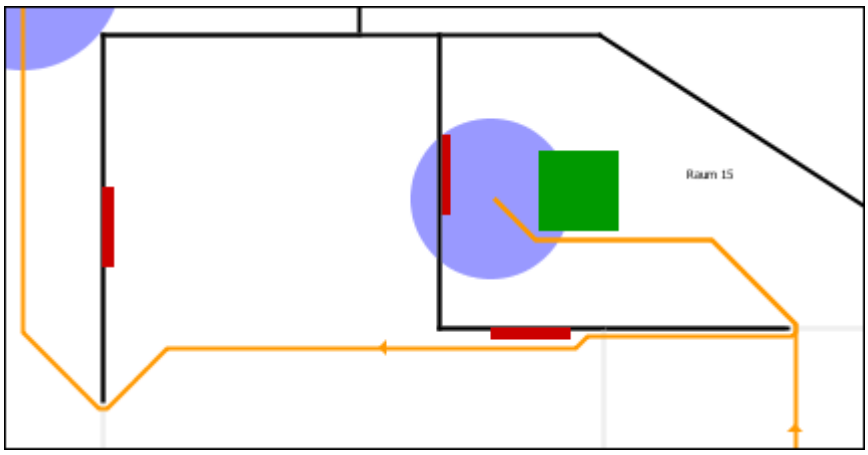

Abbildung 15: Screenshot der Wegvisualisierung in alter Applikation

Es ist klar, dass die berechneten Wege lediglich Annäherungen zum realen Weg sind. Realistisch aussehende Wege wirken viel vertrauenswürdiger. Daher lohnt es sich auch, einige Zeit zu investieren, damit die Wege etwas natürlicher wirken. Diese Anforderung wurde jedoch in eine eigene User Story ausgelagert.

Technische Aspekte
Bei der Analyse des alten Codes fielen als Erstes die zahlreichen und teilweise sehr langen Methoden auf. Die Problem Domain Objekte gelangen bis zum Algorithmus selber. Dies führt zu diversen verschachtelten if-else Blöcken, welche den Code unnötig aufblähen und komplizierter machen. Der Programmablauf ist durch die vielen Methodenaufrufe schwierig zu erkennen und dadurch träge zu lesen und zu verstehen. Zusätzlich fiel die enge Kopplung von Problem Domain und dem Algorithmus auf, welche sich negativ auf die Wartbarkeit des Codes auswirkt: Ändert sich die Problem Domain, so muss der Algorithmus angepasst werden. Genau dies ist nun bei der vorliegenden Arbeit der Fall. Die Problem Domain des Prototyps wurde verbessert, wobei diverse Änderungen am Algorithmus vorgenommen werden müssten, damit dieser wieder funktionieren würde. Eine Abstraktion macht hier Sinn.

Bei der Analyse ist uns die Debug Methode positiv aufgefallen, mit der eine Etage mit dem berechneten Weg in ASCII Form ausgegeben werden kann. Dank dieser Methode lassen sich zugleich geeignete Unittests durchführen. Die Abbildung der Etage ist lediglich ein zweidimensionales Array (auch `FloorMap` genannt). In der alten Version wurde jedoch in jedem Arrayfeld ein Objekt gespeichert. Dies kann optimiert werden indem lediglich ein ASCII-Wert (1 Byte) gespeichert wird. Als Vergleich: Alleine die Adresse des Objektes (4 Bytes) ist um das 4 Fache grösser (der Objektinhalt ist noch nicht mitgerechnet) als ein einzelnes ASCII Zeichen. Die zusätzlichen Informationen, die früher im Objekt gespeichert wurden (Koordinaten), sind redundant, da diese bereits durch den Arrayindex gegeben sind.

10.2.2 Implementierung

Klassendiagramm Wegberechnung
Wie in der Analyse bereits erwähnt, war die enge Kopplung zwischen dem Algorithmus und der Problem Domain unvorteilhaft, was unbedingt verbessert werden sollte. *Abbildung 16* zeigt das Klassendiagramm der neuen Implementierung, welche die Problematik besser abstrahiert. Es wurde besonders darauf geachtet, dass die Abhängigkeit zwischen UI und Algorithmus aufs Kleinste minimiert wird. Um die Logik vom UI zu entfernen, wurde der `WayManager` als Controller dazwischengeschaltet. Somit wird die komplette Funktionalität zur Wegberechnung an einer einzigen zentralen Stelle dem UI zur Verfügung gestellt.

Damit der Algorithmus tatsächlich unabhängig von der Problem Domain wird, musste im Controller Package ein `ObjectConverter` hinzugefügt werden. Diese Klasse wandelt die

Problem Domain Objekte (alle vom Typ `AbstractFloorObject`) in eine neutrale Darstellung um. Konkret erstellt er für jedes Museumsobjekt anhand seiner Eckpunkte eine Liste von Punkten. Diese neutrale Information genügt dem Algorithmus, um seine Wege berechnen zu können. Eine allfällige Änderung in der Problem Domain führt höchstens zu einer Änderung im `ObjectConverter`.

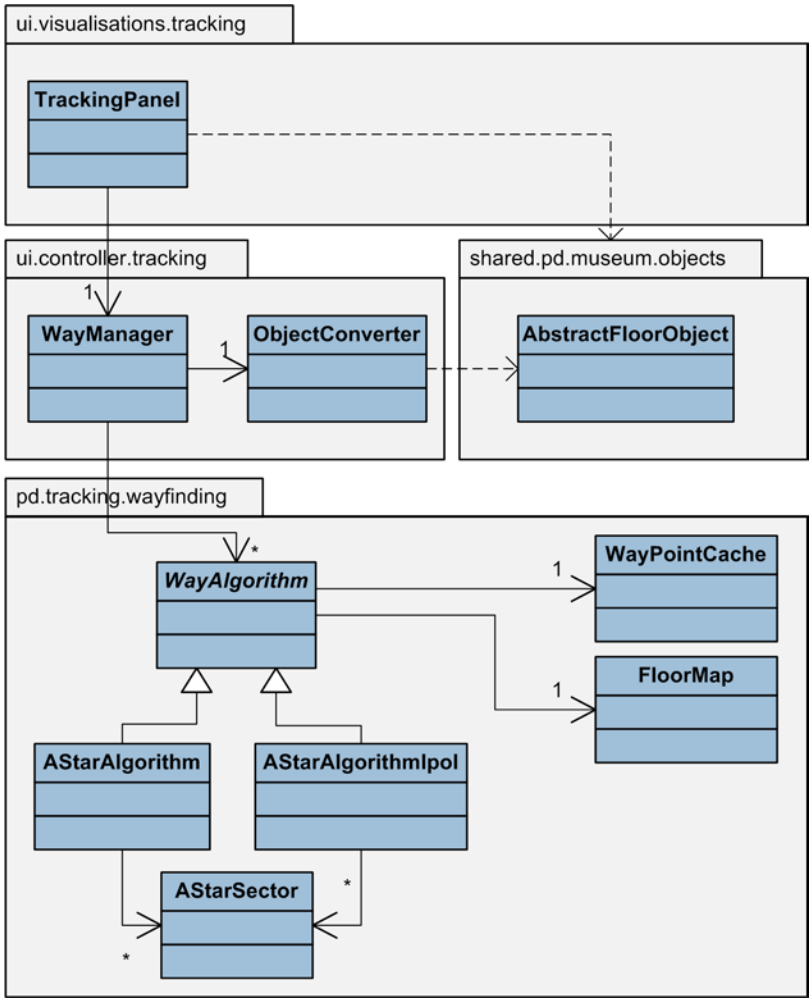

Abbildung 16: Klassendiagramm Wegberechnung

Beschreibung der Klassen

Klasse	Beschreibung
TrackingPanel	UI Klasse, welche den berechneten Weg auf einem Panel anzeigt.
WayManager	Controller Klasse, die dem UI die Möglichkeit bietet, neue Wege zu berechnen oder bereits berechnete Wege abzufragen.
ObjectConverter	Konvertiert Problem Domain spezifische Klassen in ein neutrales Format, mit welchem der Algorithmus rechnen kann, ohne dass er Kenntnis über die Problem Domain haben muss.
AbstractFloorObject	Abstrakte Basis Klasse für alle Objekte, die in einem Museum vorhanden sein können.
WayAlgorithm	Abstrakte Basisklasse, welche alle Methoden vorgibt, die ein Wegberechnungs-Algorithmus unterstützen muss. So ist es theoretisch möglich, unterschiedliche Algorithmen zu implementieren. Diese Klasse stellt gemäss dem *Template Method Pattern [GAMMA95]* bereits die meisten generischen Methoden zur Verfügung und definiert ein paar wenige abstrakte Methoden, die vom konkreten Algorithmus implementiert werden müssen.
AStarAlgorithm	Implementierung des A* Algorithmus
AStarAlgorithmIpol	Zweite Implementierung des A* Algorithmus, welche aber rundere Wege berechnen kann.
AStarSector	Hilfsklasse für den A* Algorithmus. Repräsentiert einen möglichen Punkt des kompletten Weges, welcher untersucht wird.
WayPointCache	Cache, der alle berechneten Wege speichert. So wird verhindert, dass ein Weg zwischen zwei Punkten mehrmals berechnet werden muss. Später soll ein Limit konfiguriert werden können, welches festlegt, wie viele Wege maximal zwischen zwei Punkten berechnet werden.
FloorMap	Interne Darstellung einer Etage eines Museums für den A* Algorithmus. Enthält alle Informationen über Hindernisse auf der Etage. Diese Informationen dienen als Grundlage für die Berechnung der möglichen Wege.

Tabelle 17: Beschreibung der Klassen zur Wegberechnung

Berechnung und Zeichnen des Weges

Damit das `TrackingPanel` schliesslich den berechneten Weg Pixel für Pixel zeichnen kann, erhält es vom `WayManager` eine Liste mit allen Pixel des zurückgelegten Weges. Eine erste Variante, dies zu visualisieren, war ganz einfach wie in den früheren Versionen, indem man Pixel um Pixel mit geraden Linien verbindet.

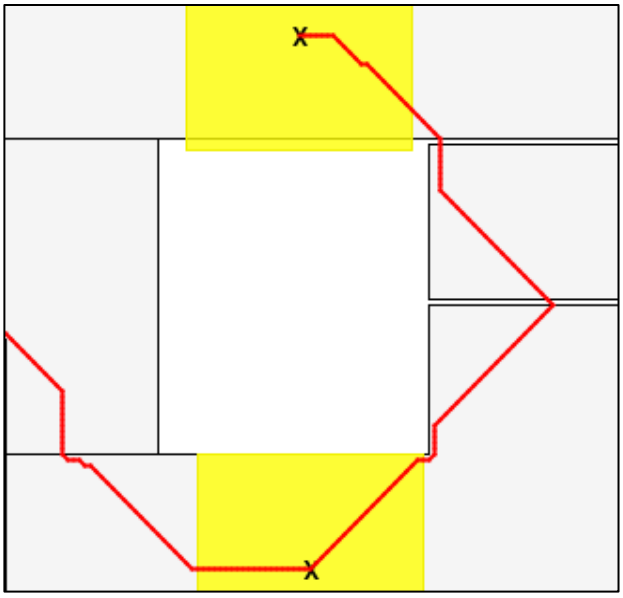

Abbildung 17: Weg ohne Interpolation und mit Zeichnen von Linien

Wie in den alten Versionen der Software führt diese Variante jedoch zu teilweise sehr eckigen und unnatürlichen Wegen. Um eine erste Verbesserung zu erzielen, wurde ein zweiter Algorithmus entwickelt (`AStarAlgorithmIpol`), der aus dem berechneten Weg gezielt einzelne Pixel heraus löscht. So erscheinen die Wege bei sehr kantigen Übergängen ein wenig runder.

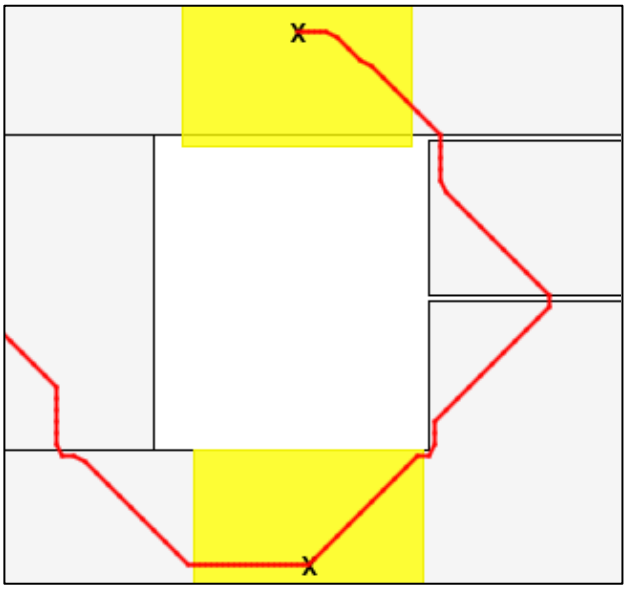

Abbildung 18: Weg mit Interpolation und mit Zeichnen von Linien

Doch auch diese Variante war nicht zufriedenstellend. Aus diesem Grund wurden weitere Nachforschungen betrieben, wobei dann festgestellt wurde, dass in der Java2D Library auch Kurven zwischen jeweils drei Punkten gezeichnet werden können. Somit konnte letztlich das beste Resultat erzielt werden, indem der neue Algorithmus eingesetzt wurde, welcher einzelne Punkte heraus löscht und beim Zeichnen jeweils drei Punkte mittels der `curveTo()` Methode verbindet.

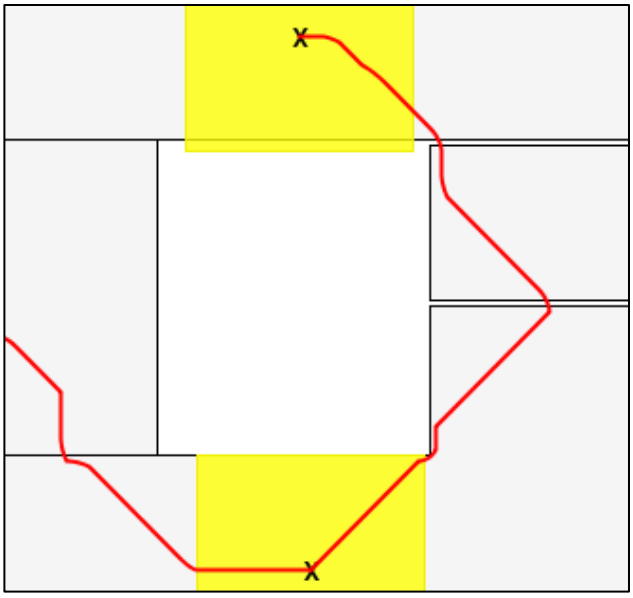

Abbildung 19: Weg mit Interpolation und mit Zeichnen von Kurven

10.2.3 Tests

Unittests

In dieser User Story konnten beinahe alle Klassen zu 100% durch Unittests abgedeckt werden. Die einzige Klasse, bei der kein Unittest möglich war, ist die Klasse `TrackingPanel`. Hierbei handelt es sich auch um eine reine GUI Klasse, was generell sehr schwierig durch Unittests abgedeckt werden kann.

Folgende Unittest Suites und Testklassen wurden für diese User Story erstellt:

ch.hsr.ifs.visivis.visual.ui.controller.ControllerTestSuite
ch.hsr.ifs.visivis.visual.ui.controller.tracking.WayManagerTester

ch.hsr.ifs.visivis.visual.pd.PDTestSuite
ch.hsr.ifs.visivis.visual.pd.tracking.wayfinding.AStarSectorTester
ch.hsr.ifs.visivis.visual.pd.tracking.wayfinding.AStarStressTester
ch.hsr.ifs.visivis.visual.pd.tracking.wayfinding.FloorMapTester
ch.hsr.ifs.visivis.visual.pd.tracking.wayfinding.WayAlgorithmTester
ch.hsr.ifs.visivis.visual.pd.tracking.wayfinding.WayPointCacheTester

Wie *Abbildung 20* und *Abbildung 21* zeigen, konnte durch diese Anzahl an Testklassen eine Testabdeckung von 100% erreicht werden.

Element	Coverage	Covered Instructions	Total Instructions
ch.hsr.ifs.visivis.visual.ui.controller.tracking	100.0 %	591	591
ObjectConverter.java	100.0 %	289	289
WayManager.java	100.0 %	302	302

Abbildung 20: Code Coverage im Package ch.hsr.ifs.visivis.visual.ui.controller.tracking

Element	Coverage	Covered Instructions	Total Instructions
ch.hsr.ifs.visivis.visual.pd.tracking.wayfinding	100.0 %	1101	1101
AStarAlgorithm.java	100.0 %	369	369
AStarAlgorithmIpol.java	100.0 %	119	119
AStarSector.java	100.0 %	153	153
FloorMap.java	100.0 %	191	191
WayAlgorithm.java	100.0 %	216	216
WayPointCache.java	100.0 %	53	53

Abbildung 21: Code Coverage im Package ch.hsr.ifs.visivis.visual.pd.tracking.wayfinding

Systemtests

Zur Ermittlung von Testdatensätzen wurde folgende Datenbankabfrage verwendet:

```
SELECT  visitor_id, count(id) as `Anzahl Recordals`
FROM recordal
GROUP BY visitor_id
ORDER BY `Anzahl Recordals` DESC
```

Besonders interessant sind die Fälle mit den meisten oder wenigsten Recordals (Grenzwerte):

visitor_id	Anzahl Recordals
378	82
239	74
9	74
32	70
10	68
246	66
238	64
42	64
266	64
376	62
377	62
47	60

visitor_id	Anzahl Recordals
140	8
141	8
19	8
71	6
264	6
31	4
138	4
139	4
3	4
224	2
4	2
337	2

Abbildung 22: Testdaten

Da diese User Story noch keine Auswahl der Besucher über das GUI beinhaltete, konnte das Testen von verschiedenen Besuchern nur getätigt werden, indem im Code statisch den gewünschten Besucher anhand der ID auswählt wurde. Dasselbe galt auch für das Muesum, wobei hier immer das Museum mit der ID 1 gewählt werden konnte, da es nur ein Museum in den Testdaten existiert.

Folgender Code war in der Klasse `MainView` nötig, um einen fixen Besucher zu selektieren:

```
// Visitor hard-coded selektieren
Visitor selectedVisitor = appController
                              .getLoadedMuseum(1) // Museum
                              .getActiveState()
                              .getVisitor(1);     // Besucher

BaseWindow<TrackingPanel> bw = new TrackingWindow(selectedVisitor);
```

Die Darstellung der Wege wurde mit folgenden Testdaten erfolgreich getestet:

Museum_id	Visitor_id	Kommentar	OK?
1	378	Meisten Recordals (82)	✔
1	238	Zweitmeisten Recordals (74)	✔
1	90	Mittlere Menge Recordals (38)	✔
1	16	Mittlere Menge Recordals (36)	✔
1	231	Wenige Recordals (10)	✔
1	141	Wenige Recordals (8)	✔
1	337	Sehr wenige Recordals (2)	✔
1	4	Sehr wenige Recordals (2)	✔

Tabelle 18: Testresultate der Systemtests für die Wegberechnung

Performance-Tests

Um die Performance des neu erstellten Algorithmus im Vergleich zum alten zu testen, wurde eine Testklasse erstellt, die auf einer Etage verschieden lange Wege berechnet. Diese Klasse heisst `AStarPerformanceTester`. Als Vorlage dient eine Etage, die eine maximale Anzahl Gänge aufweist. Der Test variiert dabei mit der Anzahl Korridoren und mit der Länge der einzelnen Korridore. Es wurde auch eine Testklasse geschrieben, die dieselben Wege mit dem alten Algorithmus berechnet. Eine Etage mit 10 Korridoren und einer Korridorlange von 200 Pixel würde wie folgt aussehen:

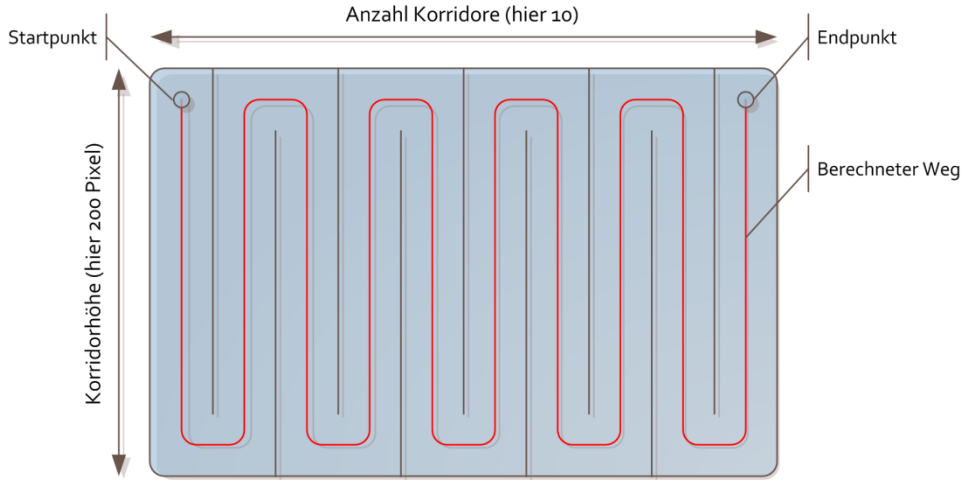

Abbildung 23: Etagendefinition für Performance Tests

Es wurde ein Testlauf durchgeführt, bei dem eine konstante Korridorlänge von 300 Pixel festgesetzt und dabei die Wegberechnungen mit 2 bis 50 Korridoren durchgeführt wurden. Wie das folgende Diagramm zeigt, ist der neue Algorithmus bei 50 Korridoren 280 Mal schneller, wobei der Steigerungsfaktor exponentiell zunimmt. Bereits bei einer Anzahl von nur zwei Korridoren ist der neue Algorithmus sechs Mal schneller. Die genauen Auswertungen sind im Verzeichnis `04 Performance Tests` zu finden.

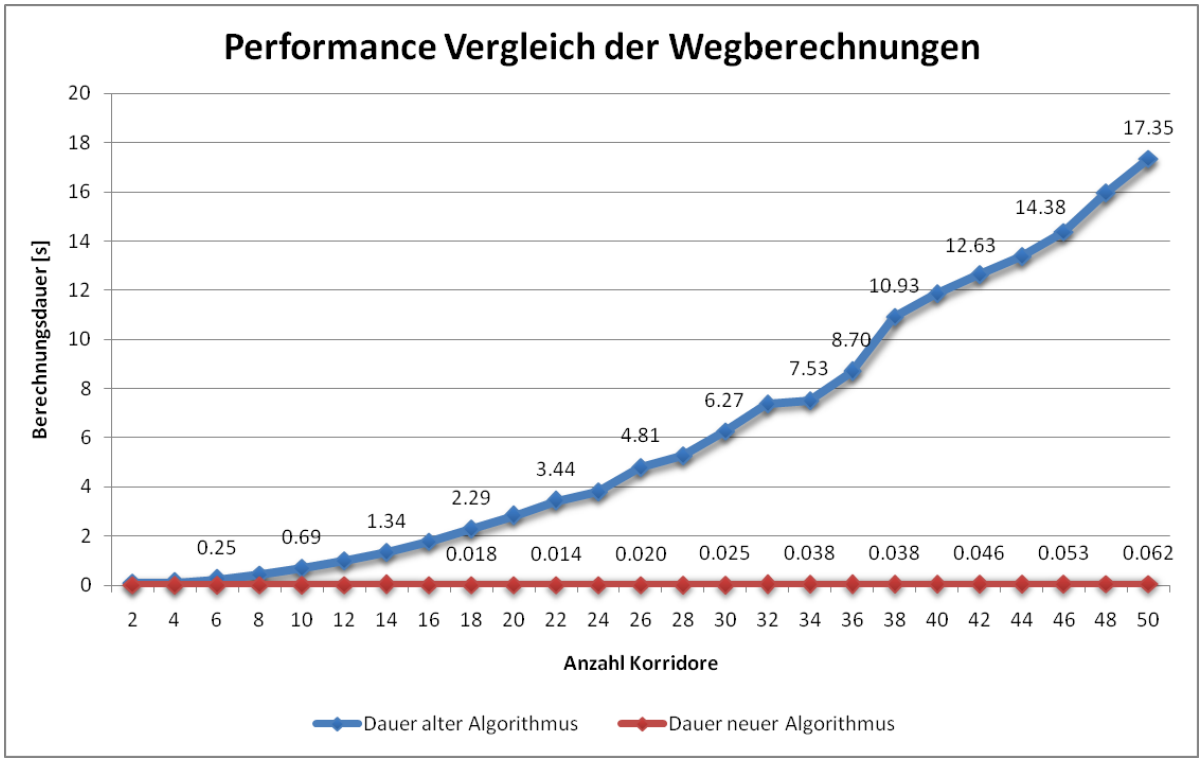

Abbildung 24: Performance Vergleich der Wegberechnungen

10.2.4 Fazit

Die Implementierung dieser User Story verschlang schliesslich beinahe das Doppelte der Zeit, die wir ursprünglich eingeplant hatten. Dennoch können wir sagen, dass sich dieser Mehraufwand durchaus gelohnt hat. Dank Pair Programming und diversen gemeinsam erstellten Skizzen auf Papier, ist schliesslich eine sehr saubere A*-Implementierung entstanden, die eine geringe Kopplung und eine stark gesteigerte Performanz aufweist. Nebenbei wird viel sparsamer mit dem Speicher umgegangen, da der Algorithmus intern nur noch mit Arrays von primitiven Datentypen arbeitet.

Aus Sicht der Visualisierung konnten wir die Darstellung der Wege etwas verschönern, indem die Kanten geglättet werden und die Wege dadurch runder erscheinen. Das Problem mit dem entlang schleichen an den Wänden besteht aber auch noch in unserer Lösung. Dies wird in einer separaten User Story nochmals analysiert und verbessert. Erste Ideen zur Verbesserung sind bereits diskutiert worden.

10.3 User Story 56: Auswahl einer Person

Diese User Story verlangt, dass der Benutzer eine Auswahlmöglichkeit übers GUI erhält, um einen beliebigen Besucher auszuwählen, dessen Weg anschliessend visualisiert wird. Es soll ein einfacher GUI Selektionsdialog umgesetzt werden, der über das Menü geöffnet werden kann. Die Auswahl mehrerer Personen soll noch nicht unterstützt werden. Des Weiteren ist noch nicht verlangt, dass der Dialog bereits über diverse Funktionen wie z.B. benutzerdefi-

nierte Annotationen verfügt. Diese Anforderung wird durch eine separate User Story abgedeckt.

10.3.1 Entwurf / Skizze

Da für die User Story 24 mehr Aufwand betrieben wurde, als ursprünglich eingeplant, blieb für die Implementation der User Story 56 so gut wie keine Zeit mehr. Wir entschieden uns deshalb dazu, einen ersten Prototyp der Personenauswahl ansatzweise fertigzustellen, so dass man am Ende des ersten Sprints wenigstens die Möglichkeit hat, die Wege von verschiedenen Personen auf den verschiedenen Etagen darzustellen. Dies erleichtert dann natürlich auch das Testen der Wegberechnung, da viel einfacher zwischen den einzelnen Personen um geschalten werden konnte.

Wie die folgende Skizze zeigt, enthält diese erste Version der Personenauswahl noch überhaupt keine speziellen Funktionen, sondern bietet lediglich die Möglichkeit, zwischen verschiedenen Besuchern und Etagen auszuwählen.

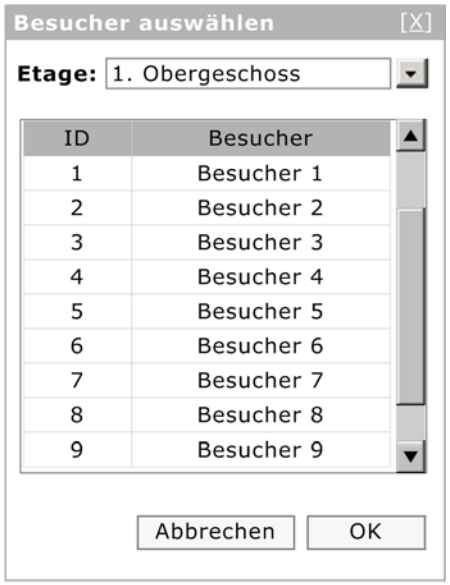

Abbildung 25: Skizze User Story 56

10.3.2 Implementierung

Da der Entscheid gefallen ist, lediglich einen ersten Prototyp der Personenauswahl zu erstellen, gibt es zur Implementierung nicht viel zu erwähnen. Es handelt sich beim GUI, wie schon in der Skizze aufgezeichnet, lediglich um einen einfachen Dialog, welcher die vorhandenen Etagen und eine Tabelle mit allen erfassten Besuchern aufzeigt.

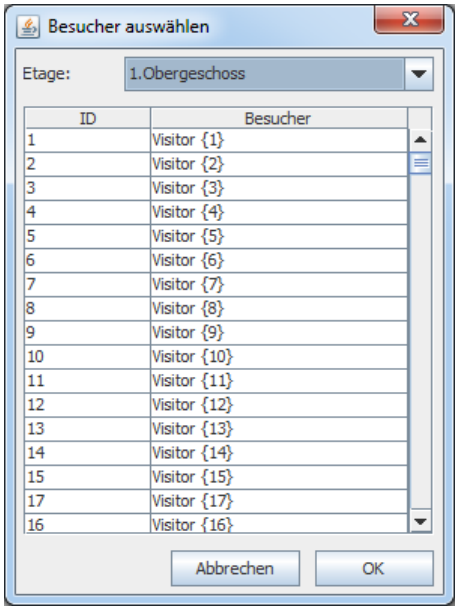

Abbildung 26: Screenshot Prototyp der User Story 56

10.3.3 Fazit

Wenn man diese User Story ganz unabhängig für sich alleine ansieht, so wäre das von aussen „Sichtbare“ schon ein wenig enttäuschend. Man muss jedoch berücksichtigen, dass bewusst mehr Zeit in die Umsetzung des A*-Algorithmus investiert wurde, da wir dort ein grosses Verbesserungspotential erkannten und alles von Grund auf neu schreiben wollten. Unter diesem Aspekt kann man sich mit dem ersten Prototypen der Besucherauswahl zufrieden geben, da sich die Besucher nun über einen Dialog auswählen lassen. Die Funktionalität ist somit ausprogrammiert, das entsprechende Dockable zur Auswahl kann sobald das User Interface Gerüst fertig implementiert ist, auch in Angriff genommen werden.

10.4 Sprint Retrospektive

Positive Aspekte des Sprints

+ Gute Zusammenarbeit & Atmosphäre im ganzen Scrum-Team
+ Daily Meetings immer pünktlich begonnen und dauerten nie länger als 15 Minuten
+ Sehr schönes Projekt-Setup mit allen Tools (Hudson, Checkstyle, Cobertura, Findbugs)
+ Pair Programming war für die Umsetzung des A* Algorithmus besser als erwartet. Man ist zwar anfangs nicht ganz so effizient, dafür kommt man eher auf eine Lösung, die von Beginn an für beide optimal erscheint. Schliesslich konnten wir durch Pair-Programming eine sehr saubere Implementierung des A* Algorithmus realisieren

Negative Aspekte des Sprints

– Scrum Planning Meeting dauerte zu lange (es wurde zu lange an Details diskutiert).
– Die Planung der Tasks am Scrum Planning Meeting war zu detailliert und verlängerte die ganze Sitzung.
– User Story 56 konnte nicht komplett umgesetzt werden, da die Implementierung des A* Algorithmus mehr Zeit brauchte.

Verbesserungsmassnahmen für nächsten Sprint

Abgeleitet aus den negativen Punkten des Sprints, wurden folgende Massnahmen zu Verbesserung des nächsten Sprints getroffen:

- Beim nächsten Scrum Planning Meeting werden versuchsweise nur noch die Story Points für eine komplette User Story im Plenum geschätzt. Die detaillierte Zeitplanung und Aufteilung der Tasks führt jedes Teilprojektteam selber durch und kommentiert die Überlegungen kurz am ersten Daily Meeting, damit allfällig vergessene Dinge korrigiert werden können.
- Auch wenn das nächste Sprint Planning Meeting trotzdem wieder gleich lange dauern würde, wird man versuchen zwischendurch eine 5 Minute Pause einzulegen.

11 Sprint 2

11.1 Sprint Tasks

ID	Beschreibung	Soll	Ist
25-1	Weg einer Person in Teilstrecken: Analyse/Design Darstellung Teilstrecken	3.0	2.0
25-2	Weg einer Person in Teilstrecken: Implementierung	14.0	29.5
25-3	Weg einer Person in Teilstrecken: Systemtest	2.0	1.0
25-4	Weg einer Person in Teilstrecken: Dokumentation	5.0	3.5
26-1	Zeiten zum Weg einer Person: Analyse Positionierung (Skizze)	3.0	1.0
26-2	Zeiten zum Weg einer Person: Implementierung	15.0	19.0
26-3	Zeiten zum Weg einer Person: Systemtest	3.0	2.0
26-4	Zeiten zum Weg einer Person: Dokumentation	4.0	5.5
		48.0	**63.5**

Tabelle 19: Tasks zum Sprint 2

11.2 User Story 25: Weg einer Person in Teilstrecken

In der User Story 25 geht es um das Visualisieren des Weges einer Person in Teilstrecken. Das heisst, es soll dem Benutzer der Anwendung ermöglicht werden, den ganzen zurückgelegten Weg in Teilstrecken Schritt für Schritt durchzugehen. Dabei wird eine Möglichkeit geboten, die Teilstrecken vorwärts wie rückwärts zu durchlaufen.

11.2.1 Analyse der bestehenden Lösung

Bei der Analyse der alten Lösung fällt direkt auf, dass die Teilstrecke anders definiert wurde, als es jetzt angedacht ist. Damals war eine Teilstrecke immer vom Mittelpunkt eines Readers zum Mittelpunkt des nächsten Readers. Die *Abbildung 27* zeigt anhand der alten Software, wie damals eine Teilstrecke innerhalb zwei unterschiedlichen Reader visualisiert wurde.

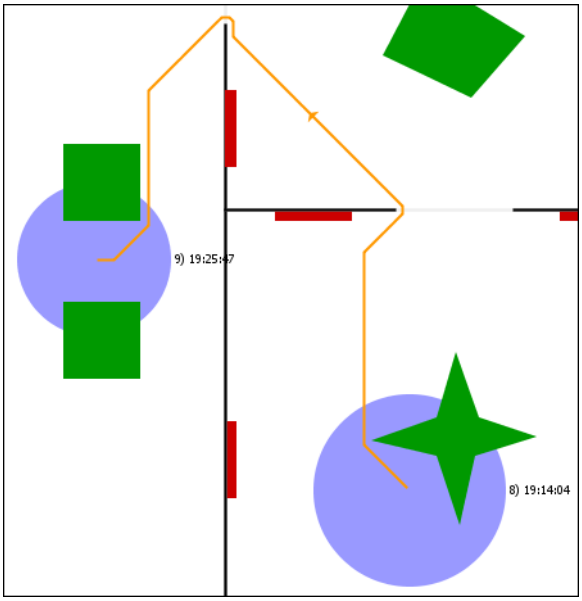

Abbildung 27: Visualisierung Teilstrecke zwischen Readern in alter Lösung

Zusätzlich zur Teilstrecke zwischen zwei RFID-Readern wurde auch ein Aufenthalt innerhalb des Erfassungsbereiches des Readers als Teilstrecke visualisiert. Dies wurde aber ohne Linie für den Weg dargestellt, sondern einfach mit der Anfangs- und Endzeit der Erfassung. Die *Abbildung 28* zeigt exemplarisch wie dies in der früheren Applikation dargestellt wurde.

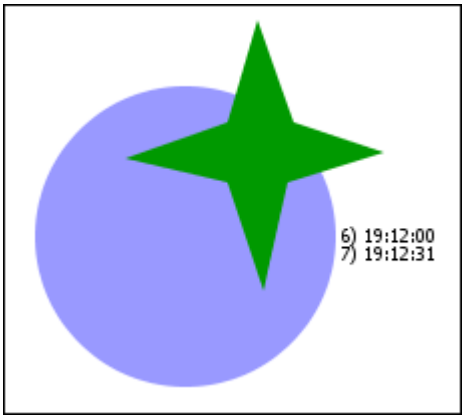

Abbildung 28: Visualisierung Teilstrecke innerhalb Reader in alter Lösung

Um schrittweise immer eine einzelne Teilstrecke nach der anderen darzustellen, hatte der Benutzer zwei Buttons in der Toolbar zur Verfügung, mit denen er immer einen Schritt nach vorn oder einen Schritt zurück schreiten konnte.

Abbildung 29: Toolbar alte Applikation

Der Nachteil dieser Lösung war, dass man nie genau wusste, bei welcher Etappe man sich gerade befindet. Eine Anzeige bei welcher Etappe man gerade steht und wie viele im Total vorhanden sind, wäre sicher für den Benutzer von Vorteil.

11.2.2 Analyse & Design der neuen Lösung

Teilstrecken

Wie bereits erwähnt, wird für die neue Applikation das Verständnis einer Teilstrecke neu definiert. Es gibt neu eine Teilstrecke, welche sich über den Eintrittspunkt in einem Reader bis zum Austrittspunkt eines Readers ausdehnt. Die zweite Teilstrecke ist vom Austrittspunkt des einen Readers bis zum Eintrittspunkt des zweiten Readers. Die *Abbildung 30* visualisiert die neue Definition der Teilstrecken. Hinzu kommt noch, dass jeder RFID-Reader nun einen Wegpunkt bestimmt, durch welchen immer alle Wege passieren. Das heisst eine Wegdarstellung eines einzelnen Besuchers geht nun nicht mehr zwingend durch den Mittelpunkt des Erfassungsbereiches.

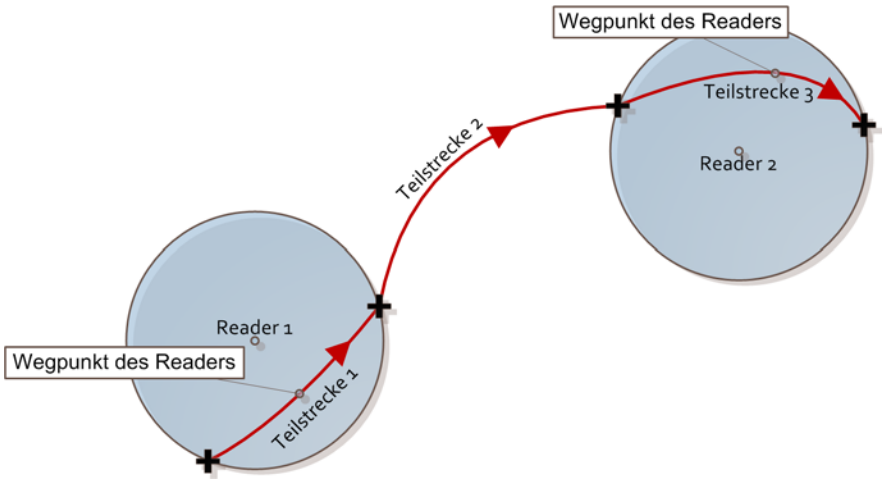

Abbildung 30: Definition von Teilstrecken

Darstellung der Teilstrecken

Im Prototyp konnte pro Etappe, die nach vorn oder zurück geschalten werden konnte, immer nur die aktuelle Teilstrecke angezeigt werden. In der neuen Version sollen neben der aktuellen Etappe auch alle restlichen Teilstrecken dargestellt werden, diese sind allerdings in einer helleren Farbe mit Transparenz und als gestrichelte Linie gezeichnet. Das Einhalten der Java Codierungsrichtlinien ist uns anfangs ziemlich schwer gefallen, da wir uns einen leicht anderen Stil angeeignet hatten. Doch mit Hilfe des Checkstyle Plug-Ins für Eclipse war es überhaupt kein Problem mehr sich daran zu halten.

Gerade bei einem Projekt, an dem mehr als nur zwei Entwickler mitarbeiten, macht es durchaus Sinn, solche Coding Standards durchzusetzen. Dies vor allem in Hinblick auf Folgeprojekte, welche mit dem bestehenden Code weiterarbeiten müssen. Dies gibt einen besseren Überblick über die komplette Strecke ohne dass dabei die Übersichtlichkeit verloren geht.

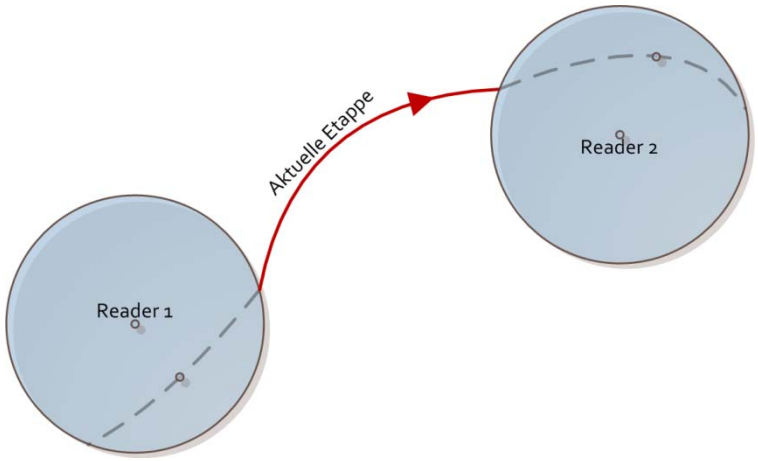

Abbildung 31: Design Etappen Darstellung

Etappenregler

Wie schon in der Analyse der alten Applikation herausgefunden wurde, soll neu eine Information vorhanden sein, die aussagt, welche Teilstrecke gerade angezeigt wird und wie viele solche Etappen überhaupt vorhanden sind. Der Etappenregler soll nicht mehr in der Toolbar, sondern als eigenständiges Dockable auf der rechten Seite der Visualisierung dargestellt werden.

Abbildung 32: Design Etappenregler

11.2.3 Implementierung

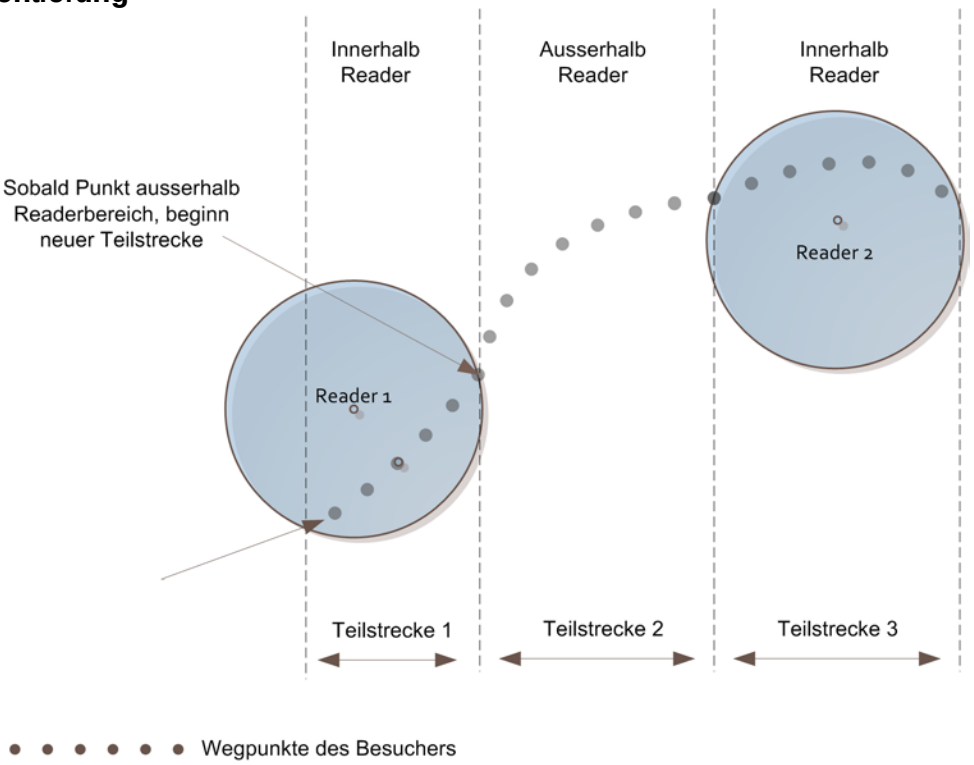

Abbildung 33: Wegaufteilung in Teilstrecken

Zur Aufteilung des Weges in einzelne Teilstrecken wird durch alle Wegpunkte (Wegpunkte in Abbildung 33 ersichtlich) iteriert und entsprechende Listen erstellt.

Abstrakt und vereinfacht formuliert sieht der Algorithmus wie folgt aus:

```
Iteriere durch alle Wegpunkte {
  IF (Wegpunkt auf Readergrenze UND Readerwegpunkt erreicht) {
    aktuelle Teilstrecke beenden
    neue Teilstrecke beginnen
  }
  aktueller Wegpunkt zur Teilstrecke hinzufügen
}
```

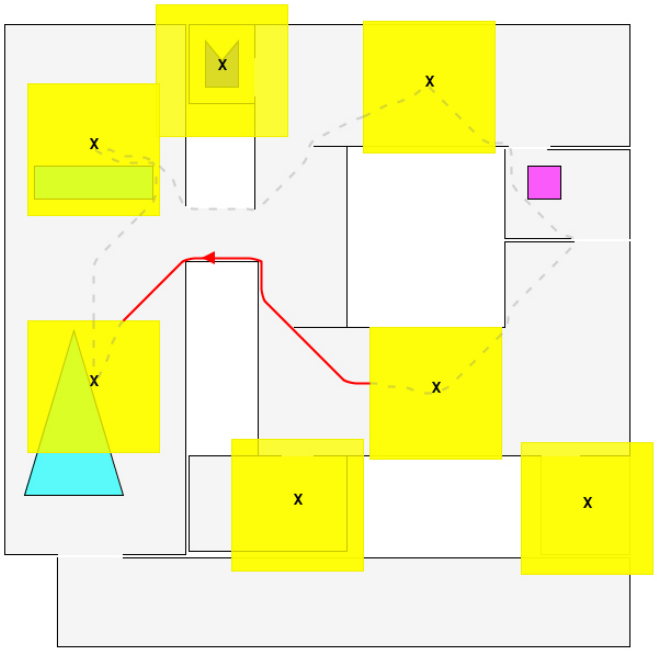

Abbildung 34: Visualisierung einer Teilstrecke

11.2.4 Systemtests

Als Testbesucher wurde dieselben Besucherdatensätze wie im Kapitel *10.2.3 Tests* beschrieben verwendet. Mithilfe des Scripts `recordals_with_ext_info_per_visitor.sql` (Verzeichnis `08 SQL Queries`) können die Recordals eines Besuchers verglichen werden:

Visitor_id	Kommentar	OK?
378	Meisten Recordals (82)	✓
238	Zweitmeisten Recordals (74)	✓
90	Mittlere Menge Recordals (38)	✓
16	Mittlere Menge Recordals (36)	✓
231	Wenige Recordals (10)	✓
141	Wenige Recordals (8)	✓
337	Sehr wenige Recordals (2)	✓
4	Sehr wenige Recordals (2)	✓

Abbildung 35: Systemtestprotokoll User Story 25

11.2.5 Fazit

Die Tatsache, dass eine Teilstrecke nicht mehr nur vom einen Reader zum anderen führt, hat uns in dieser User Story einige Schwierigkeiten gebracht. Die Berechnung der Wege findet zwar immer noch zwischen den Wegpunkten der Reader statt, doch die Teilstrecken müssen anschliessend separat berechnet werden. Letztlich haben wir aber auch all diese Schwierigkeiten lösen können und sind zu einem guten Resultat gekommen.

Die gestrichelte Darstellung der nicht aktiven Teilstrecken ist als sehr positiv empfunden worden und hat für uns praktisch keinen Mehraufwand bedeutet.

11.3 User Story 26: Zeiten zum Weg einer Person

Nachdem die vorhergehende User Story das Berechnen und Visualisieren der einzelnen Teilstrecken abdeckte, geht es nun um die Darstellung der Zeiten zum Weg einer einzelnen Person. Konkret soll für jede Teilstrecke die Anfangs- und die Endzeit dargestellt werden. Damit die Übersichtlichkeit nicht verloren geht werden die Zeiten nur im Teilstrecken-Modus und nur für die jeweils aktive Teilstrecke angezeigt.

11.3.1 Analyse

Wie bereits im Kapitel *11.2.2 Analyse & Design der neuen Lösung* erwähnt, erhält der Begriff Teilstrecke eine neue Definition. Eine Teilstrecke ist entweder die Strecke, in der sich der Besucher innerhalb eines Empfangsbereiches eines Readers aufhält oder die Strecke zwischen zwei Readern.

Damit nun aber die Zeiten korrekt an den Start- und Endpunkt der Teilstrecke gezeichnet werden können, muss festgestellt werden, aus welcher Richtung ein Weg daher kommt. Aus diesem Grund wurde der Empfangsbereich eines Readers in vier verschiedene Sektoren aufgeteilt, wie es die folgende Abbildung visualisiert:

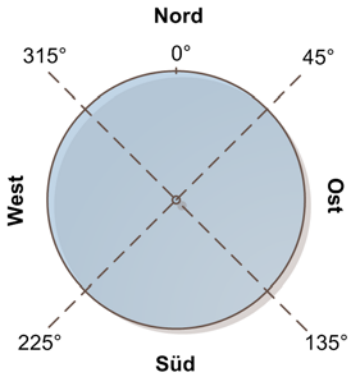

Abbildung 36: Aufteilung der Reader in vier Sektoren

Dank dieser Aufteilung kann später entschieden werden, an welcher Stelle des Eintrittspunktes eines Readers, ein Label zu positionieren ist. Kommt ein Weg von Ost oder Süd in einen Reader Bereich, so wird der Text entweder ober- oder unterhalb des Punktes platziert. Bei einem Weg aus dem Osten wird das Label rechts vom Punkt und bei einem Weg aus dem Westen links vom Punkt gezeichnet.

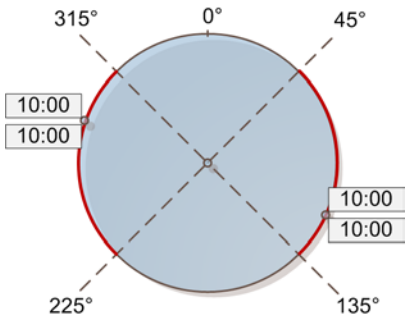

Abbildung 37: Positionierung der Labels aus Ost oder West

Kommt nun aber ein Weg von Nord oder Süd, so müssen die Labels links oder rechts vom Weg gezeichnet werden. Zusätzlich muss beachtet werden, dass bei einem Weg von Nor-

den das Label oberhalb des Punktes und umgekehrt bei einem Weg von Süden unterhalb des Punktes dargestellt wird.

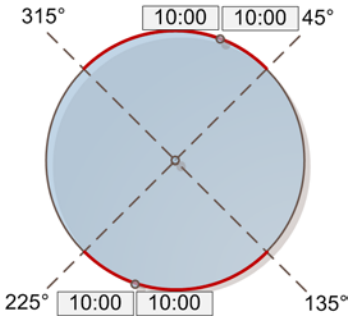

Abbildung 38: Positionierung aus Nord oder Süd

11.3.2 Design

Bei den Teilstrecken ist die Zeitpositionierung einfacher, da nur die aktive Teilstrecke gezeichnet wird und so weniger Etappen entstehen. Die folgenden Skizzen erläutern, wie die Zeiten an den Strecken dargestellt werden.

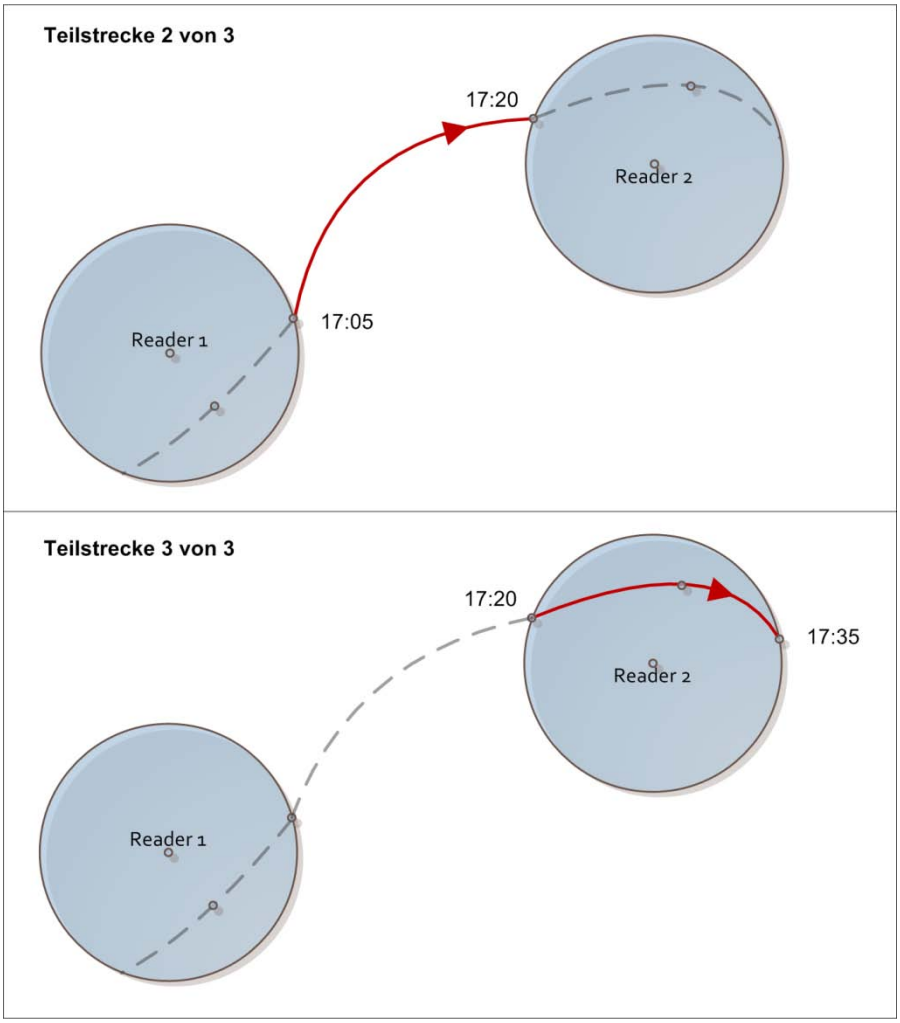

Abbildung 39: Design Darstellung der Zeiten zu Teilstrecken

11.3.3 Implementierung

Zur Berechnung der Ein- und Austrittswinkel in einem Reader Bereich wurde in der Klasse `WayManager` eine zusätzlich Methode `calculateAngle()` geschrieben, welche den Winkel zwischen zwei Punkten berechnet. Die Methode verlangt zwei Koordinaten und berechnet mittels Trigonometrie den einschliessenden Winkel zwischen den beiden Punkten. Anhand dieses Winkels und der zusätzlich neuen Methode `getOrientation()` kann anschliessend die Richtung (Nord, Ost, Süd, West) bestimmt werden, wonach die Positionierung der Zeiten sehr einfach bestimmt werden kann.

11.3.4 Unittests

Unittests

Die neu implementierten Methoden zur Berechnung der Winkel werden durch die folgenden Testklasse abgedeckt: `WayManagerTester`

Systemtests

Folgende Tests wurden für diese User Story durchgeführt:

Testfall	Erwartetes Resultat	OK?
Besucher auswählen und Teilstrecke suchen, die von Norden in einen Reader Bereich läuft.	Die zugehörige Zeit erscheint links der Linie.	✓
Besucher auswählen und Teilstrecke suchen, die von Osten in einen Reader Bereich läuft.	Die zugehörige Zeit erscheint rechts unterhalb der Linie.	✓
Besucher auswählen und Teilstrecke suchen, die von Süden in einen Reader Bereich läuft.	Die zugehörige Zeit erscheint links der Linie.	✓
Besucher auswählen und Teilstrecke suchen, die von Westen in einen Reader Bereich läuft.	Die zugehörige Zeit erscheint links unterhalb der Linie.	✓
Besucher auswählen und Teilstrecke suchen, die aus einem Reader in Richtung Norden herausläuft.	Die zugehörige Zeit erscheint rechts der Linie.	✓
Besucher auswählen und Teilstrecke suchen, die aus einem Reader in Richtung Osten herausläuft.	Die zugehörige Zeit erscheint links oberhalb der Linie.	✓
Besucher auswählen und Teilstrecke suchen, die aus einem Reader in Richtung Süden herausläuft.	Die zugehörige Zeit erscheint rechts der Linie.	✓
Besucher auswählen und Teilstrecke suchen, die aus einem Reader in Richtung Westen herausläuft.	Die zugehörige Zeit erscheint rechts oberhalb der Linie.	✓

Tabelle 20: Testfälle User Story 28

11.3.5 Fazit

Als wir beim Sprint Planning Meeting diese User Story schätzen sollten, haben wir die Tücken dieses Features ziemlich unterschätzt. Wir dachten, wir könnten zu jedem Endpunkt einfach die dazugehörige Zeit zeichnen und damit wäre die Sache erledigt.

Bei der Umsetzung sind dann aber diverse kleine Hindernisse aufgetreten, die uns das Leben erschwerten. Angefangen hatte es bei der Zuordnung der korrekten Zeit zum jeweiligen Endpunkt. Durch die noch nicht korrekte Sortierung der Quelldaten haben wir hier einige Zeit benötigt, bis wir einmal die Daten so bereinigt und sortiert hatten, dass eine korrekte Zuordnung überhaupt gemacht werden kann. Hinzu kam auch noch die Berechnung der Platzierung der Texte. Die Analyse zur Positionierung war zwar korrekt, doch wir unterschätzten die Tatsache, dass teilweise die Reader auch sehr eng beieinander stehen können und so Überlappungen entstehen.

Doch alles in Allem ist es uns gelungen diese User Story sauber und komplett umzusetzen, auch wenn wir einen gewissen Mehraufwand in Kauf nehmen mussten.

11.4 Sprint Retrospektive

Positive Aspekte des Sprints

+ Gute Zusammenarbeit auch über die Grenzen des eigenen Studienarbeits-Teams. Auftretende Fragen & Probleme konnten immer sehr schnell bearbeitet werden.
+ Kritiken bzw. Anregungen werden ernst genommen und gleich umgesetzt.
+ Es stellte sich als positiv heraus, dass am Sprint Planning Meeting nur die Zeiten für die kompletten User Storys geschätzt wurden, und nachher jedes Team selbstständig die detaillierten Tasks schätzen konnte.
+ Guter Fortschritt der gesamten Applikation

Negative Aspekte des Sprints

– Verkürzter Sprint wurde unterschätzt. Der fehlende halbe Tag hätte gut gebraucht werden können.
– Kurz vor Ende des zweiten Sprints sind noch Probleme aufgetaucht, welche uns zeitlich in Verzug brachten und zusätzlich noch Aufwand im folgenden Sprint bringen wird. Wäre nicht so tragisch, wenn es früher aufgefallen wäre.
– Am Ende des Sprints hatte es noch etwas zu viele offene Tasks bzw. TODOs im Code.
– Entwicklungsspezifische Probleme, Designentscheide u.ä. wurden immer nur bilateral besprochen, obwohl es teilweise auch das komplette Scrum Team interessiert hätte.

Verbesserungsmassnahmen für nächsten Sprint

Abgeleitet aus den negativen Punkten des Sprints, wurden folgende Massnahmen zu Verbesserung des nächsten Sprints getroffen:

- Allgemein wurde noch einmal erwähnt, dass wir darauf achten, eine User Story lieber zuerst komplett abzuschliessen, bevor man bereits mit der nächsten anfängt.
- Zu Beginn des nächsten Sprints erhält jeder Entwickler 1 Story Point um alle offenen Tasks des vorhergehenden Sprints zu bereinigen.
- Falls der Bedarf da ist irgendwelche Entscheide, Probleme oder sonstige Entwicklerfragen im Plenum zu diskutieren, so werden wir in Zukunft direkt im Anschluss an das Daily Meeting ein kurzes Entwicklermeeting einberufen.

12 Sprint 3

12.1 Sprint Tasks

ID	Beschreibung	Soll	Ist
23-1	Weg einer Person korrekt animiert: Analyse & Internes Design (Threading)	4.0	6.0
23-2	Weg einer Person korrekt animiert: Implementierung	7.0	16.0
23-3	Weg einer Person korrekt animiert: Systemtest	3.0	5.0
23-4	Weg einer Person korrekt animiert: Dokumentation	2.0	4.0
56-3	Auswahl einer Einzelperson: Analyse & Externes Design	3.0	2.0
56-4	Auswahl einer Einzelperson: Implementierung	9.0	6.5
56-5	Auswahl einer Einzelperson: Systemtest	2.0	0.5
56-6	Auswahl einer Einzelperson: Dokumentation	2.0	3.0
27-1	Geschwindigkeit bei animierten Weg veränderbar: Externes Design	2.0	2.0
27-2	Geschwindigkeit bei animierten Weg veränderbar: Implementierung	4.0	11.0
27-3	Geschwindigkeit bei animierten Weg veränderbar: Dokumentation	2.0	2.5
		40.0	**58.5**

Tabelle 21: Tasks zum Sprint 3

12.2 User Story 23: Weg einer Person korrekt animiert

In dieser User Story soll der Weg einer Person animiert dargestellt werden. Die Geschwindigkeit der Animation soll in Echtzeit sein, später kann diese jedoch um einen Faktor verändert werden. Die Geschwindigkeitsberechnung erfolgt pro Teilstrecke.

12.2.1 Externes Design Weganimation

Beim Design für die Weg-Animation wurden im Vergleich zur alten Lösung zwei Neuerungen eingebracht: Der erste Punkt ist wieder die gestrichelte Darstellung des vorhergehenden Weges. In der alten Applikation hatte man keinen Anhaltspunkt, wo der animierte Weg hinführt. In der neuen Variante sieht man danach ständig, wo der Weg noch entlang gehen wird. Eine weitere Neuerung ist die Darstellung der Zeiten zum Weg. Zum einen wird am Startpunkt die Startzeit angezeigt und zum anderen sieht man an der aktuellen Position des Besuchers immer die aktuelle Uhrzeit. Dies soll dem Anwender ein besseres Gefühl für die Geschwindigkeit und die Aufenthaltsdauer des Besuchers geben.

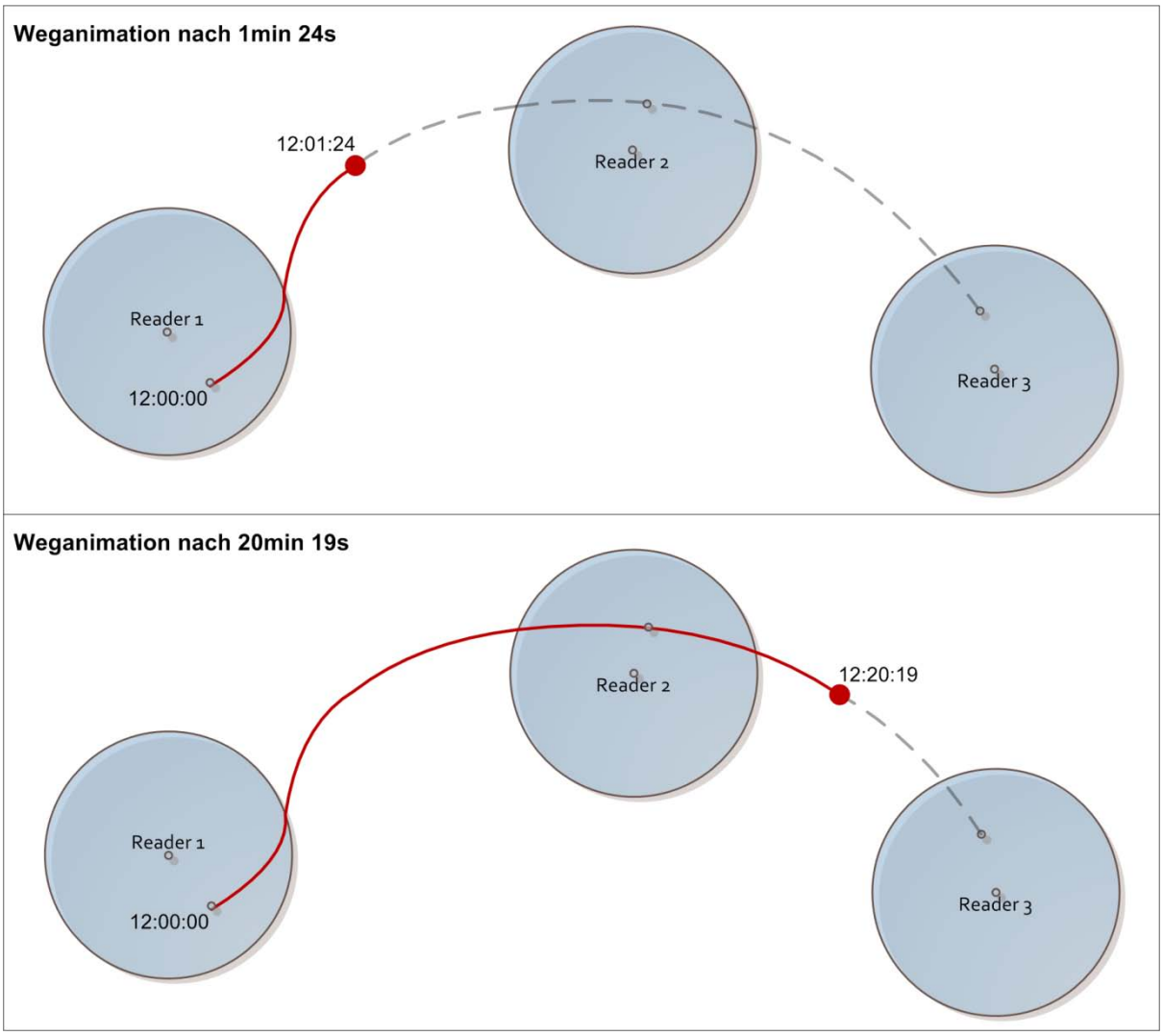

Abbildung 40: Externes Design Weg-Animation

12.2.2 Implementierung: Animation / Threading

Das Starten und Stoppen einer Weganimation stellt dem `ThreadManager` neue Anforderungen, so dass Objekte der Klasse `java.lang.Runnable` automatisch periodisch aufgerufen und zugleich pausiert werden können. Des Weiteren müssen die Threads beim Beenden der Applikation sauber terminiert und aufgeräumt werden. Dazu wurden neue Klassen eingeführt:

Klassenname	Kurzbeschreibung
PeriodicRunnable	Das `PeriodicRunnable` ist eine Implementierung von `java.lang.Runnable`. Es kapselt ein weiteres `Runnable` und führt dieses Periodisch aus. Zwischen den Perioden kann es gestoppt werden, so dass z.B. beim Beenden der Applikation die Threads entsprechend aufgeräumt werden können.
TimeUpdateRunnable	Das `TimeUpdateRunnable` aktualisiert die Zeit der `WindowSettings`, damit eine Animation dargestellt werden kann.

Tabelle 22: Beschreibung der Threading Klassen

Das Klassendiagramm in *Abbildung 41* visualisiert die Zusammenarbeit der einzelnen Komponenten:

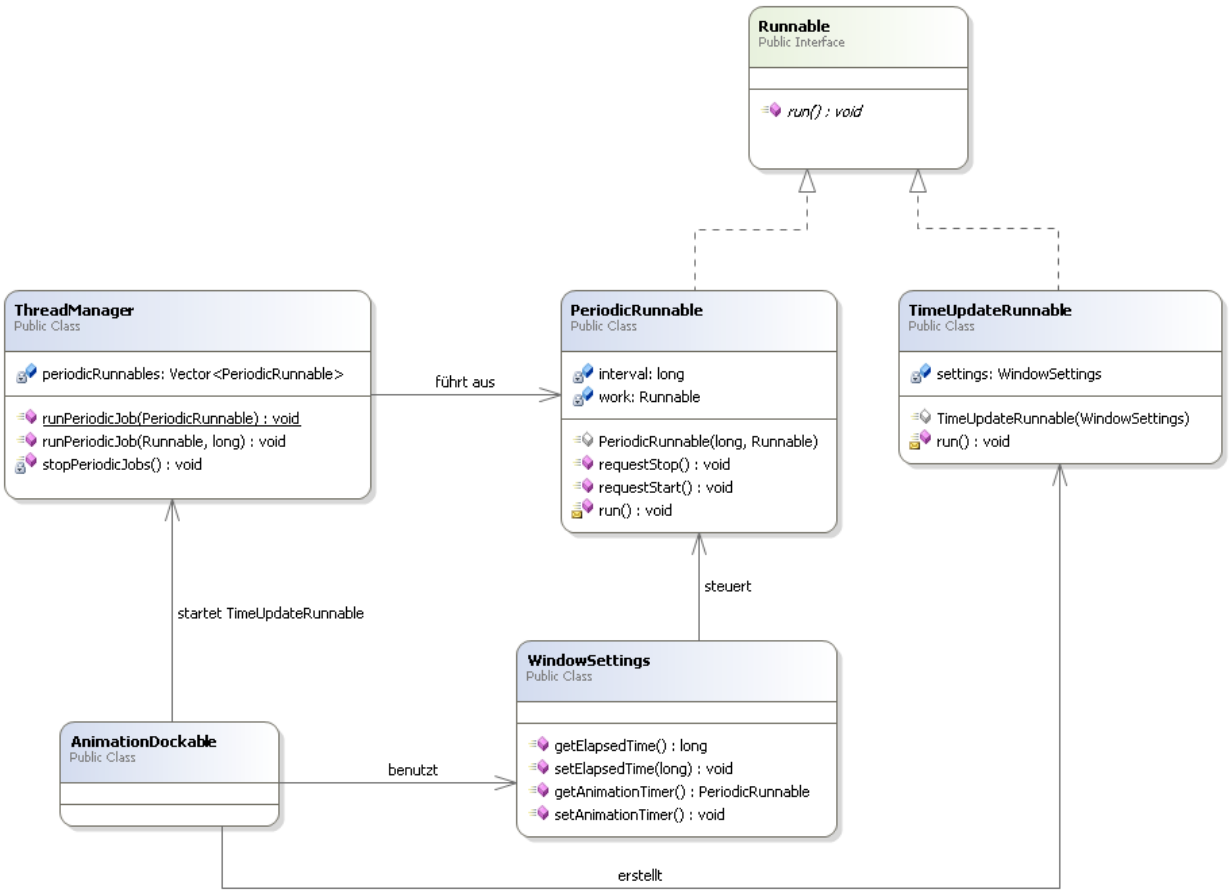

Abbildung 41: Klassendiagramm Weganimation & Thread-Handling

Zur besseren Verständlichkeit visualisiert *Abbildung 42* mit einem Sequenzdiagramm den Ablauf vom Erstellen einer Aufgabe (`TimeUpdateRunnable`), welche periodisch in einem separaten Thread wiederholt werden soll.

Das `AnimationDockable` übergibt dem `ThreadManager` lediglich die Aufgabe als `Runnable` (`TimeUpdateRunnable`), wobei der `ThreadManager` dieses `Runnable` in ein `PeriodicRunnable` packt, welches der `ExecutorService` (= Java interner Threadmanager) ausführt. Da der `ThreadManager` intern eine Liste mit allen `PerodicRunnables` führt, können diese jederzeit unterbrochen werden. Das `PeridicRunnable` kapselt lediglich ein `Runnable` aus der Java API, so können beliebige Java Threads periodisch ausgeführt werden.

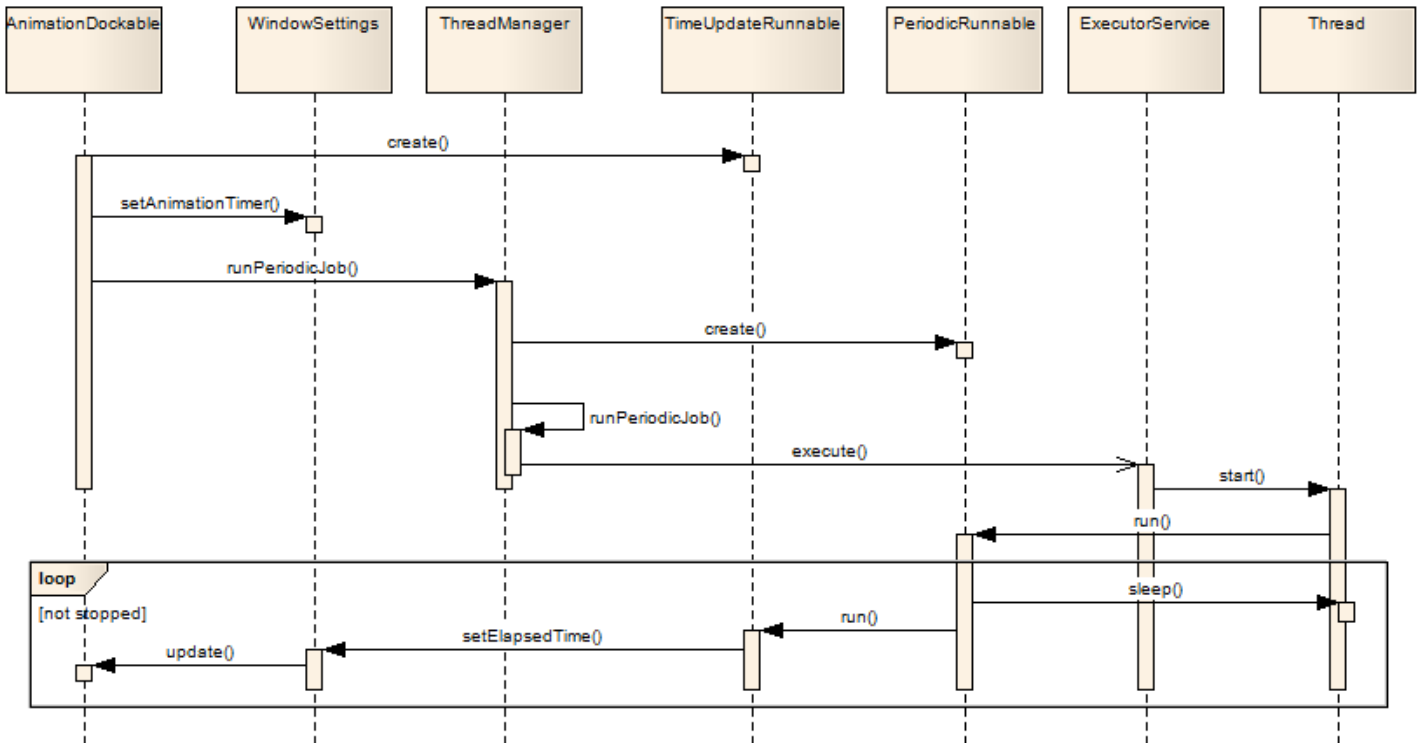

Abbildung 42: Sequenzdiagram des Thread-Handling eines PeriodicRunnable

12.2.3 Implementierung: Visualisierung der Wegstrecken

Die Visualisierung der Wegstrecken wurde in den früheren User Stories so entwickelt, dass die Wege direkt in der Klasse `TrackingPanel` gezeichnet wurden. Das `TrackingPanel` übernimmt die Darstellung der Wegvisualisierung. Durch die zusätzliche Visualisierungsvariante der animierten Wege, wurde diese Klasse sehr gross und komplex. Aus diesem Grund wurde entschieden, das Zeichnen der Wege auszulagern und die verschiedenen Arten der Weg-Darstellungen mittels *State Pattern* umzusetzen. *[GAMMA95]*

In *Abbildung 43* wird die Struktur des umgesetzten Patterns als Klassendiagramm gezeigt. In den früheren Versionen gab es lediglich das `TrackingPanel`, welches über das *Observer Pattern [GAMMA95]* darüber benachrichtigt wurde, dass es sich neu zeichnen soll. Die ganze Logik des Zeichnens war in der `paintVisualisation()` Methode implementiert.

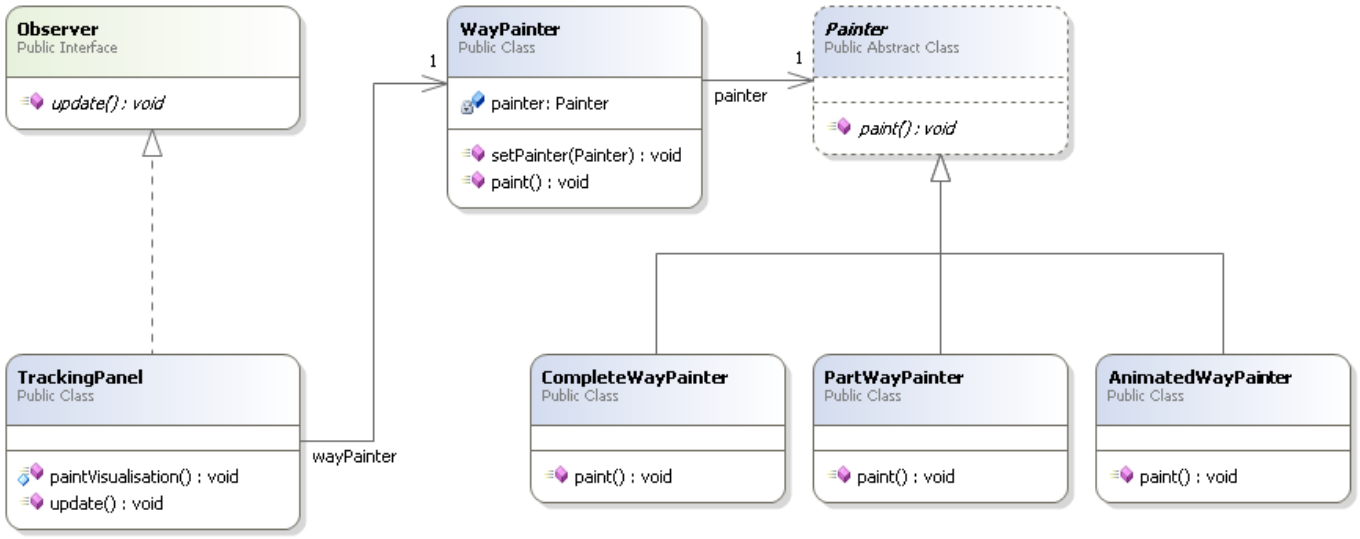

Abbildung 43: Klassendiagramm WayPainter

Neu wurde der ganze rechte Teil in die Struktur eingefügt. Das `TrackingPanel` erhält nun eine Referenz auf einen `WayPainter`. Falls nun das `Panel` über den `Observer` aufgefordert wird, sich neu zu zeichnen, so delegiert es das Zeichnen weiter an den `WayPainter`. Dieser `WayPainter` ruft dann die `paint()` Methode des aktuell gesetzten `Painter`. Welcher `Painter` gerade aktiv ist, wird indirekt vom Benutzer bestimmt, indem er in der rechten Navigation die Art der Weg-Visualisierung auswählt.

12.2.4 Systemtest

Um diese User Story zuverlässig durch Systemtests überprüfen zu können, wurden zuerst einige Testdaten direkt von der Datenbank abgefragt. Das folgende SQL Query liefert für das Testen alle Besucher mit ihrer Aufenthaltsdauer und der Start- bzw. Endzeit ihres Aufenthaltes:

```
SELECT  visitor_id AS `Visitor`,
        TIMEDIFF(MAX(time), MIN(time)) AS `Aufenthaltsdauer`,
        COUNT(*) AS `Anzahl Recordals`,
        MIN(time) AS `Startzeit`,
        MAX(time) AS `Endzeit`
FROM recordal
GROUP BY visitor_id
ORDER BY `Aufenthaltsdauer` DESC, `Anzahl Recordals` DESC
```

Das Query liefert folgendes Ergebnis:

Visitor	Aufenthaltsdauer	Anzahl Recordals	Startzeit	Endzeit
9	04:33:13	74	2009-05-08 10:55:45	2009-05-08 15:28:58
10	04:32:57	68	2009-05-08 10:56:10	2009-05-08 15:29:07
368	03:01:51	50	2009-06-01 12:33:59	2009-06-01 15:35:50
369	03:00:53	52	2009-06-01 12:34:58	2009-06-01 15:35:51
47	02:58:25	60	2009-05-10 12:23:25	2009-05-10 15:21:50
48	02:58:11	48	2009-05-10 12:23:41	2009-05-10 15:21:52

Abbildung 44: Query Resultat für alle Visitor mit deren Zeiten

Mit diesen Daten wurde die Wegvisualisierung auf ihre Korrektheit der Start- bzw. Endzeit und deren Aufenthaltsdauer geprüft. Zurzeit ist es nicht möglich, Visualisierungen über mehrere Etagen zu verfolgen. Darum verharrt die Wegvisualisierung bei einem Etagenwechsel beim Ein- bzw. Austrittspunkt der Etage solange, wie die Etage verlassen wurde. Das korrekte Verhalten wird in einer späteren User Story implementiert.

12.2.5 Fazit

Die Aufteilung der Zeichnungslogik in die verschieden Painter-Klassen hat sich als sehr gut herausgestellt. Die Grösse der einzelnen Klassen hat sich deutlich reduziert, welches zu einem wartbareren Code führt. Zusätzlich ist es für nicht involvierte Personen viel einfacher den Programmfluss zu erkennen und den Code nachzuvollziehen.

12.3 User Story 56: Auswahl einer Einzelperson

Im ersten Sprint des Projektes wurde diese User Story bereits einmal in Angriff genommen (siehe Kapitel *10.3 User Story 56: Auswahl einer Person*). Damals wurde entschieden nur einen ersten Prototypen zu implementieren. Beim Öffnen der Bewegungsverfolgung erscheint ein Popup-Dialog, welcher die Auswahl einer Einzelperson zulässt. Neu soll beim Öffnen der Wegvisualisierung kein Dialog erscheinen, sondern direkt das Visualisierungsfenster öffnen. Dieses soll noch kein Gebäude zeichnen, sondern an dieser Stelle eine Meldung darstellen, dass zuerst Daten zur Visualisierung gewählt werden sollen. Die eigentliche Besucherauswahl soll auch als Dockable implementiert werden, so dass diese immer in der Navigationsleiste auf der rechten Seite ersichtlich ist. Zusätzlich muss die Möglichkeit bestehen, jederzeit das Museum, den Museumszustand und die gewünschte Etage auszuwählen.

12.3.1 Analyse und Design

Die *Abbildung 45* zeigt wie die Auswahl des Museums von der Auswahl der Person getrennt und jeweils in einem eigenen Dockable implementiert werden soll. Neben der besseren Übersichtlichkeit hat dies auch zum Vorteil, dass die Komponente für die Auswahl des Museums im Fenster für die Statistiken wiederverwendet werden kann.

Auf Hinweis des Product Owners wurde beim Design auch berücksichtigt, dass zur jeweils selektierten Person immer alle ihre Detaildaten angezeigt werden. Man erhofft sich durch die Darstellung der demographischen Daten, dass das dargestellte Wegverhalten besser interpretiert werden kann.

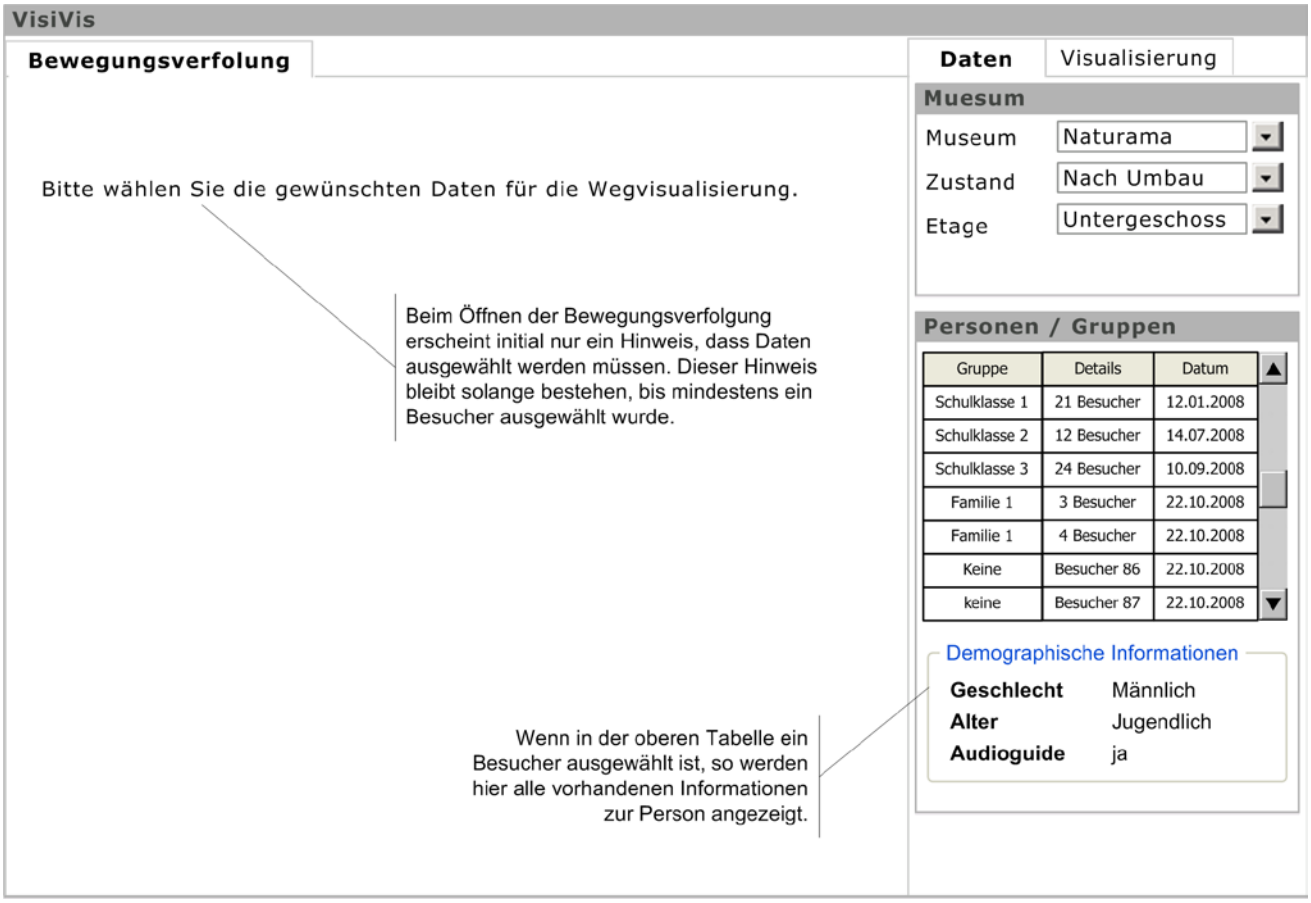

Abbildung 45: Design Auswahl Einzelperson

12.3.2 Fazit

Die Separierung der Museumsauswahl hat sich als sehr gut herausgestellt, da es nun tatsächlich in der genau gleichen Form in den unterschiedlichen Visualisierungen angewendet werden kann, ohne dass irgendwelche Spezialfallunterscheidungen im Code durchgeführt werden mussten.

Durch die Verschiebung der Personenauswahl auf die rechte Seite der Applikation, konnten zum ersten Mal ein paar Performance Tests an der laufenden Applikation durchgeführt werden. Man kann direkt Schritt für Schritt vom einen Besucher zum anderen Wechseln und die Wege werden immer dynamisch berechnet und dargestellt. Der ganze Algorithmus arbeitet so schnell, dass kaum eine Verzögerung spürbar ist, ob der Weg nun zuerst berechnet werden muss, oder ob dieser direkt aus dem Cache gelesen wird.

12.4 User Story 27: Geschwindigkeit bei animierten Weg veränderbar

Diese User Story umfasst das Design und Implementierung des Dockable, mit der ein Benutzer die Animation steuern kann. Dies beinhaltet das Starten, Stoppen, Pausieren und das Verändern der Geschwindigkeit.

12.4.1 Externes Design

Die Idee hinter der Steuerung ist analog zu einem CD-Player. Speziell ist, dass die Zeit nicht relativ, sondern als Uhrzeit angezeigt wird. Mit einem Slider kann durch die Zeit navigiert werden.

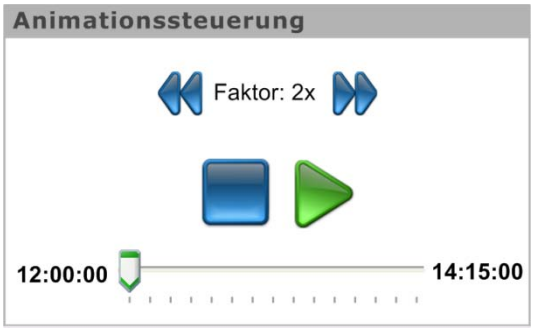

Abbildung 46: Entwurf

Abbildung 47: Umsetzung

Erste Tests haben gezeigt, dass Einerschritte bei dem Faktor sehr ineffizient sind. Darum wurde der Faktor so angepasst, dass dieser sich beim Spulen verdoppelt, respektive halbiert.

12.4.2 Implementierung

Die Implementierung beschränkte sich grundsätzlich auf das User Interface. Die eigentliche Logik ist bereits implementiert (Kapitel *12.2.2 Implementierung: Animation / Threading*).

Die `WindowSettings` beinhalten bereits alle benötigten Objekte, so kann in den `ActionListener` Implementierungen der Buttons die entsprechenden Methoden zum Starten, Stoppen oder Verändern der Geschwindigkeit aufgerufen werden. Die `AnimationDockable` Klasse besitzt keine Logik (ausgenommen UI spezifische).

12.4.3 Fazit

Die Positionierung der Elemente auf dem Dockable stellte sich mit dem `GridBagLayout` als sehr umständlich heraus und endete in einem etwas mühsameren „Try&Error“ Verfahren. Eine wirklich bessere Alternative steht für Swing nicht zur Verfügung. *Separation of Concerns [URL09]* kam bei diesem Sprint schön zur Geltung: In den `ActionListeners` der Dockables musste keine Business Logik implementiert sondern lediglich eine Methode aufgerufen werden.

12.5 Sprint Retrospektive

Positive Aspekte des Sprints

+ Bisherige Wegberechnung und Caching Implementierung konnte sehr gut für die neuen Anforderungen angewendet werden, wobei auch die Performanz sehr zufriedenstellend ist.
+ Es war gut, dass sich der Product Owner nach dem Sprint Meeting die Zeit genommen hat um die Applikation durchzutesten, Feedback dazu gab und diese offenen Punkte direkt als Tickets im Trac geführt wurden.
+ Die Idee, dass man für Bugfixing und Umsetzung der Tickets durch Review vom Product Owner Zeit einplante, fanden wir sehr hilfreich.

Negative Aspekte des Sprints

– Das Zeichnen der animierten Wege mittels Java2D war aufwändiger als angenommen, da wir abgerundete Pfade verwenden, welche nicht Pixel um Pixel gezeichnet werden können.
– Java UI Programmierung ist oftmals mühsam und zeitaufwändig.
– Am Ende des Sprints waren wieder zu viele offene Tätigkeiten vorhanden. Das Ziel wäre, dass man am Ende jedes Sprints immer alle Tasks zu 100% abgeschlossen hat. Dies lag eher am Unterschätzen des Aufwandes und nicht an mangelnder Effizienz. Das Ganze hat nun zur Folge, dass zu Beginn des nächsten Sprints wieder ein Aufräumtag eingeplant werden muss.
– Es wurde zu viel Code eingecheckt, der Checkstyle und Findbugs Warnungen enthielt.
– Bis anhin wurde dem Scrum Master noch kein schriftliches Feedback über die eingesetzten Scrum und XP Praktiken gegeben.

Verbesserungsmassnahmen für nächsten Sprint

Abgeleitet aus den negativen Punkten des Sprints, wurden folgende Massnahmen zu Verbesserung des nächsten Sprints getroffen:

- Um in Zukunft die extra eingeplanten Aufräumtage zu vermeiden und ein Sprint jeweils zu 100% pünktlich abzuschliessen, wurde das Team aufgefordert, den nächsten Sprint eher ein wenig grosszügig zu schätzen und auch eine Reservezeit einzuplanen. Für jedes Team wird mindestens noch ein Next Task bestimmt, welcher dann in Angriff genommen werden kann, falls man trotzdem schneller vorankommt als geplant.
- Um unnötige Checkstyle und Findbugs Warnungen auf dem Hudson Build Server in Zukunft zu vermeiden, wurde das Team nochmals dazu aufgefordert, immer vor dem Einchecken lokal nochmals selber zu überprüfen, ob man nicht irgendwelche Warnungen produziert hat. Dies soll im besten Fall gleich direkt mit den lokal installierten Eclipse Plug-Ins verifiziert werden oder ansonsten sicher einmal pro Tag auf dem Build Server.
- Der Scrum Master hat das Scrum Team gebeten sich einen halben Tag Zeit zu nehmen und dabei einen Zwischenbericht zu verfassen. Darin soll allgemein über das ganze Projekt reflektiert und die einzelnen Scrum bzw. XP Praktiken bewertet werden. (siehe Kapitel *18 Zwischenberichte*)

13 Sprint 4

13.1 Sprint Tasks

ID	Beschreibung	Soll	Ist
28-1	Wege über mehrere Etagen darstellen: Analyse & Design	4.0	10.5
28-2	Wege über mehrere Etagen darstellen: Implementierung	24.0	33.0
28-3	Wege über mehrere Etagen darstellen: Unittests, Systemtests	4.0	4.0
28-4	Wege über mehrere Etagen darstellen: Dokumentation	8.0	6.0
29-1	Auswahl mehrerer Einzelpersonen: Weitere Variante für Design	3.0	4.0
29-2	Auswahl mehrerer Einzelpersonen: Implementierung	10.0	5.0
29-3	Auswahl mehrerer Einzelpersonen: Dokumentation	3.0	0.0
90-4	Bereinigung Sprint 3	12.0	9.5
		68.0	**72.0**

Tabelle 23: Tasks zum Sprint 4

13.2 User Story 28: Wege über mehrere Etagen darstellen

Bis anhin konnte der Wechsel einer Etage des Besuchers nicht bestimmt werden, da keine Etagentransferpunkte definiert waren. Das Ziel dieser User Story ist die Wegevisualisierung über mehrere Etagen. Es soll also ersichtlich sein, wo ein Besucher die Etage wechselt. Des Weiteren soll der Animation- und Teilstreckenmodus ermöglichen, dass der Wechsel der Etage automatisch im selben Fenster erfolgt, wenn der Besucher die Etage wechselt.

13.2.1 Analyse

Sobald mehrere Besucher visualisiert werden, stellt sich die Frage, wann die Etage gewechselt werden soll. Die einfachste Variante wäre ein manueller Wechsel der Etage, so dass der Benutzer bei Bedarf manuell die Etage wechselt. Im Verfolgungsmodus ist jedoch eine Option erwünscht, bei der ein Besucher ausgewählt werden kann, welcher automatisch verfolgt wird. Das heisst, unabhängig von den anderen Besuchern wird die Etage jeweils auf den zu verfolgenden Besucher gewechselt.

Gemäss Sprint Planning Meeting war die Idee, dass zwei verschiedene Visualisierungsmodi entwickelt werden, je eine für die Verfolgung eines Besuchers und eine für mehrere Besucher. Diese Idee gefiel uns bei der genaueren Analyse aus folgenden Gründen nicht mehr: Für den Benutzer ist es am einfachsten, wenn er an einer zentralen Stelle die Besucher bzw. Besuchergruppen auswählt. Je nachdem wie viele Besucher ausgewählt wurden, können entsprechende Optionen im Animations-Dockable ausgewählt werden.

Beide Varianten basieren auf der Besucherauswahl, welche im Kapitel *13.3.1 Analyse* beschrieben wird.

Variante 1:

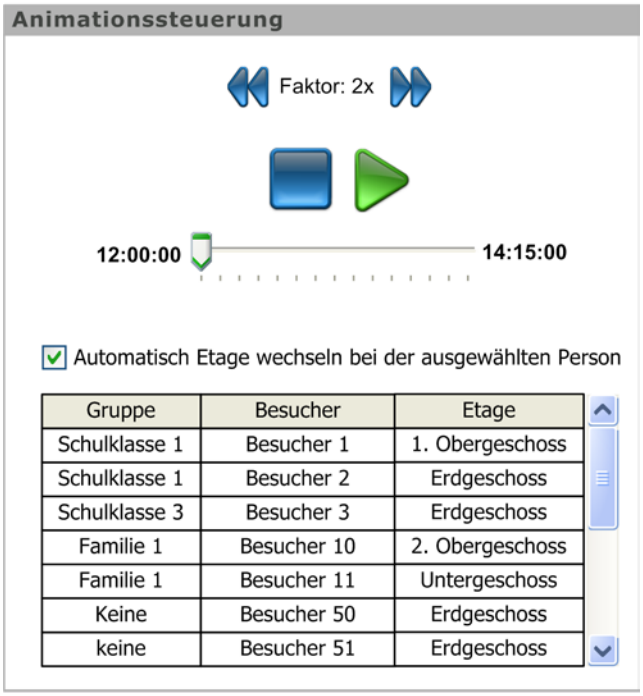

Abbildung 48: Variante 1 Dockable zur Animation mehrerer Besucher

Bei der ersten Variante erscheint unter der Animationssteuerung eine Tabelle aller ausgewählten Besucher. Eine Checkbox ermöglicht das Verfolgen der ausgewählten Person. Mit dieser Tabelle wird versucht, die Visualisation von einzelnen bzw. mehreren Besuchern zu vereinen. Sobald mehrere Besucher ausgewählt sind, kann mit dem Selektieren einer Tabellenzeile ausgewählt werden, welcher Besucher den Fokus erhält und visualisiert wird.

Variante 2

Beim Vergleichen der Besucherauswahl und der Tabelle fällt auf, dass viele Informationen doppelt vorhanden sind und unter Umständen für Verwirrung sorgen können. Vom Product Owner ist zusätzlich eine Visualisierung gewünscht, bei der mehrere Stockwerke gleichzeitig angezeigt werden. Daraus folgt eine neue Variante:

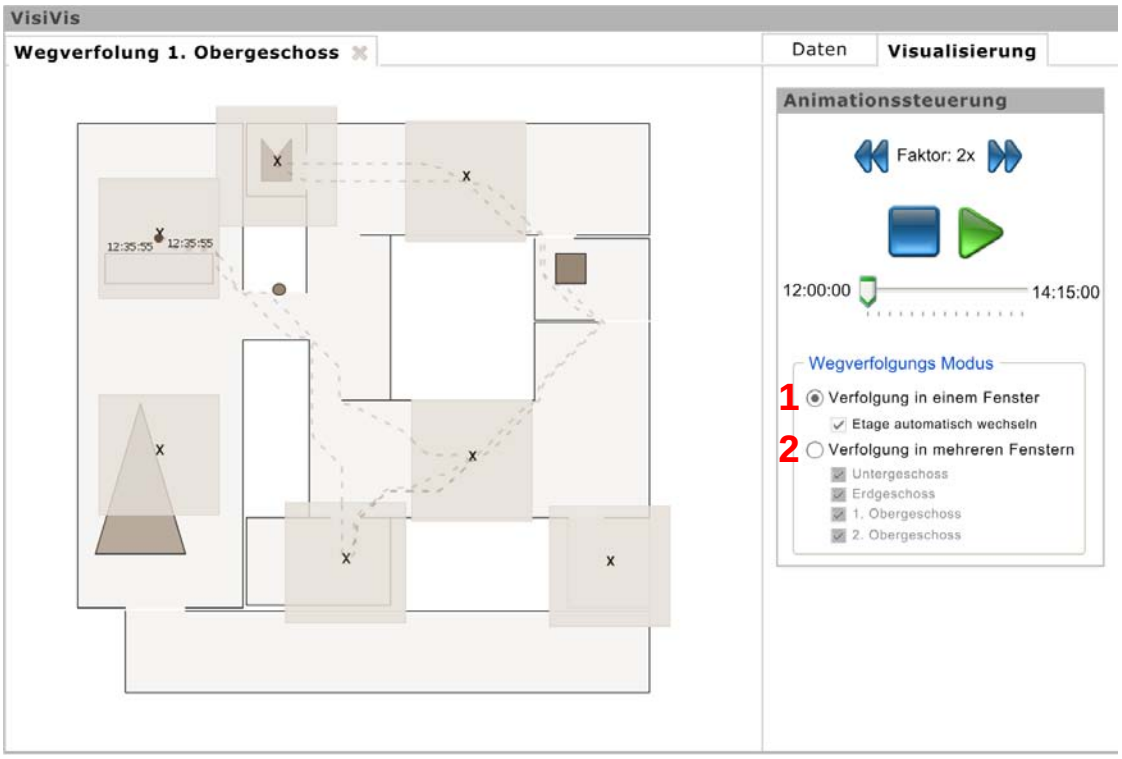

Abbildung 49: Variante 2 mit Wegverfolgung in einem Fenster

Bei dieser Variante erfolgt die Visualisierung sofern nur ein Besucher ausgewählt ist automatisch nur in einem Fenster [1]. Eine Option „Etage automatisch wechseln“ steht zur Verfügung, damit die Etage automatisch gewechselt wird.

Abbildung 50: Variante 2 mit Wegverfolgung in mehreren Fenstern

Sind mehrere Besucher ausgewählt, hat der Benutzer die Möglichkeit, die Verfolgung über mehrere Fenster zu starten [2]. Einzelne Stockwerke können mit den Checkboxen ein bzw. ausgeblendet werden. Die Möglichkeit, zwischen den Verfolgungsmodi „on-the-fly“ zu Wechseln gibt dem Benutzer viel Flexibilität, ohne dass das UI Überfüllt oder kompliziert wird.

Entscheid
Die Variante 1 ist für die Weganimation in einem Fenster eine gute Lösung. Jedoch wird es schwierig, sobald eine Visualisation über mehrere Fenster erfolgen soll. Es müsste einen weiteren Visualisierungsmodus eingeführt werden, dabei darf auf diesen nur gewechselt werden, wenn mehrere Besucher ausgewählt sind. Ein weiterer Negativpunkt ist das doppelte Darstellen von Besucherinformationen. Mit dem aktuellen Dockable-Design (Daten-/Visualisierungs Tab) hätte die Redundanz einen Vorteil: während der Animation muss für die Benutzerinformationen nicht immer auf das Daten-Tab gewechselt werden. Die Anordnung der Dockables und die Gestaltung der Tabs ist noch nicht definitiv und wird in einem späteren Sprint vom Scrum Team optimiert. Variante 2 wirkt einfacher zum Bedienen und bringt nicht so viele Nachteile mit sich. Aus diesem Grund haben wir uns für die *Variante 2* entschieden. In diesem Sprint wird jedoch nur die Logik für [1] Verfolgung in einem Fenster implementiert.

13.2.2 Implementierung

Die Implementierung dieser User Story erfordert einige Anpassungen, da nun die Wege jeweils direkt über alle Etagen berechnet werden müssen, wobei jeder Wechsel von einer Etage in die andere berücksichtigt werden muss.

FloorStay Objekt
Um einen Aufenthalt in einer Etage verständlicher zu modellieren, ist der Entscheid gefallen, alle Eigenschaften und Werte eines einzelnen Etagenaufenthaltes in einer eigens dafür erstellten Klasse zu Kapseln. Diese Klasse heisst `FloorStay` und sieht folgendermassen aus:

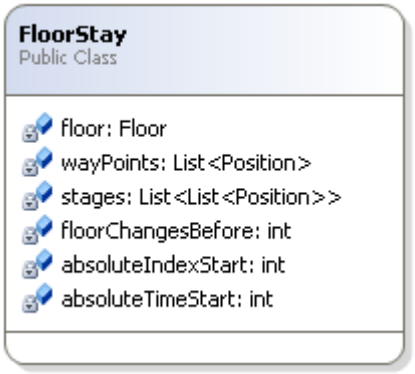

Abbildung 51: Klasse FloorStay

Was die Eigenschaften der einzelnen Felder sind und warum sie gebraucht werden, kann in der Javadoc entnommen werden. Kurz zusammengefasst enthält ein Objekt der Klasse `FloorStay` immer die dazugehörige Etage, alle Wegpunkte der RFID Reader, den Transferpunkt, wo der Besucher durchgelaufen ist und natürlich der komplett berechnete Weg aufgeteilt in seine Teilstrecken. Wichtig hierbei ist, dass ein solches Objekt für jeden einzelnen Aufenthalt in einer Etage erzeugt wird. Das heisst, wenn ein Besucher mehrmals in derselben Etage war, so wird jedes Mal ein `FloorStay` Objekt erzeugt.

Die *Abbildung 52* zeigt die konzeptuelle Sicht, wie diese `FloorStay` Objekte die externe Visualisierungs-Darstellung intern reflektieren. Auf der linken Seite der Grafik ist dargestellt, wie eine Wegvisualisierung über zwei Etagen dem Benutzer dargestellt wird. Rechts wird dann gezeigt, wie diese zwei Aufenthalte in den verschiedenen Etagen intern durch die zwei `FloorStay` Objekte abgebildet werden.

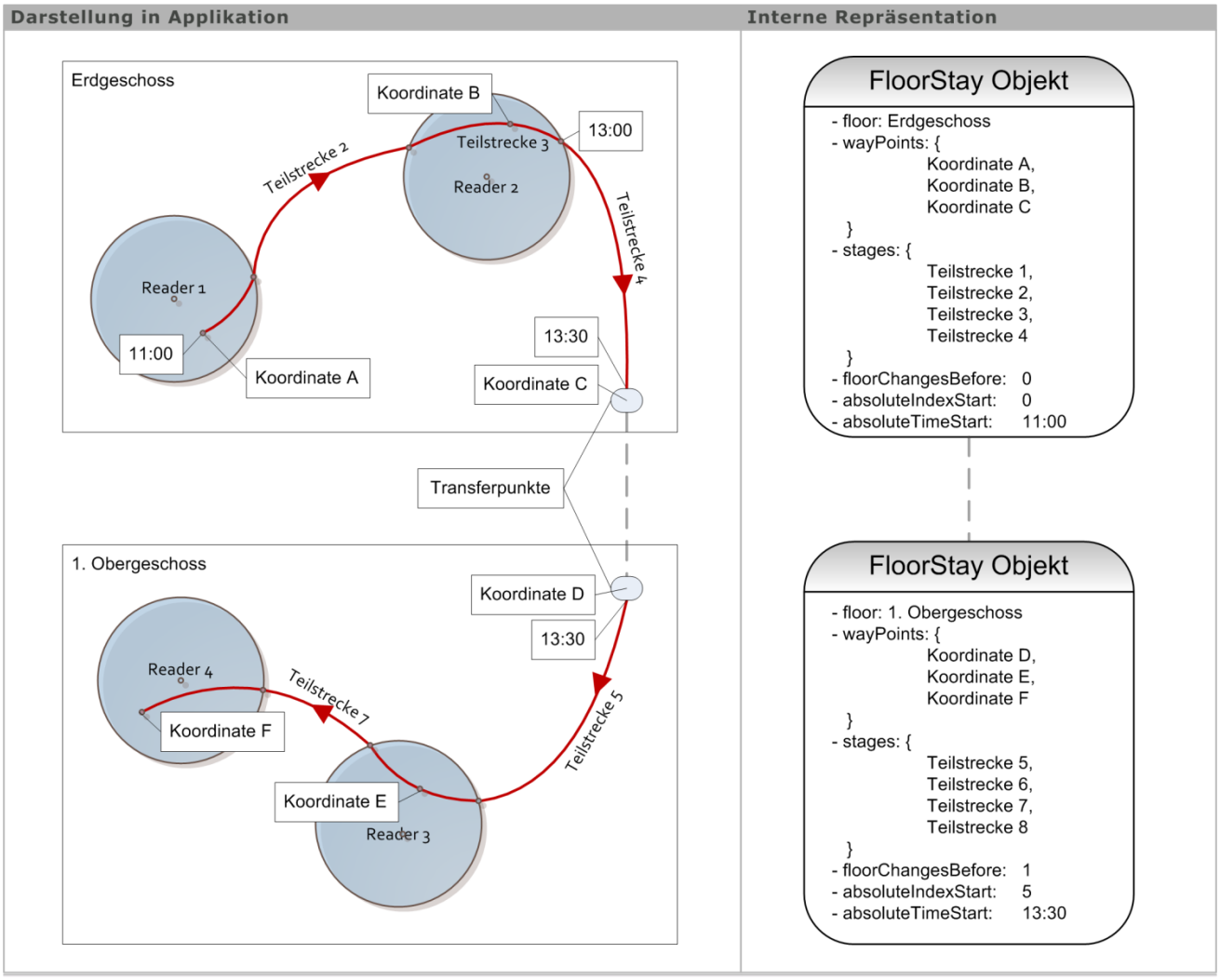

Abbildung 52: Konzept FloorStay Objekte

Implementierungsschritte

Die Berechnung und das Darstellen der Wege kann schematisch gesehen in die folgenden Schritte aufgeteilt werden:

1. `FloorStay` Objekt für jeden einzelnen Etagenaufenthalt erzeugen.
2. Bei jedem Etagenwechsel ein zusätzliches `Recordal` erstellen an der Stelle des Etagen Transferpunktes.
3. Zeit des künstlich erzeugten `Recordal` beim Transferpunkt berechnen.
4. Für jeden einzelnen Etagenaufenthalt (`FloorStay`) die Wegberechnung durchführen und danach der berechnete Weg im entsprechenden `FloorStay` Objekt speichern.
5. Gesamtzahl aller Teilstrecken über alle Etagen Aufenthalte hinweg bestimmen und abspeichern. Wird für den Etappenmodus benötigt.
6. Start- und Endzeit des Besuchers bestimmen und abspeichern.
7. Zeichnen: Über alle `FloorStay` Objekte iterieren und falls die Etage des `FloorStay` Objektes derjenigen Etage entspricht, die gerade aktiv ist, so wird das `FloorStay` Objekt dem aktuellen `WayPainter` übergeben, damit dieser den Weg zeichnen kann.

Künstlich erzeugtes Recordal bei Etagenwechsel
Wie schon im Implementierungsschritt 2 & 3 erläutert, muss bei jedem festgestellten Etagenwechsel ein Recordal künstlich hinzugefügt werden. Dieses Recordal muss genau an der Stelle des Etagentransferpunktes liegen. Da sich die Zeit bei einem Etagenübergang nicht genau bestimmen lässt, muss diese berechnet werden. Wie in *Abbildung 52* zu erkennen ist, wurde die Zeit beim Transferpunkt aus der letzten Zeit im Erdgeschoss und aus der ersten Zeit im 1. Obergeschoss berechnet. Der Einfachheit halber und aus dem Mangel von detaillieren Informationen wird die Zeit am Transferpunkt durch den Mittelwert der beiden Zeiten 13:00 und 14:00 berechnet und ist in diesem Beispiel 13:30 Uhr.

13.2.3 Tests

Unittests
Der in dieser User Story erstellte Code, wurde fast ausschliesslich in bereits bestehenden Klassen geschrieben. Das heisst, dazu mussten lediglich die bestehenden Unittest Klassen ergänzt bzw. abgeändert werden. Hier musste speziell der `WayManager` neu darauf getestet werden, dass die Wege über mehrere Etagen auch wirklich korrekt berechnet werden. Zu finden sind diese Unittest Anpassungen in der Klasse
`ch.hsr.ifs.visivis.visual.ui.controller.tracking.WayManagerTester`.

Die einzige neue Klasse in dieser User Story ist die Klasse `FloorStay`. Dafür wurde eine separate Test Klasse erstellt:
`ch.hsr.ifs.visivis.visual.pd.tracking.FloorStayTester`.

Systemtests
Um die Visualisierung über mehrere Etagen zuverlässig zu testen, werden aussagekräftige Testdaten benötigt. Mit dem folgenden SQL Query kann man den kompletten Weg eines beliebigen Besuchers abfragen. Dabei ist immer ersichtlich, zu welcher Zeit sich der Besucher in welcher Etage bei welchem RFID Reader aufgehalten hat. Anhand dieser Informationen kann danach mit Sicherheit gesagt werden, dass eine Wegvisualisierung über mehrere Etagen korrekt ist.

```
SELECT
       r.time as 'Zeit', f.name as 'Etage',
       td.trackingdevice_id as 'Reader ID',
       CONCAT_WS('/', cast(p.x as char), cast(p.y as char)) as 'Koordinaten'
FROM
       floorobject fo
       INNER JOIN floor f               ON f.id = fo.floor_id
       INNER JOIN recordal r            ON r.trackingdevice_id = fo.id
       INNER JOIN fo_trackingdevice td  ON td.trackingdevice_id = fo.id
       INNER JOIN fo_position p         ON p.id = td.waypoint_id
WHERE r.visitor_id = 8
ORDER BY r.time
```

Diese Query liefert für den Besucher mit der ID 8 das folgende Resultat:

Zeit	Etage	Reader Id	Koordinaten
2009-05-07 15:52:16	1.Obergeschoss	227	38/45
2009-05-07 15:52:16	1.Obergeschoss	227	38/45
2009-05-07 15:55:53	Untergeschoss	235	164/65
2009-05-07 15:58:07	Untergeschoss	235	164/65
2009-05-07 15:59:07	Untergeschoss	233	116/159
2009-05-07 16:00:41	Untergeschoss	233	116/159
2009-05-07 16:01:15	Untergeschoss	229	113/110
2009-05-07 16:02:23	Untergeschoss	229	113/110
2009-05-07 16:03:51	Erdgeschoss	230	178/74

Abbildung 53: Query Resultat für Abfrage des Weges von Besucher 8

Mit Hilfe des oben erwähnten SQL Queries wurden die Wege folgender Besucher verifiziert:

Visitor_id	Kommentar	OK?
378	Meisten Recordals, 5 Etagenwechsel	✓
238	Zweitmeisten Recordals, 12 Etagenwechsel, Aufenthalt in allen Etagen inkl. 2. Obergeschoss	✓
12	Mittlere Menge Recordals, nur 1 Etagenwechsel	✓
14	Mittlere Menge Recordals, 3 Etagenwechsel	✓
10	Viele Recordals, 5 Etagenwechsel	✓

Tabelle 24: Testresultate zu Systemtest von Visualisierung über mehrere Etagen

13.2.4 Fazit

Bereits die Analyse des bestehenden Codes und das Besprechen einiger möglicher Lösungen, wie die Anforderung implementiert werden könnte, haben wir zusammen an einem Notebook gemacht. So konnte innert kurzer Zeit eine gute Lösung erarbeitet werden. Diese neue Struktur in den bereits bestehenden Code einzuführen ist ein komplexes Unterfangen, da an einigen Stellen Änderungen vorgenommen werden müssen. Am Ende des vier Stündigen Pair Programming Einsatz konnten wir eine fast fertige Implementierung vorweisen, welche nur sehr wenige und einfache Bugs beinhaltete. Das Schöne an dem Pair Programming ist, dass bereits beim Programmieren Denkfehler aufgedeckt werden können, welche sonst erst beim Debuggen auffallen würden. Des Weiteren werden beide Entwickler mit dem Code vertraut, was bei solch zentralen Codestellen enorm wichtig ist.

Eine weitere Problematik in diesem Sprint sind die `WindowSettings`: Die Klasse besteht aus unzähligen Attributen. Der Animationsmodus schreibt alle 100 Millisekunden einen neuen Wert (die verstrichene Zeit) in die `WindowSettings`, worauf alle Observers notifiziert werden, obwohl die meisten Observers diese Information gar nicht interessiert. Dies hat einen Einfluss auf die Performance bei der Weganimation. Vorwärtsschauend auf die nächsten Sprints haben wir diese Problematik dem Scrum Team gemeldet.

Des Weiteren mussten wir Mängel bei der Dockableimplementation feststellen: Es gibt keine Möglichkeit die Dockables mit den `WindowSettings` zu initialisieren. Im Konstruktor sind noch keine `WindowSettings` vorhanden, weshalb die Initialisierung in die `update()` Methode verla-

gert werden muss und somit auch bei jedem Update überprüft werden muss, ob das Dockable bereits initialisiert ist.

13.3 User Story 29: Auswahl mehrerer Einzelpersonen

Diese User Story umfasst das Dockable zur Auswahl von Einzelpersonen sowie Gruppen. Die Liste soll hierarchisch dargestellt sein und die Auswahl soll zur Laufzeit verändert werden können. Beim Auswählen der zu visualisierenden Besucher soll nach Start- und Endzeit gefiltert werden können. Gruppenzugehörigkeiten müssen in einer hierarchischen Form klar ersichtlich sein.

13.3.1 Analyse

Die ausgewählten Besucher werden mithilfe einer Mischung aus Tabelle und Baum dargestellt. Dies ermöglicht das Zusammenführen einzelner Besuchern zu Gruppen und zugleich können weitere Informationen angezeigt werden. Beim Hinzufügen eines Benutzers wird eine Farbe generiert, mit der der Weg gezeichnet wird. Diese Farbe kann über einen Mini-Dialog angepasst werden. Mit der Schaltfläche „Hinzufügen“ [1] wird ein Dialog geöffnet, mit dem weitere Besucher hinzugefügt werden können.

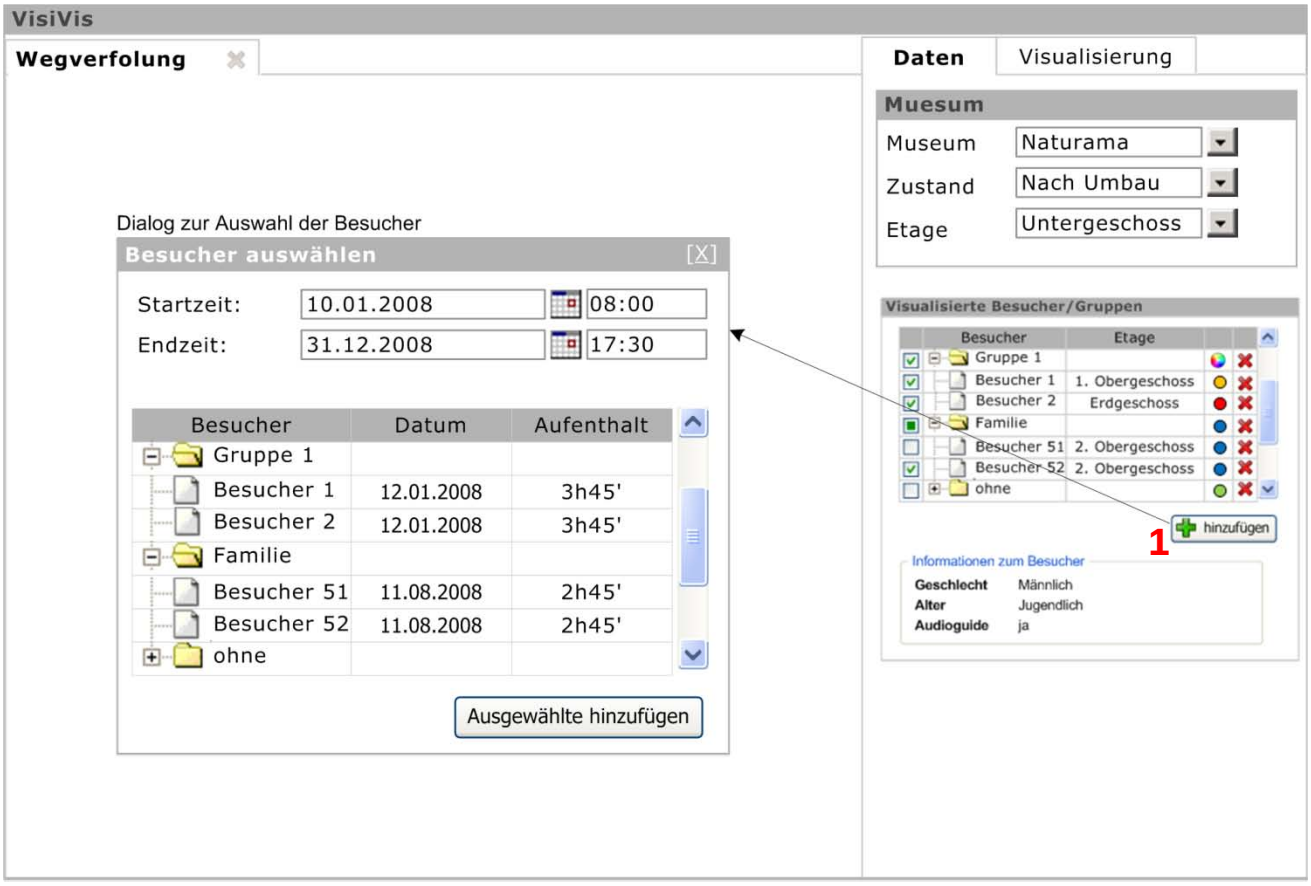

Abbildung 54: Design zur Auswahl von mehreren Besuchern

13.3.2 Fazit

Die Analyse wurde vorausschauend direkt für alle möglichen Arten von Besucherauswahlen getätigt und direkt mit dem Product Owner abgesprochen. So braucht es bezüglich der Besucherauswahl in Zukunft keine Analyse und Besprechung mehr. Mit der Implementierung wurde jedoch noch nicht angefangen. Dies wird im folgenden Sprint umgesetzt (siehe Kapitel *14.2 User Story 29: Auswahl mehrerer Personen / Gruppen*).

13.3.3 Sprint Retrospektive

Positive Aspekte des Sprints

+ Beim Projekt Management stellt sich Routine ein, was für das ganze Team weniger Management Aufwand zur Folge hat.
+ Alle geplanten Tasks konnten erledigt werden (sogar noch mehr).
+ Am ersten Sprint Entwicklungstag waren die Etagen-Transferpunkte hervorragend durch Thomas Kälin vorbereitet worden, so dass wir gleich loslegen konnten ohne Zeit zu verlieren.

Negative Aspekte des Sprints

– Detaillierte Zeiterfassung ist oftmals schwierig und teils zu ungenau, wenn man diese nicht immer nach einem halben Arbeitstag nachführt. Bereits am Abend eines kompletten Arbeitstages, kann man nicht immer zur 100% genau sagen, wie lange man an welchem Task gearbeitet hat.

Verbesserungsmassnahmen für nächsten Sprint

Abgeleitet aus den negativen Punkten des Sprints, wurden folgende Massnahmen zu Verbesserung des nächsten Sprints getroffen:

- Für die Zeiterfassung werden wir die letzten zwei Sprints das Software System Paymo *[URL08]* nutzen. Dieses zeichnet die Arbeitszeit automatisch auf und überträgt sie auf einen Server, wo dann detaillierte Auswertungen gemacht werden können.

14 Sprint 5

14.1 Sprint Tasks

ID	Beschreibung	Soll	Ist
29-4	Auswahl mehrerer Personen / Gruppen: Implementierung TreeTable	14.0	21.3
29-5	Auswahl mehrerer Personen / Gruppen: Implementierung Color Picker Cell	6.0	7.3
29-6	Auswahl mehrerer Personen / Gruppen: Implementierung Auswahl Dialog	4.0	14.0
29-7	Auswahl mehrerer Personen / Gruppen: Tests	3.0	2.5
29-8	Auswahl mehrerer Personen / Gruppen: Dokumentation	5.0	7.3
30-1	Wege mehrerer Personen: animierte Punkte: Analyse & Design	4.0	1.5
30-2	Wege mehrerer Personen: animierte Punkte: Implementierung Darstellung mehrere Personen	8.0	15.5
30-3	Wege mehrerer Personen: animierte Punkte: Implementierung Berechnung unterschiedliche Wege	16.0	0.0
30-4	Wege mehrerer Personen: animierte Punkte: Unittests, Systemtests	4.0	0.0
30-5	Wege mehrerer Personen: animierte Punkte: Dokumentation	8.0	0.0
90-5	Bereinigung Sprint 4	8.0	10.5
		80.0	**79.8**

Tabelle 25: Tasks zum Sprint 5

14.2 User Story 29: Auswahl mehrerer Personen / Gruppen

Im vorherigen Sprint wurde bereits die Analyse dieser User Story gemacht. Die Besucher bzw. Besuchergruppen sollen anhand einer Mischung eines Baumes und einer Tabelle dargestellt werden. Der Schwerpunkt dieser User Story ist die Implementierung dieser `JTreeTable`. Folgende Kapitel umfassen die Implementation und die Tests.

14.2.1 Implementierung TreeTable

Java bietet in ihrer SDK leider keine Implementierung einer „Tree-Table“. Aus diesem Grund musste für diese Anforderung selber eine Komponente programmiert werden, welche die gewünschte Funktionalität liefert. Eine Internetrecherche nach bereits bestehenden Tree-Tables in Java lieferte sehr wenige brauchbare Resultate.

Auf der offiziellen Webseite von Sun schrieb Philip Milne von Sun einen technischen Artikel über eine eigene Implementierung einer `JTreeTable` *[MILNE03]*. Diese Implementierung zeigt jedoch einige Schwächen und ist für einen produktiven Einsatz nicht zu gebrauchen. Scott Violet und Kathy Walrath schrieben darauf einen Folgeartikel *[VIOLET03]* zu P. Milnes Arbeit und verbesserten seine Implementierung sehr stark.

Wir entschieden uns auf der Basis deren Arbeiten aufzusetzen, deren Implementierung zu verwenden und wo nötig anzupassen. Die meisten Klassen konnten komplett übernommen werden, wobei kleine Anpassungen und Verbesserungen nicht zu vermeiden waren. Fol-

gendes Klassendiagramm zeigt die Struktur der `JTreeTable` Implementierung, welche Klassen Eigenimplementierungen sind und welche übernommen und angepasst wurden.

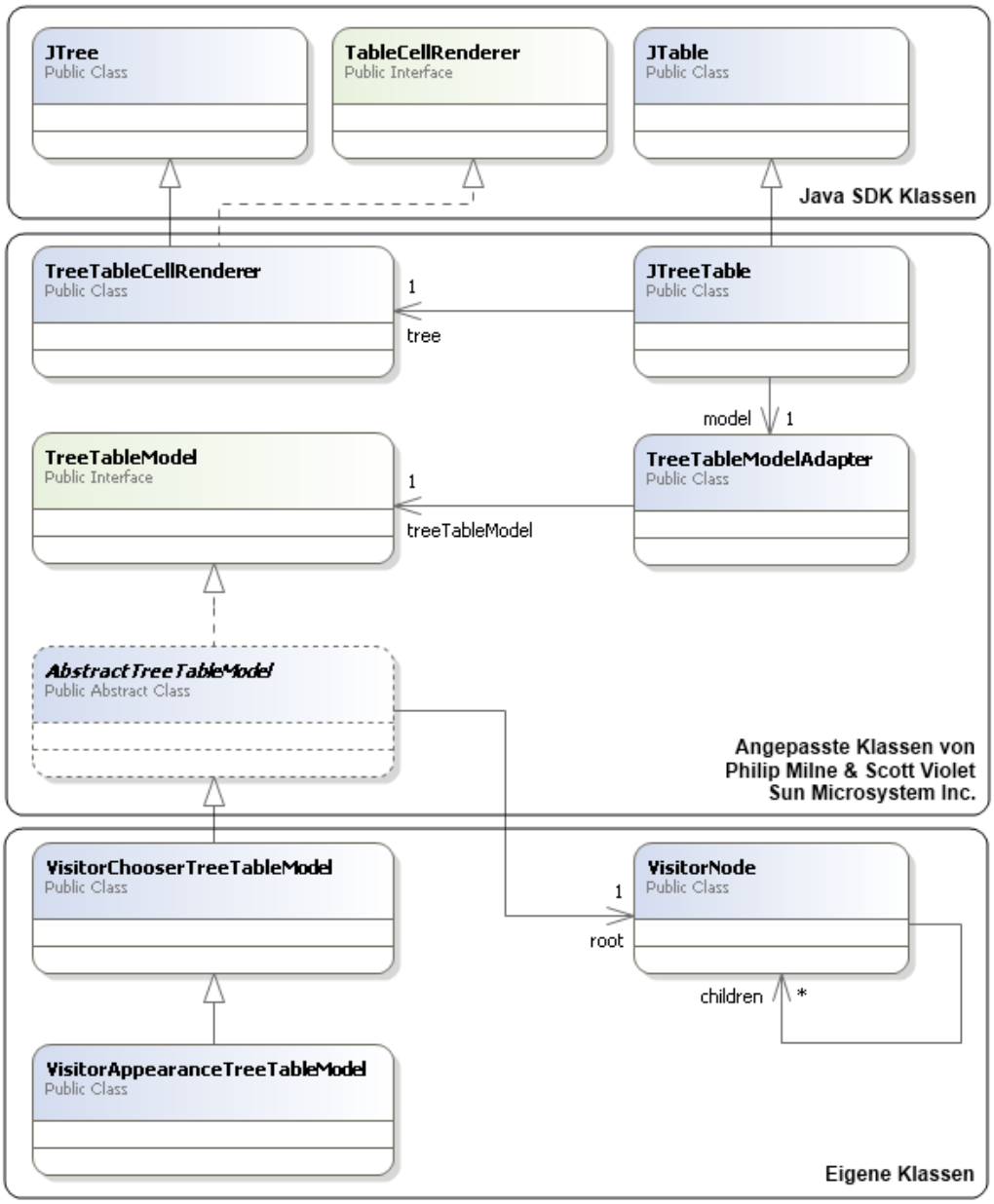

Abbildung 55: Klassendiagramm JTreeTable

Beschreibung der Klassen

Klasse	Beschreibung
JTreeTable	GUI Klasse, welche sich gegen aussen wie eine normale `JTable` verhält, und in der ersten Spalte der Tabelle einen `JTree` darstellt. Die Originalimplementierung dieser Klasse wurde ein wenig erweitert, da es bezüglich der Selektion noch ein paar Fehler beinhaltete. Beispielsweise verlor die Tabelle jegliche Selektion, sobald ein Eintrag hinzugefügt oder gelöscht wurde.
TreeTableCellRenderer	Java `CellRenderer`, welcher in der ersten Spalte einer Tabelle einen `JTree` zeichnet.
TreeTableModelAdapter	*Adapter Klasse [GAMMA95]*, welche das eigentliche `TreeTableModel` kapselt und gegen aussen als Wrapper agiert, indem es das `TableModel` Interface implementiert und die einzelnen Methoden entweder mithilfe des `TreeTableModel` oder direkt mit dem `JTree` implementiert.
TreeTableModel	Interface Definition, die vom Java `TreeModel` erbt und die Schnittstelle für alle Models für die `JTreeTable` vorgibt.
AbstractTreeTableModel	Abstrakte Implementierung des `TreeTableModel` Interfaces, in der die generellen Methoden bereits ausprogrammiert sind, so dass die spezifischen Models nur noch das implementieren müssen, was für das jeweilige Model relevant ist.
VisitorNode	Repräsentiert einen Knoten im Baum der Besucher dar. Hierbei kann es sich um den Wurzelknoten, einen Gruppenknoten oder einen Besucherknoten handeln. Ein `VisitorNode` besitzt selber wieder eine Liste mit all seinen angehängten Unterknoten.
VisitorChooserTreeTable Model	Model Klasse, welche alle Besucher eines bestimmten Museumszustandes enthält.
VisitorAppearanceTree TableModel	Model Klasse, welche alle ausgewählten Besucher enthält, die in der Wegvisualisierung dargestellt werden.

Tabelle 26: Beschreibung der Klassen zum JTreeTable

14.2.2 Implementierung Farbauswahl

Das Resultat der Umsetzung der Farbauswahl pro Besucher / Gruppe ist in Abbildung 56 ersichtlich. Wie in der Analyse im Kapitel *13.3.1* definiert, hat der Benutzer die Möglichkeit jeder Gruppe und jedem Besucher eine individuelle Farbe zuzuweisen. In der folgenden Abbildung ist gut erkennbar, wie sich die Darstellung der Farbe einer Gruppe verhält, wenn mehrere Besucher pro Gruppe ausgewählt sind. Besitzen die Besucher einer Gruppe alle die gleiche Farbe wie in der Abbildung die Familie Kälin, so ist auf der Zeile der Gruppe auch diese Farbe ausgewählt. Haben aber die einzelnen Besucher einer Gruppe eine unterschiedliche Farbe, wie zum Beispiel in der *Abbildung 56* die Familie Zürcher, so erscheint ein Symbol mit gemischten Farben.

Für die Implementierung der Farbauswahl wurde auf den Standard `JColorChooser` von Java zurückgegriffen, da dieser bereits alle nötigen Features unterstützt, die für diesen Fall notwendig sind.

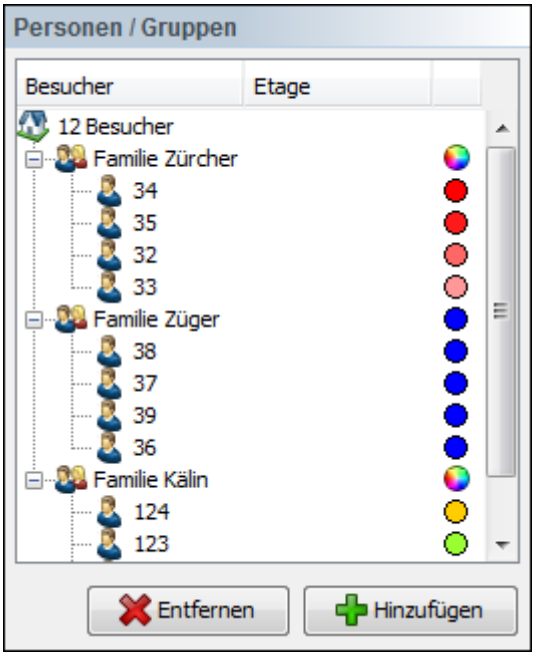

Abbildung 56: Farbauswahl in Besucher Tabelle

14.2.3 Implementierung Besucher Auswahl Dialog

Jede `JTable` unterstützt bereits das Filtern und Sortieren der Tabelle mittels der Klassen `RowFilter` und `RowSorter`, welche intern auf das `TableModel` zugreifen. Die `JTreeTable` besitzt auch ein `TableModel`, welches jedoch für eine Anwendung in der `JTreeTable` so stark modifiziert wurde, dass kein sauberes Sortieren und Filtern mehr möglich ist. Untersuchungen haben ergeben, dass eine Anpassung mehr Zeit beansprucht, als für die komplette User Story zur Verfügung steht. Aus diesem Grund wird nun beim Ändern der Filterkriterien der Baum immer neu aufgebaut. Tests haben ergeben, dass die Performanceeinbusse durch die Neugenerierung der Einträge kaum spürbar ist. Falls dies bei wachsender Anzahl Datensätze doch ein Problem werden sollte, kann ein Limit der Zeitspanne beim Filter eingebaut werden, so dass zum Beispiel maximal eine Zeitspanne von zwei Monaten ausgewählt werden kann.

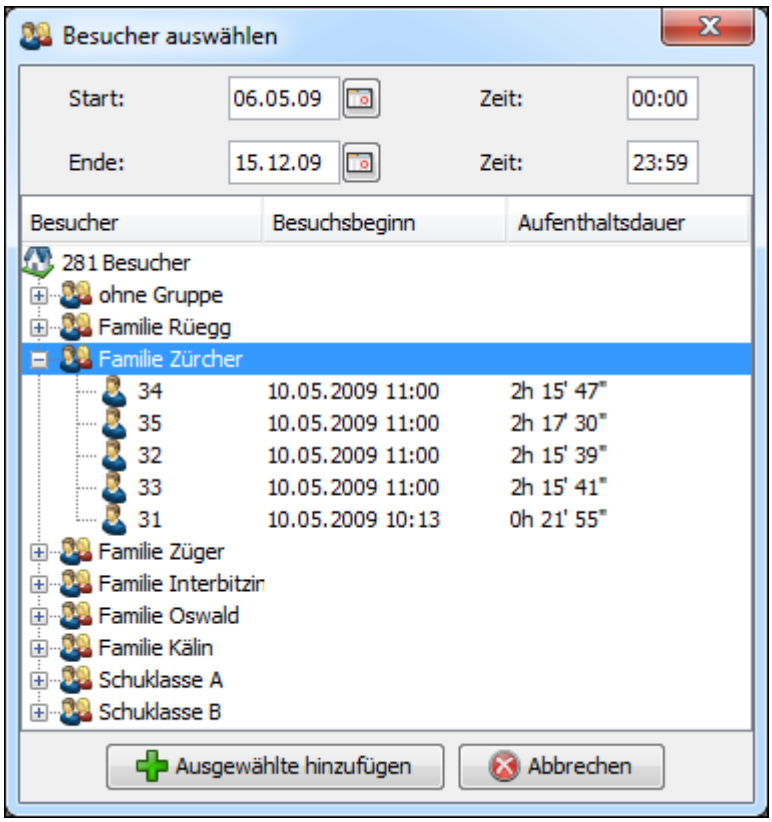

Abbildung 57: Besucherauswahl Dialog

Genauere Informationen zu einem Besucher werden per Tooltip angezeigt:

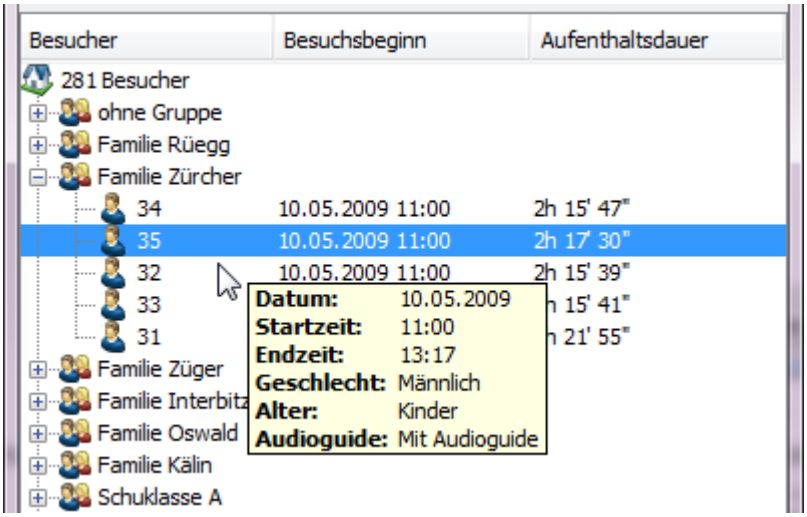

Abbildung 58: Tooltip mit Detailinformationen des Besuchers

14.2.4 Tests

Die Umsetzung der User Story befasste sich ausschliesslich mit UI Elementen, so dass nur ganz wenige Anpassungen an der Problem Domain gemacht und zwei `TreeTableModel` Klassen hinzugefügt wurden. Da die Unittests die UI Funktionen nicht abdecken können, wurde folgendes Testprotokoll erstellt:

Testfall	Erwartetes Resultat	OK?
Einzelner Besucher auswählen und *Hinzufügen* Schaltfläche betätigen.	Die zugehörige Gruppe mit dem ausgewählten Besucher erscheint in der Liste.	✓
Eine Gruppe markieren und *Hinzufügen* Schaltfläche betätigen.	Die Gruppe mit allen zugehörigen Besuchern erscheint in der Liste.	✓
Mehrere Besucher einer Gruppe auswählen (nicht alle), anschliessend *Hinzufügen* Schaltfläche betätigen.	Die Gruppe erscheint in der Liste mit nur den Markierten Besuchern.	✓
Einen bereits hinzugefügten Besucher nochmals hinzufügen.	Es passiert nichts, da der Besucher bereits in der Liste vorhanden ist und seine Wege bereits berechnet sind.	✓
Eine bereits hinzugefügte Gruppe (auch alle Besucher davon sind bereits hinzugefügt) nochmals hinzufügen.	Es passiert nichts, da die Gruppe und alle Ihre Besucher bereits in der Liste sind (Wegberechnung ist bereits schon erfolgt).	✓
Einige der Besucher einer Gruppe sind bereits hinzugefügt (nicht alle). Gruppe auswählen und *Hinzufügen* betätigen.	Die fehlenden Besucher der Gruppe werden hinzugefügt.	✓
Im Dockable einen Besucher auswählen und die *Löschen* Schaltfläche betätigen (oder Delete-Taste).	Der Besucher verschwindet aus der Liste.	✓
Im Dockable eine Gruppe auswählen und die *Löschen* Schaltfläche betätigen (oder Delete-Taste)	Die Gruppe und alle Besucher verschwinden aus der Liste.	✓
Anwählen eines Besuchers im „Komplett Pfad“ Modus.	Weg des Besuchers wird mit der entsprechenden Farbe gezeichnet.	✓
Ändern der Farbe eines Besuchers, welcher alleine in einer Gruppe ist.	Der Weg erscheint in der ausgewählten Farbe. Des Weiteren wird der Farbkreis bei der Gruppe auf die gewählte Farbe gesetzt.	✓
Eine Gruppe mit zwei Besuchern: einer rot, der andere Blau.	Der Farbkreis der Gruppe erscheint mehrfarbig.	✓
Eine Gruppe mit zwei Besuchern: einer rot, der andere Blau. Nun wird der blaue auch auf rot gewechselt.	Der Besucher wird nun auch rot gezeichnet. Der Gruppenfarbkreis wird rot, da alle Gruppenmitglieder dieselbe Farbe haben.	✓
Gruppe mit mehreren Besuchern. Auf der Gruppe eine neue Farbe wählen.	Der Gruppen- und Besucherfarbkreis aller Besucher erscheinen in der neuen Farbe.	✓
Wechseln der Farbe eines Besuchers, der in einer Gruppe ist, wo alle dieselbe Farbe haben.	Der Gruppenfarbkreis wird mehrfarbig.	✓

Hinzufügen einer Gruppe mit mehreren Besuchern.	Die Besucher erscheinen alle im selben Farbton, haben eine unterschiedliche Farbe.	

Tabelle 27: Testprotokoll User Story 29 Auswahl mehrere Personen/Gruppen

14.2.5 Fazit

Diese User Story hat uns klar die Grenzen von Swing aufgezeigt und auch verdeutlicht, dass GUI Entwicklung mit Swing sehr mühsam sein kann. Die Umsetzung der `JTreeTable` war trotz der Vorlagen eine sehr komplexe Angelegenheit. Trotzdem haben wir es geschafft alle gewünschten Funktionalitäten umzusetzen. Einige Einschränkungen hat die eigene Implementierung der JTreeTable dennoch. So kann die JTreeTable nicht mit den Standard `RowSorter` und `RowFilter` Klassen der `JTable` sortiert und gefiltert werden. Diese Funktionalitäten müssten auch wieder selber implementiert werden.

14.3 User Story 30: Wege mehrerer Personen als animierte Punkte

Diese User Story beinhaltet die Animation mehrerer Besucher gleichzeitig. Die einzelnen Besucher sollen nur noch als Punkte gezeichnet werden, so dass auch grössere Gruppen animiert dargestellt werden können. Bei der Darstellung muss speziell darauf geachtet werden, dass zwischen zwei Wegpunkten mehrere Wege berechnet werden und nicht immer derselbe Weg aus dem Cache gelesen wird. Ansonsten hätte man das Problem, dass Personen dergleichen Gruppe stets den exakt gleichen Weg zurücklegen und sich wie in einer Schlangenlinie fortbewegen. Des Weiteren muss der aktuell markierte Besucher hervorgehoben werden. Die Option „Etage automatisch Wechseln“ lässt den derzeitig markierten Besucher über alle Etagen verfolgen.

14.3.1 Analyse und Design

Damit im Sprint 4 die Besucher über mehrere Etagen visualisiert werden konnten, wurde damals die Klasse `FloorStay` eingeführt (siehe Kapitel *13.2.2 Implementierung*). Aufbauend auf diesem Konzept wurde beim Design dieser User Story eine `VisitorStay` Klasse eingeführt. Ein Objekt dieser Klasse kapselt einen kompletten Museumsaufenthalt eines Besuchers mit all seinen einzelnen Wegen über die verschiedenen Etagen. Das bedeutet, dass diese Klasse intern eine Liste mit allen `FloorStay` Objekten führt. Das ermöglicht das Führen einer Liste mit allen ausgewählten Besuchern (d.h. deren `VisitorStay` Objekte), wobei dann beim Zeichnen lediglich über diese Liste iteriert werden muss und somit jeder einzelne Besucher gezeichnet werden kann.

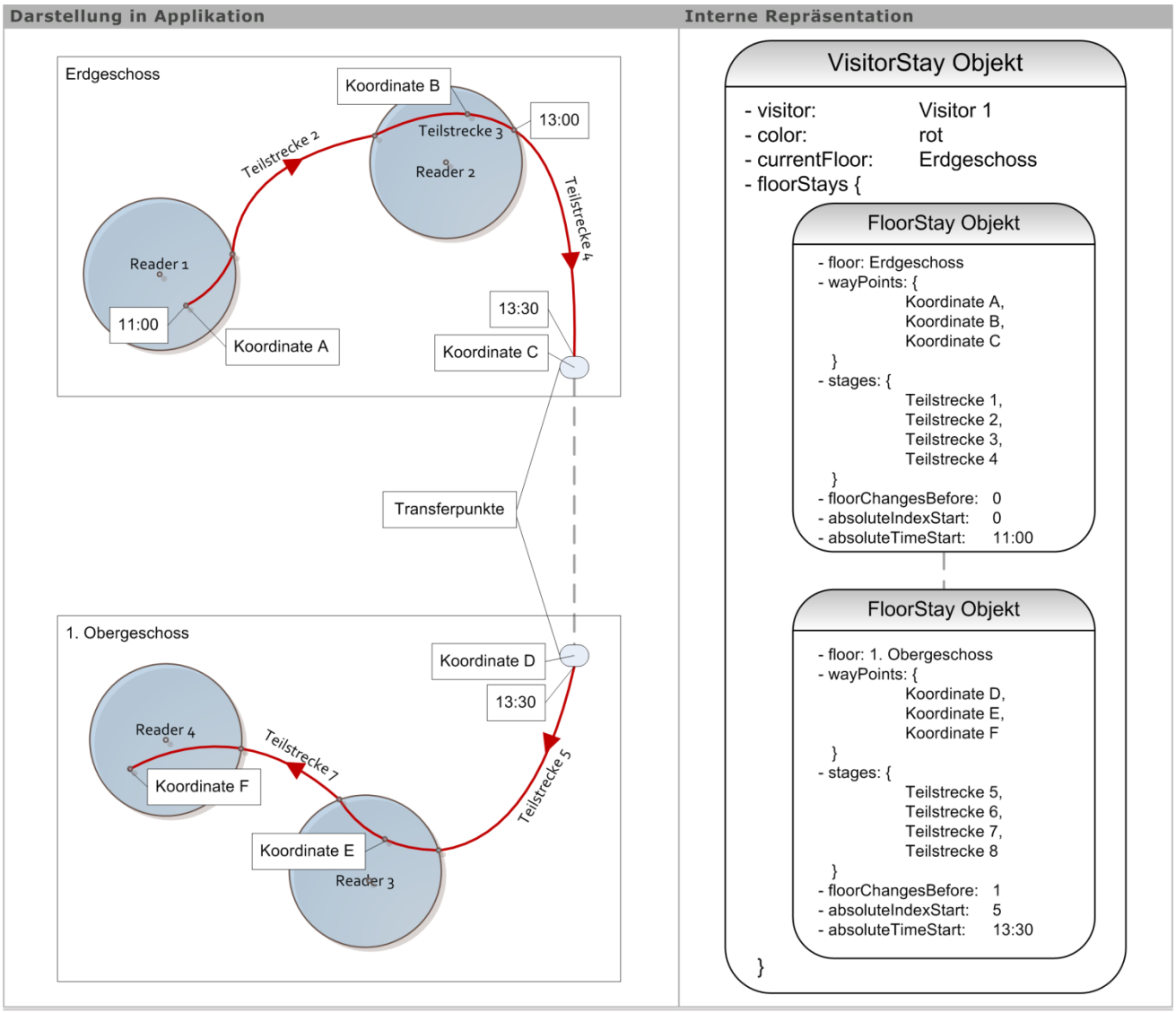

Abbildung 59: Konzept VisitorStay Klasse

14.3.2 Implementierung der Darstellung von mehreren Personen

Im Kapitel *12.2.3 Implementierung: Visualisierung der Wegstrecken* wurde bereits beschrieben, wie die unterschiedlichen Visualisierungsmodi mit Hilfe der `WayPainter` Klasse gezeichnet werden und welche unterschiedlichen `Painter` für die einzelnen Modi verantwortlich sind. Damit nun bei einer Visualisierung von mehreren Personen jeder Besucher als einen einzelnen Punkt im Museum erscheint, wurde ein neuer `Painter` mit dem Namen `AnimateVisitorsPainter` erstellt. Zusätzlich wurde in der abstrakten `Painter` Klasse eine abstrakte Methode `paintOnlySelectedVisitor()` hinzugefügt, die jeder konkrete `Painter` implementieren muss. Das bietet jedem `Painter` die Möglichkeit zu definieren, ob er immer für alle ausgewählten Besucher aufgerufen werden möchte, oder nur für den jeweils selektierten Besucher.

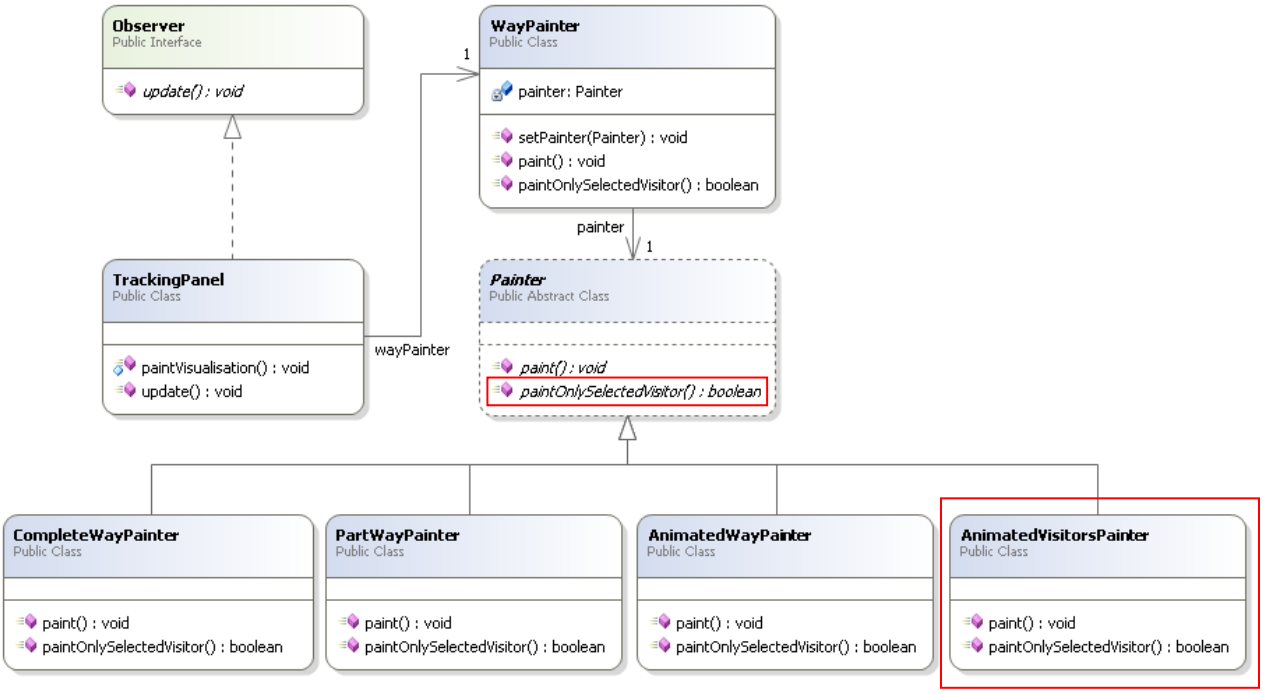

Abbildung 60: Erweitertes Klassendiagramm WayPainter

Die *Abbildung 61* zeigt wie die unterschiedlichen Besucher als farbige Punkte in der Etage dargestellt werden. Dabei ist auch klar zu erkennen, wie einige Besucher auf den exakt gleichen Wegen durchs Museum gehen. Um dieses Problem zu lösen, müssen bei der Wegberechnung mehrere unterschiedliche Wege berechnet werden und im Cache gespeichert werden. Diese Aufgabe wird im nächsten Sprint umgesetzt (siehe Kapitel *15.2.2 Implementierung der Berechnung von unterschiedlichen Wegen*).

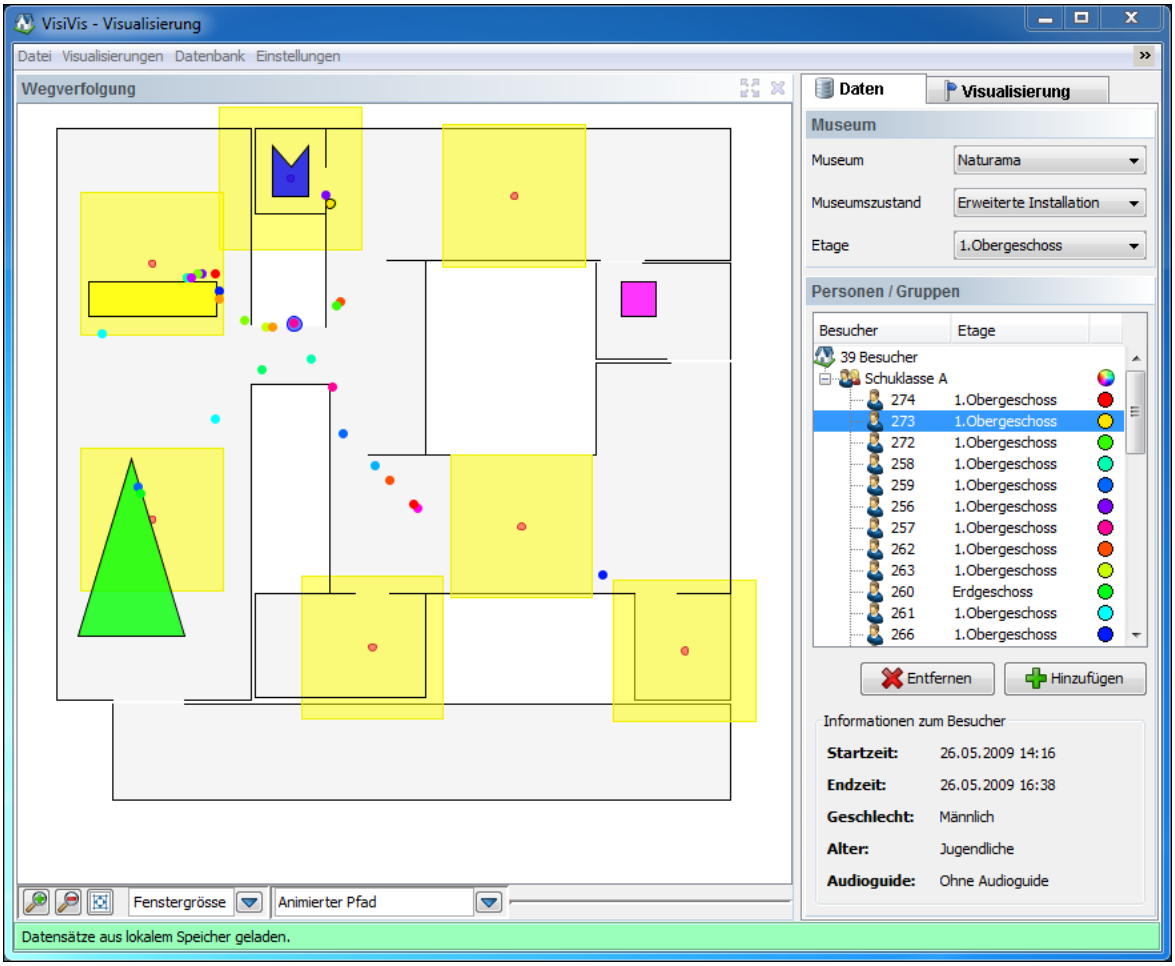

Abbildung 61: Screenshot der Visualisierung von mehreren Besuchern

14.3.3 Fazit

Da diese User Story erst als Next Task geplant und gar nicht zwingender Bestandteil dieses Sprints war, können wir sehr zufrieden mit dem Erreichten sein. Zusätzlich hat sich einmal mehr bestätigt, dass die zuvor im Pair Programming erstellten Programmteile immer noch sehr gut für alle weiteren Tasks zu gebrauchen sind und der ganze Algorithmus und das Zeichnen der Besucher sehr gut skalieren. Alles bereits Erstellte konnte so belassen werden wie es ist und wir mussten lediglich die neuen Teile dem bestehenden Code hinzufügen. Mittels den Refactoring Tools von Eclipse war dies jedoch überhaupt kein Problem.

Wie bereits erwähnt, zeigt das bereits Implementierte sehr gut, wie wichtig das Berechnen von unterschiedlichen Wegen für die Animation von mehreren Personen ist. Dabei sind wir auch sehr optimistisch, dass diese Funktionalität auch wieder sehr gut der bereits bestehenden Lösung hinzugefügt werden kann.

14.4 Sprint Retrospektive

Positive Aspekte des Sprints

+ Alle geplanten Tasks sind komplett abgeschlossen. Der Next Task konnte bereits teilweise umgesetzt werden.
+ Projektmanagement hat sich mittlerweile perfekt eingespielt, weshalb es dazu nichts Negatives anzumerken gibt.
+ Visualisierung mehrerer Personen konnte dank der sauberen Vorarbeit (mit Pair Programming) sehr gut in die bestehende Anwendung eingebaut werden.
+ Zeitabrechnung über Paymo funktioniert sehr gut. Leider können aber keine Soll Zeiten pro Task definiert werden.

Negative Aspekte des Sprints

– Umsetzung der `JTreeTable` war mit sehr grossem Aufwand verbunden, da von Java nichts vorgegeben ist.
– Für die Anpassungen und Fehlerbehebungen am Museums-Auswahl Dockable fühlte sich niemand verantwortlich. Dies hätte aber bereits früher als regulären Task geplant werden können.

15 Sprint 6

15.1 Sprint Tasks

ID	Beschreibung	Soll	Ist
30-6	Wege mehrerer Personen: Erweiterte Analyse und Besprechung mit PO	4.0	2.5
30-7	Wege mehrerer Personen: Implementierung erweiterte Analyse	4.0	6.0
30-3	Wege mehrerer Personen: animierte Punkte: Implementierung Berechnung unterschiedliche Wege	16.0	8.0
30-4	Wege mehrerer Personen: animierte Punkte: Unittests, Systemtests	4.0	4.8
30-5	Wege mehrerer Personen: animierte Punkte: Dokumentation	8.0	8.0
90-6	Bereinigung Sprint 5	8.0	4.8
		44.0	**34.0**

Tabelle 28: Tasks zum Sprint 6

15.2 User Story 30: Wege mehrerer Personen als animierte Punkte

In Sprint 5 konnte bereits ein Grossteil von dieser User Story fertiggestellt werden. Im Sprint 6 geht es nun darum, den Cache zu verbessern, so dass verschiedene Wege zwischen zwei Punkten berechnet werden, damit im Animationsmodus die Punkte nicht hinter oder sogar übereinander sind. Wie viele Wege maximal zwischen zwei Punkten berechnet werden, soll konfigurierbar sein. Des Weiteren werden mit dem Product Owner unsere Ideen bezüglich der automatischen Farbwahl beim hinzufügen von Gruppen bzw. einzelne Personen besprochen und implementiert.

15.2.1 Erweiterte Analyse

Das Berechnen von unterschiedlichen Wegen für die gleichzeitige Darstellung von mehreren Personen ist ein sehr wichtiger Teil und macht diese Visualisierung erst brauchbar. Konkret heisst dies, dass zwischen zwei RFID Empfangsbereichen immer mehrere verschiedene Wege berechnet werden müssen. Dies gilt für den Hin- wie auch für den Rückweg. Nach einer Besprechung mit dem Product Owner hat man sich darauf geeinigt für alle statischen Visualisierungen immer den ersten berechneten Weg darzustellen, da hierbei sowieso nur immer eine Person gleichzeitig dargestellt wird. Das bedeutet allerdings, dass eine Person (im Code ein `VisitorStay`) immer zwei Wege besitzt: Ein Weg für die statische Visualisierungen (immer der erst berechnete Weg) und ein Weg für die animierte Darstellung. Das hat zwar zur Folge, dass mehr Speicher gebraucht wird, dafür wird für den Besucher im Einzelmodus immer der optimalste Weg dargestellt. Die *Abbildung 62* zeigt in welchen Bereichen unterschiedliche Wege berechnet werden und wie alle Wege immer wieder beim Wegpunkt eines RFID Empfangsbereiches zusammenführen.

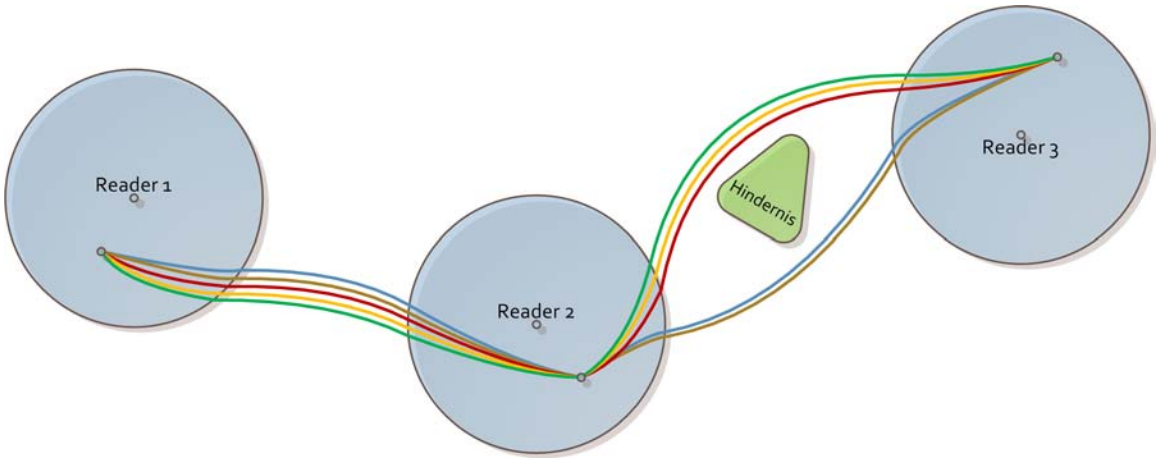

Abbildung 62: Berechnung von mehreren Wegen

15.2.2 Implementierung der Berechnung von unterschiedlichen Wegen

Damit diese Anforderung in die bestehende Anwendung integriert werden konnte, musste die Klasse `WayPointCache` so angepasst werden, dass jeder Cache Zugriff, lesend wie auch schreibend, immer mit einem Index geschehen muss. Somit kann die Klasse `WayAlgorithm` selber steuern, welchen Weg sie nun lesen oder schreiben möchte. Wie viele unterschiedliche Wege berechnet werden, ist in der Datei `visual.properties` unter dem Schlüssel `tracking.calculation.maxways` konfiguriert. Tests haben gezeigt, dass im Naturama Museum bereits fünf unterschiedliche Wege genügen. Dieser Wert hängt jedoch sehr von der Gebäudestruktur des Museums ab. So macht es beispielsweise bei einem Museum mit sehr vielen engen Gängen keinen Sinn zehn verschiedene Wege zu berechnen, da bei engen Gängen gar nie zehn Personen nebeneinander gehen können.

Damit die Visualisierung eines bestimmten Besuchers direkt und ohne Unterbrüche gestartet werden kann, wird beim Hinzufügen eines Besuchers der komplette Weg über alle Etagen gemäss dem *Eager Loading Pattern* *[POSA3]* zum Voraus berechnet. Die Anforderung der unterschiedlichen Wege wird jedoch gemäss dem *Lazy Loading Pattern* *[POSA3]* umgesetzt. Das heisst, aus Performance Gründen werden beim ersten Besucher nicht direkt alle fünf unterschiedlichen Wege berechnet, sondern nur der erste. Beim zweiten Besucher kommt dann ein zusätzlicher Weg in den Cache usw. Damit der `WayAlgorithm` unterschiedliche Wege berechnen kann, muss dieser bei jedem neuen Weg die bereits berechneten Wege aus dem Cache laden und diese in seiner internen Museumsdarstellung als zurückgelegte Wege markieren, damit diese nicht nochmals durchquert werden.

Sobald der Cache mit fünf unterschiedlichen Wegen gefüllt ist, werden gar keine Berechnungen mehr ausgeführt. Danach liefert der Cache bei jedem Zugriff einen anderen Weg, wobei immer über die vorhandenen Wege iteriert und stets der nächste Weg zurückgegeben wird. Nach fünf Zugriffen auf dieselbe Teilstrecke liefert der Cache wieder den ersten Weg.

15.2.3 Systemtests

Um die neue Wegberechnung zu testen, wird der `CompleteWayPainter` so abgeändert, dass dieser die Wege für die Animation zeichnet (Änderungen in rot):

```
// Zeichne alle Teilstrecken nacheinander
for(int index = 0; index < fs.getAnimatedStages().size(); ++index) {
  GeneralPath path = convertToPath(fs.getAnimatedStages().get(index), color, false, true);
  paintWay(path, color, false);
}
```

Des Weiteren muss `paintOnlySelectedVisitor()` angepasst werden, damit nicht nur der Selektierte Besucher gezeichnet wird:

```
@Override
public boolean paintOnlySelectedVisitor() {
  return false;
}
```

Nun werden alle Wege gezeichnet, so dass überprüft werden kann, ob unterschiedliche Wege berechnet werden. Wenn nun unterschiedliche Besucher hinzugefügt werden, sind die Wege nur an sehr engen Stellen wo es gar nicht anders möglich ist übereinander, wie in den untenstehenden Grafiken ersichtlich ist:

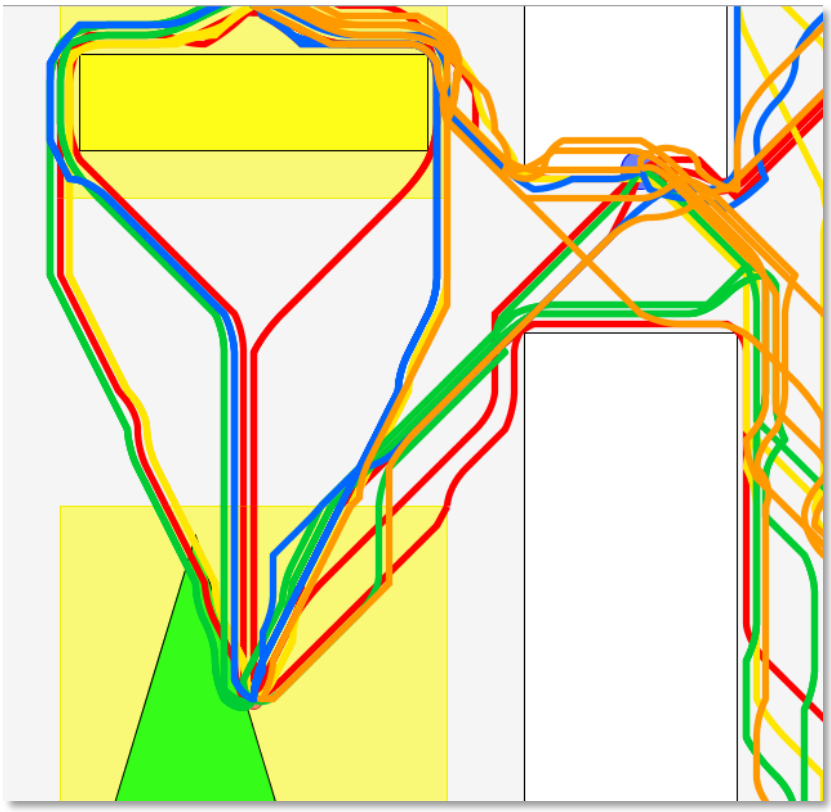

Abbildung 63: Verschiedene Wege

Für diese Grafik wurden 5 verschieden Besucher hinzugefügt.

Folgende Testfälle wurden User Story erstellt:

Testfall	Erwartetes Resultat	OK?
Zwei Besucher hinzufügen, bei denen beide einige Teilstrecken (von Reader zu Reader) dieselbe Start und Endpunkte haben. Wege im „Kompletter Pfad“-Modus analysieren.	Die Teilstrecken von den Readers müssen identisch sein. D.h. der Weg von Reader A zu Reader B muss bei beiden Besuchern identisch sein.	✓
Selber Testfall wie oben.	Wege von Reader A zu einem Reader C gehen vermeiden den Weg von Reader A zu, so dass diese sich nicht überlappen. Es kann stellen geben, bei denen dies unmöglich ist wie zum Beispiel Enge Durchgänge oder die Start bzw. Endpunkte.	✓
Änderungen am `CompleteWayPainter` gemäss vornehmen. Anschliessend mindestens fünf Besucher hinzufügen und die Wege mit dem „Kompletter Pfad“-Modus analysieren.	Wie in *Abbildung 64* ersichtlich, gibt es von einem Reader A zu einem Reader B maximal 5 (siehe Konfiguration `visual.properties`). Es ist klar ersichtlich, dass unterschiedliche Wege angezeigt werden.	✓

Tabelle 29: Testfälle User Story 30

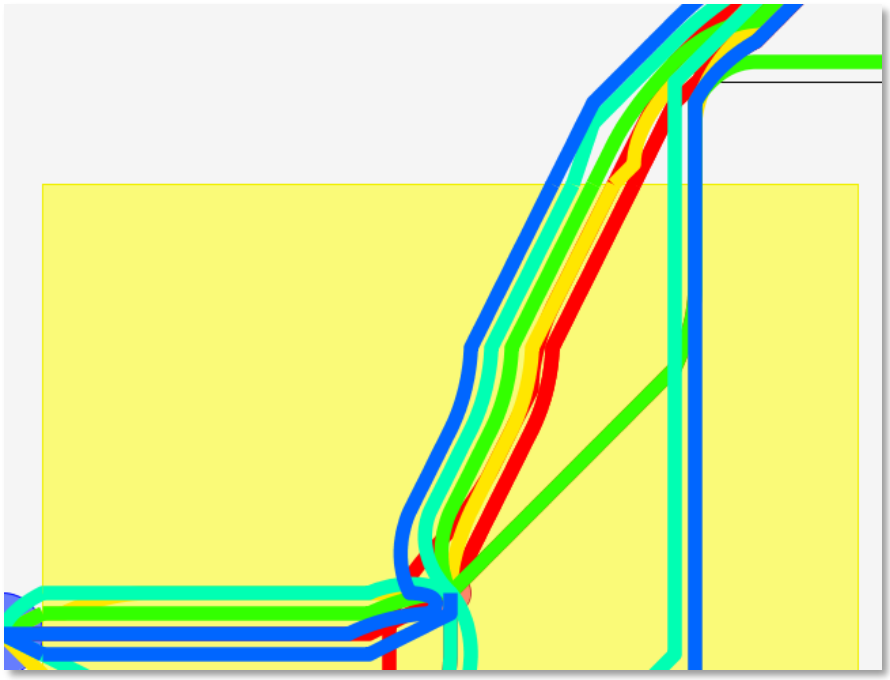

Abbildung 64: Cache Limite auf 5 Wege pro Start/Ziel Kombination

15.2.4 Fazit

Der Zeitaufwand für die reine Implementierung dieses Tasks war letztlich einiges geringer als anfangs angenommen. Dafür musste hierbei mit einem erhöhten Testaufwand gerechnet werden, da es teils sehr schwer nachzuvollziehen war, ob nun die Wege korrekt waren. Die im Kapitel *15.2.3 Systemtests* beschriebenen temporären Anpassungen am `CompleteWayPainter` waren dafür eine sehr grosse Hilfe für die Durchführung der Systemtests.

15.3 Sprint Retrospektive

Positive Aspekte des Sprints

+ Alle offenen Tickets erledigt, TODOs im Code abgearbeitet und Unittests erstellt, so dass ein sauberer Code abgegeben werden kann.
+ Der letzte grosse offene Punkt der ganzen Studienarbeit konnte durch die Berechnung unterschiedlicher Wege komplett erledigt werden.

Negative Aspekte des Sprints

– Das geforderte Poster musste nicht selbsterklärend, obwohl es nicht präsentiert wird. Somit ergibt das Poster keinen grossen Sinn, ausser dass man es bereits einmal für die Bachelorabeit geübt hat.
– Sehr grosser Dokumentationsaufwand zum Schluss der Studienarbeit.

Teil 5: Schlussfolgerungen

16 Beurteilung der Resultate

16.1 Was wurde erreicht?

Folgende Punkte wurden während der Studienarbeit zu 100% erfüllt:

- Alle bestehenden Weg-Visualisierungen wurden neu implementiert.
- Die Wegberechnung wurde komplett neu implementiert, wobei sie schneller arbeitet und zusätzlich skalierbarer in Bezug auf die Museumsgrösse ist.
- Die Anforderung mehrere Personen und Gruppen gleichzeitig in einem Fenster zu visualisieren, wurde komplett erfüllt.
- Wegverfolgungen über beliebig viele Etagen werden von allen Weg-Visualisierungen unterstützt.
- Die neue Wegberechnung generiert rundere und daher realistischere Wege.

16.2 Offene Arbeiten

Anbei werden diejenigen Arbeiten aufgelistet, die im Rahmen dieser Studienarbeiten noch gar nicht oder nicht zu 100% umgesetzt wurden. Dabei handelt es sich auch teilweise um reine Ideen, die noch nicht im Detail analysiert wurden.

16.2.1 Wegverfolgung in mehreren Fenstern

Im Sprint 4 wurde bei der Analyse (siehe Kapitel *13.2.1 Analyse*) eine Variante geplant, welche die gleichzeitige Wegverfolgung von mehreren Personen in mehreren Fenstern unterstützen soll. Aus Zeitmangel konnte dabei diese Variante nicht umgesetzt werden. Es ist jedoch möglich mehrere Personen in einem Fenster zu visualisieren. Dabei kann eine Option gesetzt werden, so dass die Etage automatisch wechselt, wenn der selektierte Besucher in eine andere Etage geht.

Für diese Anforderung wurde eine separate User Story 32 geplant.

16.2.2 Wege mehrerer Personen als Punkte mit Spur

Die Visualisierung mehrere Personen als normale Punkte wurde komplett umgesetzt. Eine weitere gewünschte Variante wäre hier die Darstellung der Einzelpersonen als Punkte mit einer leicht transparenten kurzen Spur. Die Spur soll je nach Möglichkeit je nach Geschwindigkeit eine unterschiedliche Länge haben. Der Product Owner erhofft sich dadurch dem Benutzer ein besseres Verständnis über die Geschwindigkeit und zurückgelegten Wege der einzelnen Besucher zu geben.

Für diese Anforderung wurde eine separate User Story 31 geplant.

16.2.3 Verschönerung der Wegberechnung

Die neu berechneten Wege erscheinen bereits abgerundet und wirken daher bereits realistischer als beim Prototyp. Trotzdem gibt es hier noch ein Verbesserungspotential in Bezug auf die Nähe zu den Wänden. Da der A*-Algorithmus stets den kürzesten Weg berechnet, kommt es oft vor, dass die Wege sehr nahe an Wänden entlangführen. Realistischer wäre jedoch, dass die Besucher in den meisten Fällen einen grösseren Abstand zur Wand haben.

Zusätzlich gibt es ein Optimierungspotential bei der Berechnung von mehreren Wegen, so dass die Abstände zwischen den einzelnen Wegen grösser werden. Dazu wurde in der Klasse `WayAlgorithm` bereits ein erster Test gemacht, der auf den ersten Blick sehr vielversprechend aussah. Leider bemerkten wir bei intensiveren Tests, dass es dann teilweise vorkommen kann,

dass einzelne Wege nicht berechnet werden können, da die Abstände zwischen den einzelnen Wege zu gross sind. Aus diesem Grund mussten wir die Anpassungen in der `WayAlgorithm` Klasse wieder auskommentieren.

Für diese Optimierung wurde eine separate User Story 57 geplant.

16.2.4 Darstellung von Besucherströmen

Bereits zu Beginn der Studienarbeit war die Vision einer völlig neuen Visualisierung vorhanden. Dabei sollen Besucherströme in den Etagen ersichtlich sein, indem angezeigt wird, wie viele Personen pro Zeit durch einen Gang oder Tür gehen, was dann mittels dementsprechend dickeren oder dünneren Linien gekennzeichnet wird.

Für diese neue Visualisierung wurde eine separate User Story 34 geplant.

16.2.5 Refactoring der WindowSettings

Im Fazit von User Story 30 des Sprints 4 (siehe Kapitel *13.2.4 Fazit*) wurde erwähnt, dass es in der Handhabung der `WindowSettings` Klasse noch einige Unschönheiten gibt. Da diese Klasse von allen Visualisierungen gebraucht wird, sind hier die Prinzipien *High Cohesion [URL12]* und *Separation of Concerns [URL09]* überhaupt nicht mehr gegeben. Zusätzlich werden dadurch bei einer Änderung eines Attributs dieser Klasse immer gleich alle `Observer` dieser Klasse benachrichtigt, obwohl die meisten gar kein Interesse an der Änderung dieses Attributs haben. Diese Problematik müsste in Zukunft bestimmt noch detaillierter analysiert und verbessert werden.

17 Ausblick

Für die Zukunft des Museumsprojekts sind bereits zwei weitere Arbeiten geplant, die an den in diesem Semester erstellten Resultaten weiterarbeiten.

In einer folgenden Bachelorarbeit im Frühjahrsemester 2010 wird nochmals ein intensiver Feldtest im Naturama Museum in Aarau durchgeführt und die neu entwickelte Software damit getestet. Zusätzlich werden noch einige neue Features hinzukommen und evtl. sogar ein paar der in Kapitel *16.2 Offene Arbeiten* erwähnten Punkte umgesetzt.

Die Firma Albis Technologies aus Zürich, welche die RFID Hardware dieses Systems herstellt hat inzwischen an ihrem System weiterentwickelt, so dass neu punktgenaue Lokalisierungen von Gegenständen bzw. Personen mittels RFID Technologie möglich wird. Mit dieser Grundlage wird im Frühjahrsemester 2010 eine weitere Studien- oder Bachelorarbeit ausgeschrieben, die diese neue Technologie genauer analysiert und testet wobei evtl. einige Teile der vorliegenden Studienarbeit wiederverwendet werden können.

Teil 6: Persönliche Berichte

18 Zwischenberichte

18.1 Zwischenbericht Stefan Züger

Mittlerweile sind bereits drei von sechs geplanten Sprints vorbei und es ist sehr vieles geschehen in den letzten sechs Wochen. Zur Halbzeit des Projektes möchte ich kurz über die vergangenen Wochen reflektieren.

18.1.1 Projekt Management

Als ich anfangs in der Aufgabenstellung gelesen habe, das komplette Projekt solle mit den Softwareentwicklungsmethoden Scrum und XP umgesetzt werden, war ich sehr motiviert als Alternative zu RUP eine neue Entwicklungsmethode kennenzulernen. Ich wusste jedoch nicht, ob es eine gute Idee ist, gerade bei einer Semesterarbeit etwas einzusetzen, was für Tobias und mich komplett fremd ist. Der Begriff Scrum habe ich zwar schon gehört, Details dazu waren mir aber überhaupt keine bekannt. Gewisse Zweifel an der Machbarkeit waren also schon ein wenig vorhanden. Unser Projektleiter (Scrum Master) Thomas Kälin hat aber diesbezüglich sehr gut gehandelt und gleich beim Projekt Kickoff dem ganzen Team eine sehr gute Kurzeinführung in die beiden Methoden gegeben. So waren bereits nach dem ersten Meeting praktisch alle Zweifel wie verflogen. Natürlich musste sich der ganze Prozess zuerst noch bewähren, doch es schien mir nach den Ausführungen von Thomas alles sehr plausibel und sinnvoll.

Das gesamte Projekt war von Anfang an schon so gut organisiert, dass bereits der erste Sprint ohne grössere Hindernisse bewältigt werden konnte. Man erkannte zwar ein paar verbesserungswürdige Punkte, diese wurden dann aber gerade am nächsten Sprint Planning Meeting angesprochen und Verbesserungsmassnahmen vorgeschlagen. So dauerte beispielsweise das erste Meeting vier Stunden, das nächste Meeting konnte dann aber durch Verbesserungsvorschläge auf die Hälfte gekürzt werden. Allgemein empfinde ich es als sehr positiv, dass bei jedem Sprint auch die negativen Aspekte angesprochen werden und die meisten Punkte gleich beim nächsten Sprint verbessert wurden.

Ganz allgemein gesehen bin ich bis jetzt mit der Projekt Organisation durch Thomas Kälin sehr positiv überrascht. Auch das Konzept der täglichen Standup Meetings hat sich sehr gut bewährt. Dazu war ich anfangs eher ein wenig skeptisch, da ich es für viel zu ineffizient empfand, sich täglich zu einer Besprechung zu treffen. Da es bis jetzt aber wirklich immer so geklappt hat, dass die Meetings nie länger als 15 Minuten dauerten und sich alles nur immer aufs Wesentliche beschränkte, finde ich es eine gute Art den Entwicklungsstand der einzelnen Teilnehmer auszutauschen.

Obwohl ich zu Scrum mehrheitlich wirklich nur Positives zu berichten habe, gibt es doch auch Punkte, die an Scrum bis jetzt nicht so optimal waren. Ein Beispiel sehe ich in der Aufteilung der User Stories. Als man zu Beginn alle User Stories für das Product Backlog zusammenstellte, konnte man sich noch nicht ein sehr klares Bild verschaffen, wie die eine oder andere User Story evtl. sehr stark mit einer anderen zusammenhängt. So haben wir für die Auswahl der zu visualisierenden Besucher vier separate User Stories, bei denen aber die Ziellösung für alle gleich aussehen wird. Man macht sich dadurch mehr Aufwand, als es nötig wäre, da man für jede User Story eine separate Analyse durchführt und diese dann separat implementiert. Theoretisch würde es reichen, wenn man eine einzige Analyse durchführen würde, wo man definiert, wie die Ziellösung aussehen soll. Dann könnte man die Implementierung in zwei User Stories aufteilen, in der man in einem ersten Schritt die Basis Funktionalitäten umsetzt und danach in der zweiten User Story noch alle restlichen Funktionen. Einen weiteren negativen Punkt sehe ich in Überschaubarkeit des gesamten Projektfortschrittes. Das Scrum Team plant

zwar immer schön die Tätigkeiten für einen einzelnen Sprint und nimmt sich gerade immer genau so viel vor, wie es in zwei Wochen umsetzen kann. Nichts desto trotz interessiert es den Product Owner auch, wann das gesamte Projekt zu einem Abschluss findet. Oder dass zumindest Meilensteine definiert sind, die sagen, bis wann welcher Teil der Applikation fertig sein muss. Die Burndown Charts für die einzelnen Sprints finde ich eine sehr gute Übersicht über den Projektfortschritt innerhalb eines Sprints. Doch meiner Meinung bräuchte es für den Scrum Master, den Product Owner und das ganze Scrum Team auch so etwas Ähnliches, welches den Fortschritt über das ganze Projekt hinweg verfolgt.

18.1.2 XP-Praktiken

Neben Scrum ist auch das Extreme Programming (XP) völlig neu für mich. Da XP sehr viele Praktiken und Entwicklungsmethoden beschreibt, empfand ich es als sehr gut, dass Thomas Kälin ein eingeschränktes Set an Praktiken definierte, an die wir uns halten möchten. Speziell hervorheben möchte ich aber bis jetzt nur das Pair Programming. Obwohl ich das in früheren Projekten schon öfters unbewusst praktizierte, sah ich darin anfangs nur einen erhöhten Aufwand und eine ineffizientere Arbeitsweise. Doch gerade bei sehr komplexen Programmieraufgaben hat es sich als sehr effizient herausgestellt. Ein wunderbares Beispiel für den Einsatz des Pair Programming lieferte uns die Implementierung des A*-Algorithmus. Diesen haben wir komplett zu zweit an einem Rechner umgesetzt. Anfangs hat man zwar das Gefühl, es sei viel zu ineffizient, doch gerade an diesem Beispiel haben wir bemerkt, dass es sich gelohnt hat. Wir hatten gleich von Anfang an eine Lösung zu Stand gebracht, die sauber und sehr performant ist. Die Qualität dieses Codes hat sich auch später wieder bewiesen, als wir neben dem ganzen Weg noch Teilstrecken und animierte Wege darstellen wollten. Es benötigte praktisch keine Anpassungen am A*-Algorithmus. Obwohl man anfangs vielleicht länger braucht, ist man über das gesamte gesehen sicher viel effizienter, weil man im Nachhinein nicht mehr viel verbessern oder anpassen muss.

Natürlich würde ich das Pair Programming nicht für jede Entwicklungstätigkeit anwenden, da es für triviale Implementierungen tatsächlich einen viel zu grossen Overhead ist. Da ist man effizienter wenn man einfache Aufgaben auf Einzelpersonen verteilt.

18.1.3 Projekt Setup

Zum gesamten Projekt Setup gibt es tatsächlich nichts Grosses zu berichten, da alles sehr mustergültig aufgesetzt ist. Die Projekt Integration und der Build Prozess über den Hudson Server sehe ich als eine sehr grosse Hilfe an. Besonders die vielen zusätzlichen Features des Hudson sind sehr hilfreich. FindBugs, Checkstyle und Cobertura sind alles Plug-Ins, die ich in einem zukünftigen Projekt sofort wieder einsetzen würde. Auch das Trac ist etwas, das im Vergleich zu anderen Lösungen in vergangenen Projekten klar am besten abschneidet. Im SE2 Projekt haben wir dieselben Features über Microsoft Groove verwendet, doch finde ich Groove längst nicht so nützlich wie das Trac.

Zum Projekt Setup kann ich wirklich nichts bemängeln. Dieses würde ich in einem weiteren Java Projekt sofort wieder so einsetzen.

18.2 Zwischenbericht Tobias Zürcher

18.2.1 Projektstart

Im Software Engineering Projekt wurde uns RUP als Dokumentations- und Managementmethode in einer Art aufgezwungen, welche mich persönlich überhaupt nicht überzeugen konnte. Viele redundante Dokumentation musste geschrieben werden, der nutzen Stand nicht im Vordergrund, sondern es musste „einfach da“ sein.

Projektmanagement ist für mich ein etwas leidiges Thema, dies ist darauf zurückzuführen, dass ich bis zum jetzigen Zeitpunkt noch keine zufriedenstellende Methodik gefunden habe.

Aus diesem Grund kam RUP für die Studienarbeit nicht in Frage, sondern ich wollte etwas Neues ausprobieren.

Thomas Kälin hat bereits im Voraus informiert, dass wir das Projekt mit Scrum realisieren werden. Aus Neugier habe ich im Voraus kurz nachgeschaut, was Scrum ist: „Es fokusiert auf das Resultat und versucht dabei unnötigen Management Ballast zu reduzieren" wäre Scrum in einem Satz zusammengefasst. Dies hörte sich für mich schon sehr vielversprechend an und freute mich entsprechend auf das Kickoff Meeting, bei dem Thomas Kälin eine Einführung ins Projekt gab.

Hier muss bereits ein grosses Lob an unseren Scrum Master ausgesprochen werden: Das Projekt war von Anfang an hervorragend organisiert. Obwohl es für alle einige Unklarheiten gab, hatte das Projekt einen Rahmen, an dem man sich festhalten konnte.

18.2.2 Scrum

Da kein Projektmitglied Erfahrung mit Scrum hatte, fielen die ersten Meetings etwas langatmiger aus. In den Scrum Retrospektiven konnte jedoch jeder einzelne positive sowie negative Aspekte einbringen, welche auch direkt verbessert wurden. Mittlerweile ist etwas Routine eingekehrt, so dass auch die Meetings sehr produktiv ablaufen. Viele kleine Konzepte von Scrum wie zum Beispiel das Daily Stand-Up Meeting bewährte sich sehr: man ist sehr gut über den Stand der anderen Arbeiten informiert, ohne gross Zeit zu verlieren. Des Weiteren konnten immer wieder an den Meetings Kleinigkeiten zusammen geklärt werden und es vermittelt, obwohl jeder an seinen User Storys alleine arbeitet, ein gutes Teamgefühl.

Besonders gefällt mir den Fokus auf das Resultat: bei jedem Sprintende wird lauffähige Software präsentiert. User Stories werden im Voraus vom Product Owner definiert und priorisiert. Die Entwickler konzentrieren sich auf das, was sie am besten können: Software entwickeln. Beim Erstellen des Sprint Backlog besteht die Möglichkeit, den Product Owner Fragen zu den einzelnen User Storys zu stellen. Durch das regelmässige Feedback vom Product Owner entwickeln die Programmierer nicht an den Anforderungen vorbei. Gemachte Erfahrungen in den Sprints werden direkt auf die neuen User Storys angewendet, indem zum Beispiel Umpriorisierungen gemacht werden. Scrum ist mit diesen Eigenschaften höchst agil.

Scrum gefällt mir ausserordentlich gut, verfügt aber auch über kleinere Schwachstellen. Zum Beispiel ist es sehr schwierig die Anforderungen in User Storys umzuformen, so dass Sie aus Entwicklersicht wirklich getrennt abgearbeitet werden können. Ein Beispiel waren die User Storys „Weg einer Person anzeigen" und „Weg einer Person in Teilstrecken anzeigen". Sofern man massives Refactoring verhindern will, sollte man bei der Implementierung von „Weg einer Person anzeigen" die zweite User Story mindestens im Hinterkopf haben.

Scrum gibt allen einen guten Überblick über den Status eines Sprints, jedoch fehlt eine Statistik über das gesamte Projekt. Bei Scrum wird grundsätzlich angenommen, dass der Entwicklungsprozess so komplex ist, dass sich dieser im Voraus weder in abgeschlossenen Phasen noch in einzelne Arbeitsschritte planen lässt. So haben wir momentan keine Ahnung, ob wir bis Ende Sprint 6 alles abschliessen können.

18.2.3 XP-Praktiken

Das Planning Gaming ist eine gute Idee, jedoch darf damit nicht übertrieben werden: Auch wenn mehrere Entwickler Aufgaben schätzen, es bleibt eine Schätzung! Darum sollte das Planning Game etwas zügig gespielt werden, so dass man sich nicht in Details verliert.

Einige angewendete Praktiken halte ich ein wenig für „selbstverständlich" und werden von richtig gebildeten Entwicklern implizit getätigt. Ein Beispiel ist das „Refactoring".

Herausheben möchte ich speziell Pair Programming. Ich habe dies in früheren Projekten bereits praktiziert, ohne zu wissen, dass dies eine XP-Praktik ist. Das Konzept unserer Wegbe-

rechnungsarchitektur haben wir zusammen aufgezeichnet und anschliessend im Pair Programming implementiert. Dabei kam eine qualitativ sehr hochwertige Lösung für unsere Problematik zustande, welche sich bis jetzt bewiesen hat. Zu zweit geschriebener Code enthält viel weniger Bugs. Gerade endlose Debugging-Sessions sind sehr motivationsraubend und führen zu einer geringeren Produktivität. Zu zweit ist ein Problem viel schneller eingegrenzt und somit auch schneller behebbar. Grössere Refactoringaktionen an dieser Architektur für die neuen Anforderungen in späteren User Storys werden wir bestimmt wieder im Pair Programming bewältigen.

Pair Programming hat in meinen Augen nur einen Haken: die beiden Entwickler müssen mehr oder weniger auf demselben Niveau sein, so dass niemand über- oder unterfordert ist. Im Pair Programming lernt man auch Tricks und Vorgehensweisen von anderen Entwicklern kennen, dies ist ein schöner Nebeneffekt.

18.2.4 Projekt Setup

Das Projektsetup ist mit Hudson, Checkstyle, Findbugs, Trac etc. vorbildlich aufgesetzt. Bei einem weiteren Java Projekt würde ich alle diese Tools wieder verwenden. Da wir doch einige Librarys verwenden, hätte ich für das Dependency Management Maven verwendet.

19 Abschlussberichte

19.1 Abschlussbericht Stefan Züger

Die 14 Arbeitswochen unserer Studienarbeit sind bereits vorüber, weshalb ich die Gelegenheit nutzen möchte meine Erkenntnisse und Erfahrungen kurz festzuhalten.

Wie bereits in meinem Zwischenbericht (Kapitel *18.1*) erwähnt, war anfangs eine gewisse Skepsis in Bezug auf das ganze Scrum Projekt vorhanden. Niemand im ganzen Team hatte jemals eine solche Arbeit mit Scrum durchgeführt, weshalb auf keine Erfahrungen zurückgegriffen werden konnten. Doch nach 14 Wochen intensiver Arbeit kann ich fast nur Positives zum gesamten Projekt Management hervorheben. Ich hätte nicht gedacht, dass alles bereits von Anfang an so gut klappt, wie es nun der Fall war. Ein grosser Teil dieses Erfolgs können wir Thomas Kälin verdanken. Er hat seine Rolle als Scrum Master hervorragend wahrgenommen, war für alle Meetings immer bestens vorbereitet und zeigte sich immer sehr offen für Verbesserungsvorschläge. Aus dem gesamten Projekt Management konnte ich wohl einiges für spätere Projekte lernen. So bin ich mir jetzt schon im Klaren, dass ich einige Teile des gesamten Scrum Prozesses in meiner Bachelorarbeit einsetzen möchte. Dabei werden wir sicherlich nicht mehr alles von Scrum so gut umsetzen können, wie in einem so grossen Team wie jetzt. Doch einige Artefakte und Tätigkeiten möchte ich bestimmt wiederverwenden. Gleiches gilt auch für die verwendeten XP Praktiken.

Bezüglich der technischen Seite unserer Arbeit kann ich sagen, dass ich mit dem Erreichten doch sehr zufrieden sein kann. Hierbei möchte ich besonders die Qualität des geschriebenen Codes erwähnen. Dies war mein erstes Projekt, bei dem von Anfang an konsequent darauf geachtet wurde, dass für alle Klassen Javadoc und JUnit Tests geschrieben wurden. Dies war anfangs jedoch ein wenig mühsam und gewöhnungsbedürftig, letztlich haben wir aber dadurch eine Test Abdeckung von über 92% und einen sehr gut Dokumentierten Source Code erhalten. Allgemein konnte ich bezügliche moderner Software-Entwicklung einiges lernen, was mir für spätere Arbeiten sicherlich helfen wird.

Zum Schluss möchte ich mich noch bei unserem Betreuer Prof. Dr. Lothar Müller für seine tolle Betreuung der Studienarbeit bedanken. Seine Feedbacks waren stets sehr konstruktiv und ausführlich, so dass wir immer genau wussten, woran wir stehen. Nicht selbstverständlich waren auch seine intensiven Tests der Software am Ende jedes Sprints, dank denen wir doch noch einige Verbesserungen vornehmen konnten. Ein zusätzlicher Dank richtet sich bestimmt

auch an Thomas Kälin für seine hervorragende Arbeit als Scrum Master und an meinen Studienarbeitspartner Tobias Zürcher für die super Zusammenarbeit, die stets konstruktiven Design-Diskussionen und für seinen geleisteten Einsatz!

19.2 Abschlussbericht Tobias Zürcher

14 Wochen Semerstarbeit sind nun vorbei. Gerne blicke ich auf diese intensive und interessante Zeit zurück. Einige Gedanken zum Projekt habe ich bereits im Zwischenbericht (Kapitel *19.2 Abschlussbericht Tobias Zürcher*) erläutert.

Scrum konnte mich bis zum Ende des Projektes überzeugen. Dies ist auch auf unseren Scrum Master Thomas Kälin zurückzuführen, welcher eine hervorragende Arbeit geleistet hat! Er war immer offen für Verbesserungsvorschläge und bereitete seine Meetings beispielhaft vor. Auch kleine Details wie zum Beispiel einzelne Unittests die vergessen gegangen sind, hatte er stets unter Kontrolle. Bezüglich Projekt Management konnte ich einiges von ihm profitieren, was mir bei zukünftigen Projekten eine grosse Hilfe sein wird.

Mit unserem Endprodukt bin ich sehr zufrieden. Es macht Spass, die Anwendung mit „realen" Daten aus dem IFS Test anzuschauen. Im technischen Bereich möchte ich speziell unseren Code herausheben, welcher sehr vorbildlich mit Javadoc dokumentiert und zu 92% mit Unit-Tests abgedeckt ist. Bis jetzt habe ich noch nie so konsequent auf solche Details geachtet, werde dies jedoch in Zukunft wieder tun, weil sich der Mehraufwand lohnt.

Zum Schluss möchte ich mich herzlich bei unserem Betreuer Prof. Dr. Lothar Müller für seine vorbildliche Betreuung bedanken. Seine umfangreichen Feedbacks waren sehr hilfreich. Bei Fragen oder Problemen stand er uns jederzeit zur Verfügung.

Ein letzter Dank geht an meinen Studienarbeitspartner Stefan Züger für seinen mustergültigen Einsatz. Die vielen konstruktiven Diskussionen sowie das Pair Programming habe ich sehr geschätzt. Es ist bereits unsere dritte gemeinsame Arbeit und ich freue mich mit ihm die Bachelorarbeit in Angriff zu nehmen.

20 Reflexionen zu angewandten Methoden

20.1 Scrum

Wie bereits in den beiden persönlichen Zwischenberichten (Kapitel *18 Zwischenberichte*) beschrieben, waren wir schon zu Beginn sehr zufrieden mit Scrum. Diese Ansicht hat sich bis zum Projektende nicht geändert. Trotzdem wollen wir zusammenfassende Gedanken zu Scrum in diesem Kapitel festhalten.

20.1.1 Positive Aspekte

Aus Entwicklersicht punktet Scrum besonders mit seinem ergebnisorientiertem Charakter. Die Tasks werden im Voraus vom Product Owner definiert. Im Sprint Planning Meeting werden Fragen der Entwickler geklärt, so dass die Tasks fast ohne weitere Abklärungen abgearbeitet werden können. Mit den Daily Meetings sind alle Entwickler auf dem aktuellen Stand.
Es gibt keine monatelangen Entwicklungszeiten, wo grosse Releases entstehen. Durch die kleinen Iterationen wächst die Software inkrementell, dies gibt dem Product Owner Kontrolle über das Produkt und ermöglicht ein frühzeitiges Eingreifen.

20.1.2 Negative Aspekte

Die Entwickler geniessen bei Scrum viele Freiheiten, dies fordert einen hohen Grad an Selbstorganisation und Disziplin. Scrum kann nicht funktionieren, wenn dominante Teammitglieder Aufgaben verteilen und so die Selbstorganisation stören. Die Entwicklungsabteilung muss auf hohem Niveau und mit modernen Programmiersprachen arbeiten, damit in solch kurzen Abständen lieferbare und qualitätsgetestete Software ausgeliefert werden kann. Scrum fordert eine sehr hohe Abdeckung durch Unittest, damit auch bei häufigen Änderungen stabilen Code produziert werden kann. Dies steht mit den kurzen Iterationen im Widerspruch: Die Zeit für ausgiebige Unittests ist meistens knapp. Die Flexibilität birgt auch einen Nachteil: Durch häufige Änderungen ist der Aufwand an Refactoring sehr gross.

Der wichtigste Negativpunkt in unseren Augen ist der fehlende Gesamtüberblick. Dies ist eine Eigenschaft, mit der agile Vorgehensmodelle Mühe haben. Des Weiteren besteht ein mangelnder architektonischer Überblick durch die fehlende Designphase.

20.1.3 Welche Projekte eignen sich für Scrum?

Gemäss Wikipedia *[URL13]* wird bei Scrum grundsätzlich angenommen, dass der Entwicklungsprozess so komplex ist, dass dieser weder in geschlossene Phasen noch Arbeitsschritte unterteilt werden kann. Aus unserer Sicht eignet sich Scrum auch für kleinere Projekte. Wichtig ist, dass eine Rollenteilung vorhanden ist: Zum Beispiel sollte der Product Owner nicht gleichzeitig ein Entwickler sein. Falls dies nicht möglich ist, können die vielen kleinen Elemente von Scrum als „Baukasten“ verwendet werden: Daily Meetings, Product Backlog, Burndown Chart etc. sind gute Hilfsmittel um ein Projekt zu zweit zu führen. Auch im VisiVis Projekt musste Scrum etwas an unsere Bedürfnisse angepasst werden und Elemente wie zum Beispiel die Next-Tasks oder ein Aufräum-Task eingeführt werden.

Damit bei den häufigen Änderungen stabiler Code produziert werden kann, muss eine moderne Programmiersprache verwendet werden, welche auch Hilfsmittel wie Unittesting und Refactoring unterstützt.

Da Scrum sehr entwicklungslastig ist, müssen einige Vorarbeiten bereits gegeben sein: Anforderungsspezifikation, Design (insbesondere Grundarchitektur) und grundlegende Analysen. Anforderungen dürfen vor oder nach einem Sprint angepasst werden. Projekte, bei denen grössere Vorstudien oder erweiterte Analysen gemacht werden müssen eignen sich nicht für Scrum. Je nach Grösse könnte in der Entwicklungsphase Scrum verwendet werden.

20.1.4 Eignete sich Scrum für VisiVis?

Die etwas spezielle Konstellation des VisiVis Projekts eignet sich sehr für Scrum. Die Architektur und die Anforderungen (Product Backlog) waren zu Projektbeginn in einem vernünftigen Rahmen gegeben. Alle Teams arbeiteten in einem isolierten Bereich. So konnte der im Kapitel *20.1.2 Negative Aspekte* erwähnte Negativpunkt bezüglich der mangelnden architektonischer Überblick entgegen gewirkt werden.

Trotz dieser Isolation gab es gemeinsame Teile, an denen über die Gruppe hinaus gearbeitet wurde. Mithilfe der Daily Meetings konnten gemeinsame Probleme so angesprochen und gelöst werden.

20.2 Extreme Programming Praktiken

Zu Beginn des Projektes hat sich das Projektteam darauf geeinigt zehn XP Praktiken anzuwenden. (siehe Kapitel *4.2.1 Im Projekt berücksichtigte Extreme Programming Praktiken*) Nachfolgend eine kurze Zusammenstellung aller Praktiken und die entsprechenden Feedbacks zu jedem einzelnen Punkt:

20.2.1 Planning Game

Das gemeinsame Schätzen von Aufwänden mit Hilfe des Planning Poker Spiels war für alle Projektteilnehmer neu. Trotzdem hat es schon vom ersten Sprint an sehr gut funktioniert und es gab einige Male, an denen die Zeitschätzungen der einzelnen Personen sehr unterschiedlich ausgefallen sind. Dies führte dazu, dass man sich nochmals genauere Gedanken über die Aufwände einer User Story machte.

Leider kam es in den ersten Sprints trotz des Planning Games oftmals dazu, dass das ganze Team die Aufwände unterschätzt hat, weshalb nicht immer alle Tasks bis zum Ende eines Sprints fertiggestellt werden konnten. Das Planning Game sollte jedoch genau solche Punkte verhindern. Das Team lernte aber aus den vergangenen Sprints, so dass die Schätzungen gegen Ende des Projekts immer genauer wurden.

Zusätzlich muss beim Planning Game darauf geachtet werden, dass man sich nicht in Detaildiskussionen verliert, bei der auf Stundenbasis verhandelt wird, wieso ein Task länger oder kürzer dauert. Denn letztlich handelt es sich immer noch um eine Schätzung, welche nie auf die Stunde genau sein wird.

Nichtsdestotrotz würden wir das Planning Game wieder anwenden, wenn es sich um ein Projekt dieser Grösser handelt.

20.2.2 Small Releases

Die Dauer eines Sprints von zwei Wochen war für ein Projekt mit einer Gesamtdauer von 14 Wochen gerade ideal. Kürzere Sprints wären zu unproduktiv, da nur sehr kurze User Storys innerhalb eines Sprints erledigt werden könnten. Längere Sprints wären auch nicht besser, da die Schätzungen schwieriger werden, je mehr Zeit zur Verfügung steht.

20.2.3 Simple Design

Die XP Praktik des inkrementell wachsenden, simplen Designs ist in unserem Projekt sehr schön an der Besucherauswahl ersichtlich. Zuerst war die Besucherauswahl nur statisch über den Code möglich. Danach kam eine erste Prototyp Version als Popup, bei welchem die Möglichkeit vorhanden war, eine einzelne Person und deren Etage auszuwählen. In einem weiteren Sprint war die Besucherauswahl in einem Dockable möglich, wobei das Museum und die Etage unabhängig vom Besucher gewechselt werden konnte. Zum Schluss wurde die endgültige Version umgesetzt, die eine Besucher und Gruppenauswahl erlaubte.

Dieses Beispiel zeigte sehr gut das wachsende Design, doch dies bedeutete in unserem Fall einen ziemlichen Mehraufwand durch die mehrfach getätigten Analysen und Implementie-

rungen. Es ist uns bewusst, dass nicht von Anfang an eine definitive Lösung entworfen werden kann, doch spätestens zur Projekthalbzeit sollte eine Ziellösung im Hinterkopf sein, damit beim Entwickeln einer neuen Komponente an die endgültige Variante gedacht werden kann.

20.2.4 Refactoring

Zum Refactoring gibt es nichts Grosses zu erwähnen, ausser dass es bei einem agilen Softwareentwicklungsprozess wie Scrum nicht wegzudenken wäre. Glücklicherweise bietet Eclipse mittlerweile eine sehr optimale integrierte Refactoring Unterstützung, so dass dies gar kein grosses Thema mehr für uns war.

20.2.5 Testing

Anfangs dachten wir, eine Unittest Abdeckung von mehr als 90% sei ein reiner Overhead und nicht wirklich hilfreich. Glücklicherweise hat aber unser Scrum Master Thomas Kälin von Anfang an darauf geachtet, dass die Unittests überall geschrieben werden, wo es möglich ist. So haben wir zum Schluss tatsächlich eine Testabdeckung von über 90% erreicht.

Rückblickend können wir nun sagen, dass es in einem agilen Softwareentwicklungsprozess sehr hilfreich ist, wenn zuverlässige Unittests vorhanden sind. Denn gerade nach grösseren Refactoring Aktionen ist man sehr beruhigt, wenn man Unittests hat, die zum Schluss bestätigen, dass alles noch so läuft wie vor dem Refactoring.

Nebenbei kann auch angemerkt werden, dass wir während der gesamten Projektdauer nie einen schwerwiegenderen Bug entdeckten, welcher nicht innert kurzer Zeit behoben werden konnte. Natürlich kann man diese Tatsache nicht einfach alleine mit den vorhandenen Unittests begründen, doch diese haben bestimmt ihren Anteil beigetragen.

20.2.6 Pair Programming

Wie bereits in den persönlichen Berichten erwähnt, kamen wir bei diesem Projekt sehr oft zum Einsatz des Pair Programming. Bei anspruchsvolleren Entwicklungstätigkeiten, wie bei uns zum Beispiel die Implementierung des A*-Algorithmus, bietet sich diese Praktik sehr gut an. Unsere Erfahrung zeigt, dass zu zweit erstellter Code von Anfang an eine bessere Qualität und weniger Fehler aufweist. Nebenbei kann man gegenseitig von den Vorgehensweisen des anderen lernen. Pair Programming ist etwas, dass wir bestimmt auch in unserer Bachelorarbeit des Öfteren einsetzen werden.

Pair Programming hat aber auch seine Einschränkungen und soll in unseren Augen nicht in jedem Fall angewendet werden. Einfachere Entwicklungstätigkeiten sollten wenn möglich immer alleine umgesetzt werden, da man zu zweit viel zu ineffizient ist. Da kann es genügen, wenn der allein erstellte Code zu einem späteren Zeitpunkt einem kurzen Review eines anderen Entwicklers unterzogen wird. Des Weiteren ist das Pair Programming nur effizient und für beide Entwickler gewinnbringend, wenn beide ungefähr auf demselben Niveau sind.

20.2.7 Collective Ownership

Der Punkt des Collective Ownership müsste unserer Meinung nach nicht explizit als XP Praktik erwähnt werden, da wir dies in einem gemeinsamen Projekt als selbstverständlich ansehen. Trotzdem kann es in einem Projekt hilfreich sein, diesen Punkt nochmals extra aufzuführen, damit sich alle nochmals der Sache bewusst werden.

In anderen Projekten kann es doch vorkommen, dass gewisse Entwickler sich angegriffen fühlen, wenn in ihrem Code Änderungen vorgenommen werden. Dies war in unserem Projekt zum Glück nie der Fall.

20.2.8 Continous Integration

Dank der sehr sauberen Installation unseres Build Servers, sind wird mit diesem Punkt gar nie in Kontakt gekommen. Da geschah der ganze Buildprozess immer automatisch, wobei wir sogar per E-Mail benachrichtigt wurden, sobald etwas schief lief. Bei einem weiteren Java Projekt in diesem Umfang würden wir bestimmt wieder einen Build Server wie Hudson verwenden.

20.2.9 Ten-Minute Build

Die XP Praktik des Ten-Minute Build könnte in unseren Augen ohne Probleme weggelassen werden, da das Kompilieren und Builden einer Java Anwendung in diesem Grössenumfang sowieso niemals über zehn Minuten dauert. Der Build Taks auf dem Hudson Server hat bei uns nie länger als eine Minute gedauert.

20.2.10 Coding Standards

Das Einhalten der Java Codierungsrichtlinien ist uns anfangs ziemlich schwer gefallen, da wir uns einen leicht anderen Stil angeeignet hatten. Doch mit Hilfe des Checkstyle Plug-Ins für Eclipse war es überhaupt kein Problem mehr sich daran zu halten.

Gerade bei einem Projekt, an dem mehr als nur zwei Entwickler mitarbeiten, macht es durchaus Sinn, solche Coding Standards durchzusetzen. Dies vor allem in Hinblick auf Folgeprojekte, welche mit dem bestehenden Code weiterarbeiten müssen.

Teil 7: Anhang

21 Aufgabenstellung

Studienarbeit, HS 2009

Softwareunterstützung für ein Museum - Visualisierungen: Wegrekonstruktionen

Aufgabenstellung

Thematik

Wie kann man etwas über das Verhalten von Besuchern in einem Museum erfahren? Welche Räume werden am meisten besucht? Was sehen sich die Besucher an? Wo bleiben sie wie lange? Welchen Weg nehmen sie durch das Museum? Ausgehend von einem konkreten Projekt wurde in mehreren Vorgängerarbeiten untersucht, wie man diese Fragen mittels RFID-Technologie beantworten kann.

In den Vorgängerarbeiten wurden

- Prototypen entwickelt: Erfassung von Gebäuden, Simulation von Besucherbewegungen, verschiedene Visualisierungen,
- die Hardware getestet und
- die Prototypen wurden in einem Feldtest unter realen Museumsbedingungen getestet.

Aufbauend auf den vorliegenden Erfahrungen soll nun ein produktiv einsetzbares System entwickelt werden. Dabei sollen in Rahmen von 2 Studienarbeiten die Visualisierungen realisiert werden, wobei es sich zum Teil um verbesserte Versionen der Prototypen, zum Teil um neue Visualisierungen handelt.

Besonderheiten

- Beide Arbeiten bauen auf einem vorgegebenen Datenmodell und einer z.T. vorgegebenen Problemdomain auf.
- Parallel zu den Studienarbeiten werden die Erfassung von Gebäuden, die Anbindung an die Hardware und weitere Funktionen neu entwickelt.
- Die zu realisierenden Visualisierungen sollen sich stets in das Gesamtsystem einfügen.
- Die Leitung des Gesamtprojekts übernimmt Thomas Kälin im Rahmen seiner MSE-Masterarbeit. Er übernimmt auch die Verantwortung für ein integriertes Vorgehen.
- Für das Projektmanagement wird als SE-Methode Scrum verwendet.
- Es wird erwartet, dass die Entwicklungsumgebung (Projektwiki, Trac, FEST, Hudson Build, usw.) aktiv genutzt wird.

Thema dieser Arbeit & Aufgabe

Im Rahmen dieser Arbeit sollen die Visualisierungen für die Wegrekonstruktionen realisiert werden. Dabei handelt es sich beispielsweise um

- Weg einer Person in verschiedenen Darstellungen
- Wege mehrerer Personen bzw. von Gruppen in verschiedenen Darstellungen

Die konkret zu realisierenden Funktionen werden im Laufe des Projekts iterativ festgelegt und jeweils in den Projektsitzungen abgesprochen.

Es soll ingenieurmässig vorgegangen werden.

Es ist nach einem Projektplan zu arbeiten. Der tatsächliche Arbeitsaufwand ist zu erfassen.

Erwartete Resultate

Lösung der vorstehend beschriebenen Aufgabe.

Abzugeben ist ein Bericht (in 1 Exemplar auf Papier, in 2 Exemplaren als CD-ROM), welcher enthält:

- die Aufgabenstellung der Arbeit (im Original),
- eine Zusammenfassung der Arbeit auf einer Seite auf dem dafür vorgegebenen Formular,
- eine Management-Summary,
- einen Bericht über die Arbeit,
- die bei der Arbeit erstellten Dokumente,
- die erstellen Programme (als Softcopy),
- den Projektplan und die tatsächlichen Aufwände.

Aus dem Bericht muss klar hervorgehen, wer für welchen Teil der Arbeit und des Berichts verantwortlich ist.

Bei der im Projekt vorgesehenen agilen Vorgehensweise ist zu beachten, dass die für die Endabgabe erforderlichen Dokumente begleitend entstehen. Dies sind insbesondere:

- Anforderungen (in geeigneter Form)
- ggf. Problemanalysen
- UI-Entwürfe
- Dokumentation des Designs, z.B. Design-Entscheide, JavaDoc
- Testdokumentation

Rechte

Die Rechte werden in der nachstehenden Vereinbarung geregelt.

Termine

Abgabe der Aufgabenstellung und Beginn der Arbeit: 14.9.2009

Abgabe des Berichts und Ende der Arbeit: 18.12.2009, 17:00 h

Betreuung

Betreuer: Lothar Müller

Mit allen Beteiligten einschliesslich dem Betreuer werden regelmässige Sitzungen durchgeführt. Bei Bedarf können weitere Termine vereinbart werden.

Rapperswil, 14.9.2009

Prof. Dr. Lothar Müller

Vereinbarung

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit "Softwareunterstützung für ein Museum - Visualisierungen: Wegrekonstruktionen" von Tobias Zürcher und Stefan Züger unter der Betreuung von Prof. Dr. Lothar Müller geregelt.

2. Urheberrecht

Die Urheberrechte stehen den Studenten zu.

3. Verwendung

Die Ergebnisse der Arbeit dürfen von der HSR nach Abschluss der Arbeit verwendet und weiter entwickelt werden.

Rapperswil, den 14.9.2009 .....

Tobias Zürcher

Rapperswil, den 14.9.2009 .....

Stefan Züger

Rapperswil, den 14.9.2009 .....

Prof. Dr. Lothar Müller

22 Glossar

Begriff	Beschreibung
A* Algorithmus	Suchalgorithmus, der den kürzesten Pfad zwischen zwei Knoten berechnet. Er wurde das erste Mal 1968 veröffentlicht *[HART68]*.
Barcode Reader	Lesegerät, welches Barcodes (Strichcodes) lesen und registrieren kann. Dient in der Museumsanwendung dazu, Besucher am Eingang des Museums zu erfassen.
Controller	Schicht vom *Model View Controller* Pattern *[GAMMA95]*
Dockable	Kleines Fenster einer Anwendung, welches vom Benutzer an verschiedenen Stellen im GUI „gedockt“ werden kann.
Javadoc	Software-Dokumentationswerkzeug, welches aus Java Source Code HTML Dateien generiert, die alle Kommentare des Source Codes visuell darstellen.
Reader	Lesegerät, welches die ausgesendeten Signale von RFID Transpondern empfängt und registriert.
Recordal	Aufzeichnung eines RFID Signals beim RFID Reader. Beinhaltet neben der Tag-ID (Identifikation des Besuchers) den Aufnahmezeitpunkt, sowieso die Identifikation des aufzeichnenden RFID Readers.
RFID	Radio Frequency Identification Auf Deutsch bedeutet dies so viel wie die Identifizierung mit Hilfe von elektromagnetischen Wellen. Ein RFID System besteht im Wesentlichen aus einem Transponder, der sich an einem Gegenstand bzw. Lebewesen befindet und dabei elektromagnetische Signale aufnimmt und automatisch beantwortet bzw. weiterleitet. Auf der Empfangsseite ist ein RFID Reader, der die zurückgesendeten Signale vom Transponder empfängt. *[URL01]*
RUP	Rational Unified Process: Vorgehensmodell der Firma Rational Software und benutzt Unified Modeling Language als Notationssprache.
SVN	Software für das Versionsmanagement der Dokumente und Programmcode
Swing	Java UI Framework/Library
SZ	Stefan Züger
Test Suite	Sammlung von Unittests
Tracking-Device	Technische Bezeichnung für ein Empfangsgerät. Im Kontext der VisiVis Studienarbeit handelt es sich bei einem TrackingDevice immer um einen RFID oder einem Barcode Reader.
TreeTable	GUI Komponente, welche einen Tree (Baumstruktur) mit einer Tabelle vereint.

TZ	Tobias Zürcher
UI, User Interface	Grafische Benutzerschnittstelle einer Software-Anwendung
VisiVis	Projektname, Abkürzung für „Visitor Visualisation“
XP	Extreme Programming; definiert Praktiken für das Softwareengineering

Tabelle 30: Glossar

23 Verzeichnisse

23.1 Tabellenverzeichnis

23.2 Abbildungsverzeichnis

23.3 Literaturverzeichnis

[Beck00] Beck, K.; Fowler, M. 2000 *Planning Extreme Programming*. 1st. Addison-Wesley Longman Publishing Co., Inc.

[Gamma95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. 1995 *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc.

[Hart69] Hart, P.E.; Nilsson, N.J.; Raphael, B., *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, Systems Science and Cybernetics, IEEE Transactions on , vol.4, no.2, pp.100-107, July 1968

[Kälin07] Kälin Thomas, Ferrari Fabio, *Softwareunterstützung für ein Museum - Visualisierung von Besucherbewegungen*, HSR, Rapperswil, Studienarbeit Herbstsemester 2007

[Kälin08] Kälin Thomas, Ferrari Fabio, *Softwareunterstützung für ein Museum - Visualisierung von Besucherbewegungen*, HSR, Rapperswil, Bachelorarbeit Frühjahrssemester 2008

[Milne03] Milne P. 2003, *Creating TreeTables in Swing*, http://java.sun.com/products/jfc/tsc/articles/treetable1, letzter Zugriff am 12.12.2009

[POSA3] Kirchner M.; Prashant J., *Pattern-Oriented Software Architecture – Patterns for Resource Management*, John Wiley & Sons, Inc., 2004

[Violet03] Violet S.; Walrath K. 2003, *Creating TreeTables: Part 2 - Completing the Example Program,* http://java.sun.com/products/jfc/tsc/articles/treetable2, letzter Zugriff 12.12.2009

[Url01] Wikipedia Artikel zu RFID http://de.wikipedia.org/wiki/RFID, letzter Zugriff 12.12.2009

[Url02] Einführung in XP von Don Wells http://www.extremeprogramming.org, letzter Zugriff 12.12.2009

[Url03] Java Codierungsrichtlinien http://java.sun.com/docs/codeconv, letzter Zugriff 12.12.2009

[Url04] Open Source Software Checkstyle zum Überprüfen der Einhaltung von Java Codierungsrichtlinien http://checkstyle.sourceforge.net, letzter Zugriff 12.12.2009

[Url05] Open Source Software Findbugs zur statischen Codeanalyse http://findbugs.sourceforge.net, letzter Zugriff 12.12.2009

[Url06] Open Source Software Cobertura zur Überprüfung der Code Abdeckung http://cobertura.sourceforge.net, letzter Zugriff 12.12.2009

[Url07] Open Source Java Build Server http://hudson.dev.java.net, letzter Zugriff 12.12.2009

[Url08] Online Zeiterfassungstool Paymo http://www.paymo.biz, letzter Zugriff 12.12.2009

[Url09] Wikipedia Artikel zu Separations of Concerns
http://en.wikipedia.org/wiki/Separation_of_concerns, letzter Zugriff 15.12.2009

[Url10] Ausführliche theoretische Dokumentation über das Vorgehensmodell Scrum
http://www.scrum-master.de, letzter Zugriff 16.12.2009

[Url11] Agile Methodology Training and Project Management Consulting Firma gegründet im Jahre 1993
http://www.mountaingoatsoftware.com, letzter Zugriff 16.12.2009

[Url12] Wikipedia Artikel zu High Cohesion
http://en.wikipedia.org/wiki/High_cohesion, letzer Zugriff 17.12.2009

[Url13] Wikipedia Artikel zu Scrum
http://de.wikipedia.org/wiki/Scrum, letzter Zugriff 17.12.2009

24 Erklärung über die eigenständige Arbeit

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Ort, Datum: Stefan Züger

_____ _____

Ort, Datum: Tobias Zürcher

_____ _____