Softwareunterstützung für ein Museum
Visualisierungen: Statistiken

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2009/2010

Autoren:	Patrick Oswald,
	Simon Inderbitzin
Betreuer:	Prof. Dr. Lothar Müller
Projektpartner:	Albis Technologies Zürich, Naturama Aarau
Gegenleser:	Wolfgang Giersche

Eigenständigkeitserklärung

Erklärung

Hiermit erklären wir,

- 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:

Ort, Datum:

Name, Unterschrift:

Name, Unterschrift:

17. Dezember 2009

1 Dokumentinformationen

1.1 Gültigkeitsbereich

Der vorliegende Bericht beschreibt das Ergebnis der Studienarbeit zum Thema „Softwareunterstützung für ein Museum – Visualisierungen: Statistiken“. Das Dokument ist gültig während der gesamten Projektdauer. Alle Änderungen wurden in den einzelnen Sprints oder im Kapitel Änderungsgeschichte nachgeführt.

2 Inhaltsverzeichnis

3 Änderungsgeschichte

Datum	Änderung
04.12.2009	Erstellung des Dokumentes.
07.12.2009	Erste Grundstrukturen.
07.12.2009	Einheitliches Design gewählt mit Kopf- und Fusszeilen.
10.12.2009	Aktualisierung der Struktur.
10.1.2009	Literaturstruktur erstellt.
15.12.2009	Hineinkopieren aller fehlenden Dokumente.
16.12.2009	Orthografische Verbesserung mit kleinen Umstrukturierungen.
17.12.2009	Finalisierung des Dokumentes.
18.12.2009	Letzte orthografische Verbesserungen.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Studienarbeit, HS 2009

Softwareunterstützung für ein Museum - Visualisierungen: Statistiken

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.

www.hsr.ch Oberseestrasse 10 CH-8640 Rapperswil
T ++41 (0)55 222 49 35 F ++41 (0)55 222 44 00 lothar.mueller@hsr.ch Mitglied der Fachhochschule Ostschweiz FHO

Thema dieser Arbeit & Aufgabe

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

- Anzahl Personen pro Reader oder pro Raum
- Aufenthaltsdauern pro Reader oder pro Raum
- Auswahl des Zeitraums der Auswertung
- Auswahl der Personengruppen der Auswertung (demografische Daten, Gruppen)

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: Statistiken" von Simon Inderbitzin und Patrick Oswald 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 .....

Simon Inderbitzin

Rapperswil, den 14.9.2009 .....

Patrick Oswald

Rapperswil, den 14.9.2009 .....

Prof. Dr. Lothar Müller

5 Einordnung der Arbeit

Unsere Studienarbeit stellt nur ein Teilbereich der gesamten vom Team erarbeiteten Applikation dar. Zudem besteht eine Vielzahl an Vorgängerarbeiten, welche nachfolgend aufgelistet werden. Der Blick über die eigenen Projektgrenzen hinweg ist notwendig, damit unsere Arbeit in einen Gesamtkontext gestellt werden kann. In folgendem Kapitel widmen wir uns einer kurzen Zusammenfassung der Ausgangslage und der aktuellsten Version der Museumsapplikation.

5.1 Vorgängerarbeiten

Bestehende Arbeiten zum Thema „Museumsapplikation“:

- 07 / HS / Studienarbeit: Betschart, Thoma (Simulator)
- 07 / HS / Studienarbeit: Ferrari, Kälin (Visualisierung)
- 08 / FS / Bachelorarbeit: Ferrari, Kälin (Gebäudeerfassung, RFID-Hardware)
- 08 / HS / Diplomarbeit: Reichenbacher (Realistischere Simulationen)
- 08 / HS / Diplomarbeit: Rüegg (Schleusen)
- 08 / HS / MSE: Kälin (Bedürfniserhebung)
- 08 / HS / Studienarbeit: Eberle, Federer (RFID-Hardware)
- 09 / FS / MSE: Kälin (Mobile Navigationsystem)
- 09 / FS / Bachelorarbeit: Kohler, Rotta (Feldtest Naturama)

5.2 Aktuelle Arbeiten

Für diese Studienarbeit wurden gleich zwei Arbeiten ausgeschrieben:

- 09 / HS / SA: Softwareunterstützung für ein Museum - Visualisierungen: Statistiken
    - Zuständig: Tobias Zürcher / Stefan Züger
- 09 / HS / SA: Softwareunterstützung für ein Museum - Visualisierungen: Wegverfolgung
    - Zuständig: Patrick Oswald / Simon Inderbitzin

Diese zwei Themen bauen auf den erarbeiteten Grundlagen der Vorgängerarbeiten auf. Aus diesem Grund ist die Datenbankstruktur, Problemdomain und Architektur bereits vorgegeben. Der Fokus unserer Arbeit liegt bei der Auswertung und Visualisierung von RFID Erfassungen.

Zusätzlich zu den beiden Studienarbeiten laufen parallel folgende Projekte:

- 09 / HS / Masterarbeit: Softwareunterstützung für ein Museum: Gebäudeerfassung, Anwendungskern und Projektmanagement.
    - Zuständig: Thomas Kälin
- Externer Mitarbeiter vom IFS: RFID-Hardware mit Kassenapplikation.
    - Zuständig: Michael Rüegg

Zusammengefasst lässt sich folgendes visualisieren:

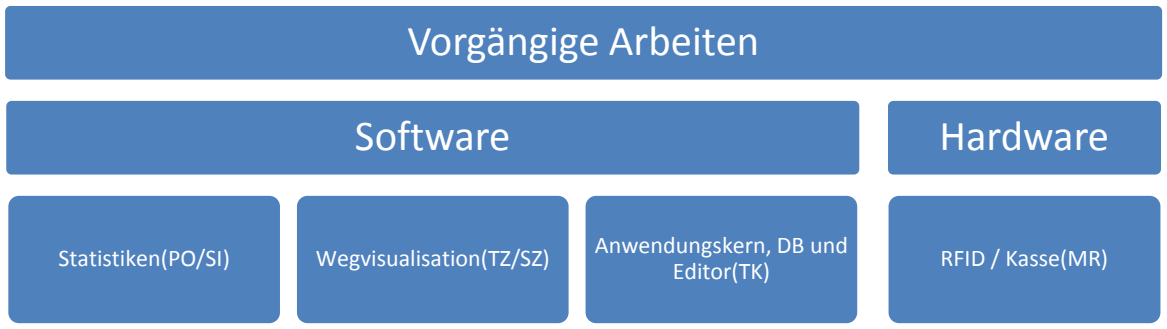


Abbildung 1: Aufbau und Verantwortungsbereich der aktuellen Software.

Personenlegende	
TK	Thomas Kälin
MR	Michael Rüegg
PO	Patrick Oswald
SI	Simon Inderbitzin
TZ	Tobias Zürcher
SZ	Stefan Züger

6 Abstract / Kurzfassung

Abteilung	**Informatik**
Name[n] der Studierenden	**Patrick Oswald, Simon Inderbitzin**
Studienjahr	**HS 09/10**
Titel der [Studien-/Diplom-/Bachelorarbeit]	**Softwareunterstützung für ein Museum – Visualisierungen: Statistiken**
Examinatorin / Examinator	**Prof. Dr. Lothar Müller**
Themengebiet	**Software**
Projektpartner	**Albis Technologies Zürich, Naturama Aarau**
Institut	**IFS: Institut für Software**

Der Themenbereich "Softwareunterstützung für ein Museum" wurde schon durch mehrere Studien-/Bachelor- und Diplomarbeiten analysiert. Bis dato fehlte jedoch eine aussagekräftige und geeignete Applikation für die Visualisierung der gesammelten Erfassungen. Basierend auf den vorliegenden Erfahrungen, mit Messdaten, Problemdomain, unzähligen Analysen und einigen Prototypen entwickelten wir als 2er Gruppe in einem Team von 6 Entwicklern ein produktiv einsetzbares System zur Visualisierung der gesammelten Messdaten.

Unser Schwerpunkt lag dabei auf der Darstelllung von unterschiedlichen statistischen Auswertungen. Dies ermöglicht (z.B. der Museumsleitung) eine genaue Analyse des Besucherverhaltens. Die statistischen Auswertungen werden dabei auf den Grundriss einer Museumsetage projiziert, oder in einem separaten Graphen dargestellt. Im Rahmen dieser Arbeit wurden unterschiedliche statistische Visualisierungen realisiert, dabei handelt es sich beispielsweise um:

- Anzahl Personen pro Reader / Raum und
- Aufenthaltsdauern pro Reader / Raum
- Auswahl des Zeitraums der Auswertung
- Auswahl der Personengruppen der Auswertung (demografische Daten, Gruppen)
- Regulierung der Farbskalen (dient der visuellen Untermalung dargestellter Zahlen)
- Statistiken über das ganze Museum
- Demographische Verteilung durch Diagrammen (Geschlecht, Alter usw.)

Das ganze Projekt wurde primär in Java mit Eclipse entwickelt. Als Schnittstelle dient ein MySQL-Server, der für die Persistenz des gemessenen Besucherverhaltens verantwortlich war. SVN, Hudson, Paymo, Dropbox, Wiki und viele mehr zeigten sich als nützliche Hilfsmittel um eine optimierte Arbeitsumgebung und Arbeitsablauf zu schaffen. Basierend auf diesem Produkt können die Besucherdaten einfach und aussagekräftig dargestellt werden. Für weitere Visualisierungen mitsamt produktiven Feldtests wurde nun der Grundstein gelegt.

7 Management Summary

7.1 Ausgangslage

Wie können Informationen über das Verhalten von Besuchern in einem Museum ausgewertet werden? Welche Räume / Objekte wurden am Meisten frequentiert oder lange begutachtet?

Diese und weitere Fragen wurden in vorherigen Studien- und Bachelorarbeiten analysiert und Prototypen für die Erfassung von Gebäuden entwickelt. Aufbauend auf diesen vorliegenden Erfahrungen sollen im Rahmen dieser Arbeit die statistischen Visualisierungen neu implementiert werden, wobei es sich zum Teil um verbesserte Versionen der Prototypen handelt.

7.2 Vorgehen

Die konkret zu realisierenden Funktionen wurden im Laufe des Projekts iterativ festgelegt und jeweils in den Projektsitzungen abgesprochen. Insgesamt wurden sechs zweiwöchige Iterationen durchgeführt, was eine sehr agile Softwareentwicklung mit testbaren Endergebnissen ermöglichte. Mängel oder Verbesserungen konnten bereits früh erkannt und in der nächsten Iteration umgesetzt werden.

7.3 Ergebnisse

Die wichtigsten Ergebnisse werden nachfolgend mit Bildern detaillierter illustriert und erläutert:

- Auswertungen pro RFID Reader oder pro Raum (mit Farben untermalt):

Abbildung 2: Auswertungen pro RFID Reader

Abbildung 3: Auswertung pro Raum

- Auswertungen von Besucherzahlen (absolut / relativ), Aufenthaltsdauer und Gesamtbesucherzeit:

Abbildung 4: Absolute Besucherzahl

Abbildung 5: Prozentuale Darstellung

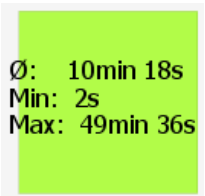


Abbildung 6: Aufenthaltsdauer

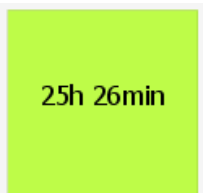


Abbildung 7: Verweildauer

- Anzeigen demographischer Verteilungen als Kuchen- oder Balkendiagramm:

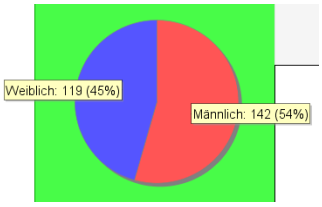


Abbildung 8: Geschlecht

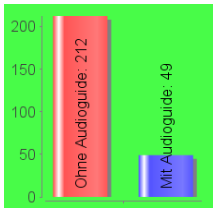


Abbildung 9: Audioguide

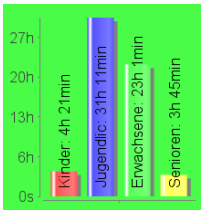


Abbildung 10: Alter

- Filterung der Auswertungen nach Zeit, demographischen Eigenschaften oder Gruppen:

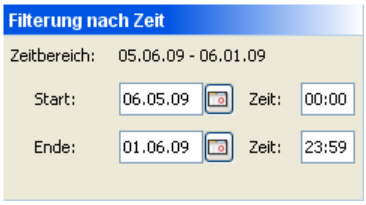


Abbildung 11: Dockable der Zeiteinstellungen

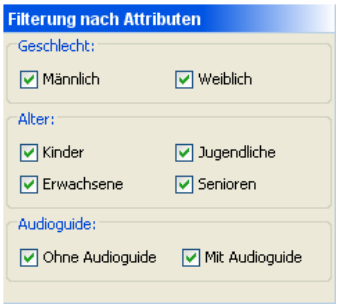


Abbildung 12: Dockable der Attributtypen

Abbildung 13: Dockable der Gruppen

- Auswertungen über ein gesamtes Museum pro Tag (inkl. Ansicht demographischer Daten):

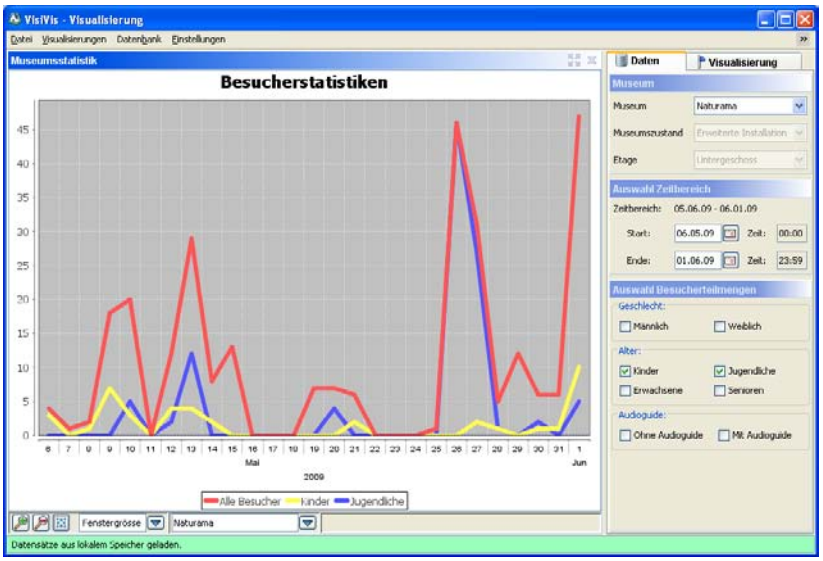


Abbildung 14: Bildschirmfoto von dem Museumsstatistik

7.4 Ausblick

Mit dieser Arbeit wurde ein solider Grundstein gelegt, auf dem aufgebaut werden kann. Die erarbeitete Software wird in einer Nachfolgearbeit in einem Feldtest produktiv getestet. Zusätzlich wird die Software verbessert und um weitere Visualisierungen bereichert, welche weitere Auswertungen der gesammelten Daten ermöglichen.

8 Projekt Management

8.1 Projektorganisation

8.1.1 Beteiligte Personen

		
Name	Patrick Oswald	Simon Inderbitzin
Verantwortung Rolle	Softwareentwickler	Softwareentwickler
Primärer Fokus	Datenverarbeitung für die Statistiken	Visualisierung der Statistiken

8.1.2 Alle Beteiligten

Name	Verantwortung / Rolle
Prof. Dr. Lothar Müller	Betreuer, Productowner
Thomas Kälin	Projektleiter, Museumseditor, Scrum-Master
Michael Rüegg	Assistent IFS, Kassenapplikation, Hardwareanbindung
Stefan Züger	Visualisierungen: Wegverfolgung: Softwareentwickler
Tobias Zürcher	Visualisierungen: Wegverfolgung: Softwareentwickler

8.1.3 Auftraggeber

Unsere Industriepartner sind Albis Technologies Zürich und das Naturama in Aarau. Diese Studienarbeit wurde an der Hochschule für Technik Rapperswil unter der Betreuung von Prof. Dr. Lothar Müller realisiert. Zusätzlich übernahm Herr Müller die Rolle des Productowners (ersichtlich im Unterkapitel Scrum).

8.1.4 Organisation

Primäre organisatorische Abwicklungen wurden von Thomas Kälin (Scrum-Master) geführt. Er hat im Rahmen der gesamten Applikation die Projektleitung für das insgesamt sechsköpfige Team übernommen. Die Arbeitspakete (Userstories) wurden vorgängig mit dem Betreuer Prof. Dr. Lothar Müller besprochen und anschliessend abgearbeitet. Zur Kontrolle und Verbesserung der ausgeführten Arbeiten erhielt der Betreuer regelmässig Auszüge aus diesem Bericht. Das Team legte auch kontinuierlich lauffähige Prototypen der Software vor.

8.1.5 Besprechung

8.1.5.1 Regelmässige Sitzungen mit dem Betreuer/Productowner

Die regelmässigen Sitzungen mit dem Betreuer wurden an keinem fixen Zeitpunkt festgelegt, fanden aber in der Regel jeweils wöchentlich dienstags oder mittwochs statt. Für kleinere Anfragen und Problembesprechungen konnten wir jederzeit mit Herrn Müller ein Kurzmeeting einlegen.

8.1.5.2 Regelmässige Sitzung für pro Sprint

Am Ende eines zweiwöchigen Sprints wurde ein Sprintmeeting gehalten. Bei diesen Sitzungen aktualisierte das Team die Arbeitspakete (Sprint Backlog) und führte eine Sprintretrospektive durch (detaillierte Erklärungen dazu sind im Kapitel Scrum einzusehen). Diese Treffen stellten den gemeinsamen Kontext dar, an dem das gesamte Team beteiligt war.

8.1.5.3 Sitzungsprotokolle

Für die oben genannten Sitzungen wurden Sitzungsprotokolle der behandelten Themen geführt. Diese Protokolle sind im Sitzungsprotokoll-Ordner[1] für spätere Einsicht abgelegt worden.

8.1.5.4 Dailymeetings

Das Prozessmodell Scrum sah für jeden Entwicklungstag ein Dailymeeting vor. Diese (Standup-) Meetings wurden stehend geführt, damit sich die beteiligten Personen möglichst kurz hielten und nicht zu lange diskutierten.

Diese Meetings wurden an folgenden Daten gehalten:

- Dienstag 09:00-09:15
- Mittwoch 13:00-13:15
- Donnerstag 13:00-13:15.

Während diesen kurzen aber intensiven Treffen wurde erklärt, was das jeweilige Entwicklerteam erledigt hat, wo Probleme aufgetaucht sind und welche Arbeiten noch bevor stehen. Am Ende des Meetings wurde der Burndown-Chart aktualisiert, damit alle Beteiligten und insbesondere der Productowner auf einen Blick einsehen konnte, wie der Fortschritt des aktuellen Sprints aussieht.

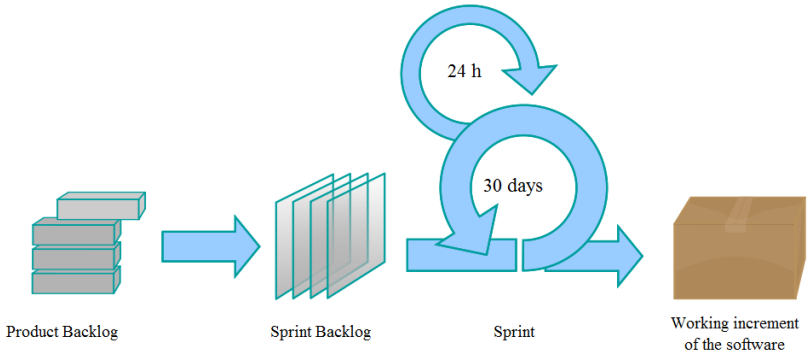


Abbildung 15: Iterativer Ablauf von Scrum[2]

[1] CD-ROM: \4. Projekt Management\Sitzungsprotokolle\...
[2] http://upload.wikimedia.org/wikipedia/commons/5/58/Scrum_process.svgf, letzter Zugriff 01.12.09

8.2 Projektplanung

8.2.1 Zeitfenster

Die Studienarbeit war im Zeitraum der Ausschreibung 18.05.09 bis zur Abgabe am 18.12.09 definiert. Der offizielle Start wurde mit 14.09.09 datiert. Das heisst es konnte während einer Zeitspanne von schätzungsweise drei Monaten gearbeitet werden.

8.2.2 Arbeitszeit

Die Studienarbeit für eine Person wird an der HSR mit 8 ECTS-Punkten berechnet, was einem Gesamtaufwand von ca. 240 Stunden entspricht. Dies entspricht pro Woche etwa 15-20 Stunden.

8.2.3 Arbeitspakete

Nachfolgend ist ein Auszug aus dem letzten ProductBacklog[3] von unserem Team ersichtlich. Die Arbeitspakete wurden in die Etagenvisualisierung (Statistiken – lokal) und die Museumsvisualisierung (Statistiken – global) eingeteilt. Die Wichtigkeit wurde vom Productowner festgelegt und während der Sprintmeetings aktualisiert. Die Einheit SP steht für einen „StoryPoint“, welcher mit etwa 8 Stunden gerechnet wird.

[3] Product Backlog.xls, Thomas Kälin 2009

ID	Anwendung	User Story	Wichtigkeit	Umfang [SP]	Sprint	Status
3	Statistiken - lokal	Personen pro Reader anzeigen	70	3	1	Erledigt
4	Statistiken - lokal	Personen: Farben anzeigen	70	1	1	Erledigt
5	Statistiken - lokal	Personen: Zahlen anzeigen absolut	69	2	1	Erledigt
2	Statistiken - lokal	Personen pro Raum anzeigen	68	1	1	Erledigt
1	Statistiken - lokal	Auswahl des Zeitbereiches und Etage (Basic, Beachtung von Zuständen)	67	4	2 / 3	Erledigt
7	Statistiken - lokal	Aufenthaltsdauer pro Reader anzeigen	65	1	2	Erledigt
8	Statistiken - lokal	Aufenthaltsdauer: Farbe anzeigen	65	1	2	Erledigt
9	Statistiken - lokal	Aufenthaltsdauer: Zahlen anzeigen (Maximum, Minimum und Durchschnitt)	64	1	2	Erledigt
6	Statistiken - lokal	Aufenthaltsdauer pro Raum anzeigen	63	1	2	Erledigt
10	Statistiken - lokal	Festlegen von Filtern (Geschlecht, Alter, Audioguide, andere Kriterien? Gruppen?)	62	3	3	Erledigt
12	Statistiken - lokal	Personen: Umschaltung absolut / relativ - Farbskala automatisch anpassen (falls nötig)	61	0.5	3	Erledigt
13	Statistiken - lokal	Personen: Zahlen anzeigen relativ	61	0.5	3	Erledigt
11	Statistiken - lokal	Farbskalen einstellbar global für ganze Anwendung	60	5	4	Erledigt
14	Statistiken - lokal	Farbskalen bleiben über Programmende hinaus erhalten.	58	0	4	Erledigt
55	Statistiken - lokal	Auswahl des Zeitbereiches und Etage (Extended)	57	2	6	Analyse erledigt.
15	Statistiken - lokal	Personen: Verteilungen (Geschlecht, Alter, anderes) --> Diagramme	49	Nicht geschätzt	6	Erledigt
16	Statistiken - lokal	Aufenthaltsdauer: Verteilung (Geschlecht, Alter, anderes) --> Diagramme	48	Nicht geschätzt	6	Erledigt
17	Statistiken - lokal	Darstellung von zeitlicher Entwicklungen (z.B. Besucherzahlen) pro Raum / Reader	20			
19	Statistiken - global	Besucher pro Tag anzeigen (Kurve)	70	4	5	Erledigt
18	Statistiken - global	Auswahl des Zeitbereiches (keine Unterscheidung nach Zuständen)	69	1	5	Erledigt
20	Statistiken - global	Festlegen von "Filtern" (Geschlecht, Alter, Audioguide --> weitere Kurven hinzufügen)	65	2	6	Erledigt
21	Statistiken - global	Tabellarische Übersicht (Inhalt? Evt. wichtigste Kennzahlen? Vom Benutzer definiert?)	20			
22	Statistiken - global	Prognosefunktion für Besucheraufkommen basierend auf historischen Daten	10			

8.3 Prozessmodell

8.3.1 Scrum

Für die Projektorganisation orientierten wir uns an Scrum. Dieser Begriff steht (im englischen) für „Gedränge“ und ist ein Vorgehensmodell, das mit Meetings, Artefakten, Rollen, Werten und Grundüberzeugungen beim Entwickeln von Produkten agiler Softwareentwicklung verwendet wird. Der Grundgedanke der von Scrum ist, dass während der Entwicklungszeit möglichst agil zu arbeiten.

Hinweis: Beispielhafter Ablauf kann in den Reviewdokumenten eingesehen werden, dort wurden einige Traktanden aus einem solchen Meeting aufgeführt.

Der Aufbau und welche Artefakte im Scrum existieren, zeigen folgende Beschreibungen aus Wikipedia:

8.3.1.1 *Product Backlog*

„Das Product Backlog enthält die Features des zu entwickelnden Produkts. Es umfasst alle Funktionen, die der Kunde wünscht, zuzüglich technischer Abhängigkeiten. Vor jedem Sprint werden die Elemente des Product Backlogs neu bewertet und priorisiert, dabei können bestehende Elemente entfernt sowie neue hinzugefügt werden. Hoch priorisierte Features werden von den Entwicklern im Aufwand geschätzt und in den Sprint Backlog übernommen. Ein wesentliches Merkmal des Backlogs ist die Tiefe der Beschreibung von einzelnen Features. Hoch priorisierte Features werden im Gegensatz zu niedrig priorisierten sehr detailliert beschrieben. Somit wird viel Zeit für die wesentlichen Elemente und wenig für unwesentliche verwendet. .“ [4]

8.3.1.2 *Sprint Backlog*

„Das Sprint Backlog enthält alle Aufgaben, die notwendig sind, um das Ziel des Sprints zu erfüllen. Eine Aufgabe sollte dabei nicht länger als 16 Stunden dauern. Längere Aufgaben sollten in kurze Teilaufgaben zerlegt werden. Bei der Planung des Sprints werden nur so viele Aufgaben eingeplant, wie das Team an Kapazität aufweisen kann.“ [5]

[4] http://de.wikipedia.org/wiki/Scrum, *letzter Zugriff 16.12.09*
[5] http://de.wikipedia.org/wiki/Scrum, *letzter Zugriff 16.12.09*

8.3.1.3 *Burndown Chart*

„Das Burndown Chart ist eine graphische, pro Tag zu erfassende Darstellung des noch zu erbringenden Restaufwands pro Sprint. Im Idealfall fällt die Kurve kontinuierlich (daher Burndown) und der Restaufwand ist am Ende des Sprints Null. Das Chart lässt anhand der Verlängerung der negativen Steigung bereits während des Sprints erkennen, ob der anfangs geschätzte Aufwand umgesetzt werden kann.“ [6]

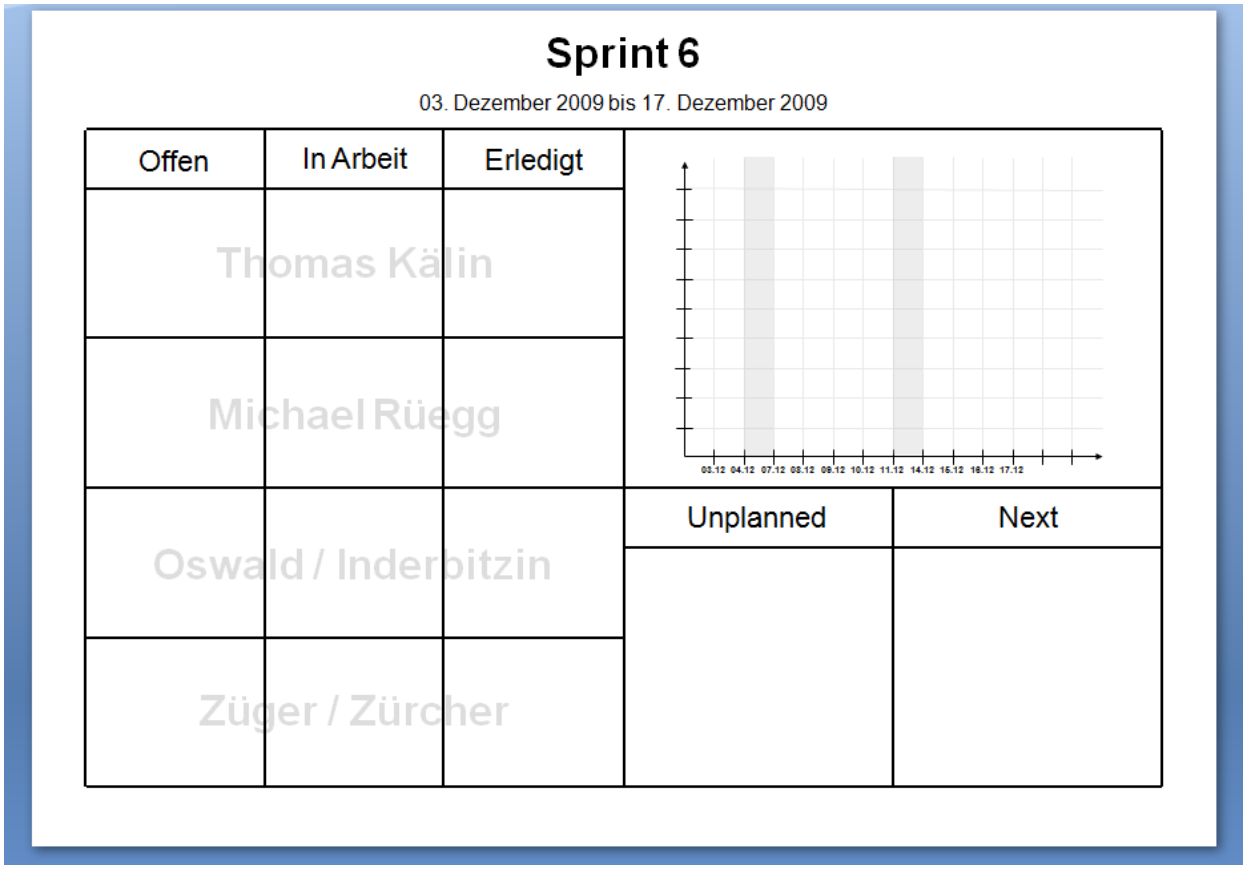


Abbildung 16: Burndown-Chart[7]

Unsere Burndownchart (Abbildung 16: Burndown-Chart) wurde auf A1 ausgedruckt. Während den Standupmeetings klebte man die einzelnen Tasks als PostIt-Zettel in die entsprechende Spalte der Chart.

[6] *http://de.wikipedia.org/wiki/Scrum, letzter Zugriff 16.12.09*
[7] http://www.xqual.com/resources/images/scrum_burndown_chart.gif, letzter Zugriff 10.11.2009

8.3.2 Extreme Programming

Im Team wurde nicht nur Scrum, sondern auch vermehrt XP-Praktiken angewendet. Extrem Programmierung - kurz XP - ist ein Vorgehen, welches das Lösen einer Programmieraufgabe und nicht dessen formale Vorgehen in den Vordergrund stellt. Diese Methode ermöglicht den Entwicklern eine iterative Annäherung an die gestellten Anforderungen des Productowners.

Typisch für XP Praktiken sind beispielsweise, dass oft Gewichtungen, Prioritäten und Zeitschätzungen angepasst werden. Das hat zur Folge, dass der Productowner am Ende des Projekts das erhält, was er bereits während des Prozesses einsehen konnte. Zusätzlich soll bei Möglichkeit (situationsbedingt) Pair-Programming betrieben werden. Dieses Vorgehen schafft Aspekte, die einem Einzelnen entfallen wären. Damit wird der Code verbessert und schafft eine gemeinsame Basis, mit welcher beide Entwickler einverstanden sind.

Abbildung 17: Schematischer Zyklus des XP[8]

8.4 Zeitplanung

Der Zeitaufwand einzelner Sprints und Tasks wurden im Plenum während der Scrum-Meetings festgelegt.

8.4.1 Zeiterfassung mit Paymo

Für unsere eigene Zeiterfassung verwenden wir ausschliesslich Paymo[9]. Diese Applikation erlaubt eine off und online Verwaltung der investierten Zeit. Im Tool selber können einzelne Projekte, Task und Userstories festgelegt werden. Während der Arbeit kann so ein Entwickler seine Zeit an ein solches Kriterium verbuchen. Sobald ein Sprint mit den Userstories definiert wurde, vermerkten wir die Arbeitspakete in Paymo. Ab diesen Zeitpunkt konnten minutengenaue Zeitabrechnungen erstellt werden. Auf diese Art wurde die Zeitabrechnung detailliert erfasst und es konnten durch Paymo Zeitauswertungen erstellt werden.

8.5 Arbeitsumgebung

8.6 Entwicklungsumgebung

8.6.1 SVN

Bei unserer Studienarbeit wurde Subversion (SVN) eingerichtet und rege benutzt. SVN ist eine freie Software zur Versionsverwaltung von Dateien und Verzeichnissen.

[8] http://enterpriseblog.net/wp-content/uploads/2009/03/xp1.png, letzter Zugriff 10.12.09
[9] https://studienarbeit.paymo.biz/dashboard, letzter Zugriff 17.12.09

8.6.2 Eclipse

Eclipse ermöglicht eine Programmierumgebung, die zur Entwicklung von Software verschiedenster Art dient. Da wir ausschliesslich Java programmieren, zeigt sich Eclipse als besten gemeinsamen Nenner aller Teammitgliedern.

8.6.3 Hudson

Vom Projektleiter wurde ein Hudsondienst eingerichtet. Dieser ermöglicht die Überwachung des Buildprozesses, Testabdeckung, Checkstyle und Codemetriken.

8.6.4 Trac

Der Projektleiter richtete den Trac ein, welches eine Wiki-Oberfläche für allgemeine Dokumentationen und ein Ticketsystem darbietet.

8.7 Qualitätsmassnahmen

8.7.1 Coderichtlinien

Die umgesetzten Coderichtlinien sind im Richtlinien-Dokument[10] einzusehen.

8.7.2 Tests

Jeder Sprint hat ähnliche, wenn nicht sogar gleiche, Qualitätsmassnahmen. Da wir diese nicht jedes Mal erneut erwähnen wollen, ist hier eine kurze Auflistung unserer Qualitätsmassnahmen ersichtlich:

- UnitTests über ProblemDomain und interner Strukturen (wie z.B. das Abzählen der Besucher)
- Gui-Besprechungen / -Tests mit externen Personen (unter anderem Productowner)
- Code-Review im Team, d.h. nach grossen Codeveränderungen, setzten wir uns gezielt für 15-20 Minuten an dem Codestück und besprechen Verbesserungen.
- XP-Praktiken (beispielsweise das Pair-Programming)
- Designs werden mindestens ein Mal wöchentlich vom Productowner und anderen Teammitgliedern eingesehen.

8.7.3 Usability-Tests

Nach jedem Sprintmeeting prüfte der Productowner (Prof. Dr. Müller), die ihm vorgelegte aktuellste Version. Während dieses Usability-Tests kamen einige Fragen, Anregungen oder Probleme auf, die an die Entwickler weiter kommuniziert wurden. Für jedes dieser Anfragen wurde ein Ticket erstellt. Jegliche Tickets sind im Kapitel 34 ersichtlich.

8.8 Dokument Management

Schon in früheren Projektarbeiten, zeigte sich die Handhabung der Dokumentation als aufwändig. Denn oft wurden Dokumente nicht in einem Stück erstellt, sondern wuchsen parallel zum erarbeiteten Quellcode. Der Dokumentaustausch durch USB-Sticks oder im SVN-Repository zeigte sich als unzureichend. Anfänglich realisierten wir den Austausch von Gedanken und Dokumente über eine eigene Wikipage. Im Verlaufe der Studienarbeit erwies sich diese Methode jedoch als ungeeignet und wurde durch Dropbox ersetzt. (Begründungen sind im Kapitel Erfahrungsbericht ersichtlich)

[10] CD-ROM: „\8. Ressourcen\Entwicklungsumgebung \CodeConventions.pdf“

8.8.1 Dropbox

Dropbox[11] ist eine Applikation welche die Synchronisation von Daten zwischen unterschiedlichen Rechner ermöglicht. Nach einer kurzen Registrierung und Installation des Clients, kann der Benutzer auf einfache Art und Weise beliebige Windows-Ordner über das Internet teilen. Sobald sich ein Dokument oder eine Datei im geteilten Ordner ändert, wird diese bei allen angemeldeten Benutzern aktualisiert. Zusätzlich unterstützt Dropbox eine Versionshistorie, damit auf frühere Versionen oder gelöschte Dateien zugegriffen werden kann.

8.8.2 Eclipse Modeling Framework (OMONDO)

Für die Darstellungen der Klassen und Methoden in UML wurde OMONDO[12] verwendet. Dieses Framework erstellt ein Modell aus dem Code und generiert daraus Grafiken. Änderungen Code lässt werden in Grafiken automatisch aktualisiert.

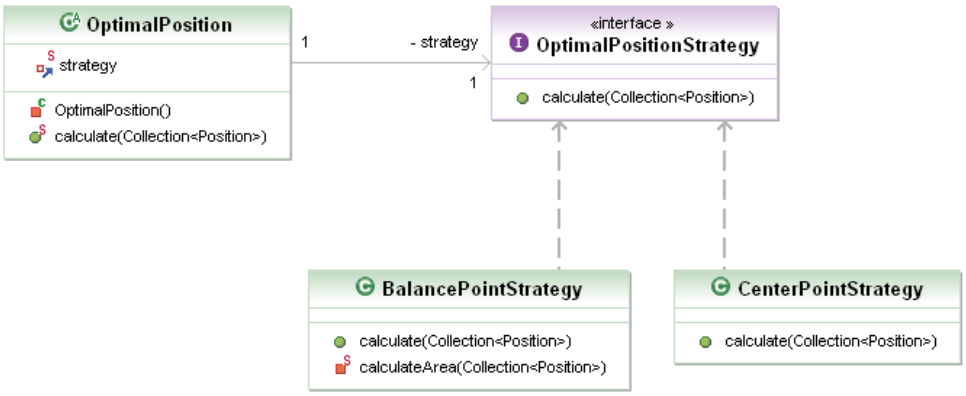


Abbildung 18: Omondo Klassendiagramm

[11] https://www.dropbox.com/, letzter Zugriff 17.12.09
[12] http://www.uml2.org/, letzter Zugriff 17.12.09

9 Auswertung der Arbeitszeiten

Bei den nachfolgenden Graphen sind die gesamten Arbeitszeiten unter Berücksichtigung unterschiedlicher Gesichtspunkte ausgewertet worden. Bei den Teilabschnitten mit Schätzungen, wurde zusätzlich eine Gegenüberstellung von Soll/Ist vorgenommen.

9.1 Total

Als Erstes werden die erfassten Zeiten in Zahlen vorgestellt. In der folgenden Tabelle wurden auf der linken Seite die Scrum-Phasen aufgeführt. In der Kopfzeile befinden sich die Auswertungen nach sinnvollen Kriterien. Mit „nicht geschätzt“ sind die Zeiten gemeint, zu welchen keine Schätzungen existieren. Dies betrifft vor allem die Meetings und Arbeitsschritte, welche keine Zuordnung bezüglich Userstories haben. Alle dargestellten Werte beziehen sich auf den Zeitraum vom 14.09.2009 bis 17.09.2009.

9.1.1.1 Auswertung

Phase	nicht geschätzt	Soll (laut Schätzung)	Ist	Analyse	Implementation	Dokumentation	Patrick	Simon	Total
Vorbereitung	32	0	0	0	0	14	15	17	**32**
Sprint 1	23	57	55	16	23	8	44	35	**79**
Sprint 2	9	44	68	21	31	16	31	45	**76**
Sprint 3	29	56	80	8	55	17	61	53	**114**
Sprint 4	9	40	62	12	38	12	27	44	**71**
Sprint 5	7	56	64	25	31	8	29	40	**69**
Sprint 6	51	16	18	12	29	19	44	25	**69**
Endbericht	80	0	0	0	0	80	18	18	**36**
Total	**240**	**269**	**347**	**94**	**207**	**160**	**269**	**277**	**546**

9.1.1.2 Analyse

Die Soll- und Ist-Zeiten zeigen eine Abweichung von 30%. Erklären lässt sich dies durch die zu optimistischen Schätzungen im Verlauf der Arbeit. Gegen Ende der Studienarbeit konnte dieses Verhältnis besser ausbalanciert werden. Beim Betrachten der einzelnen Disziplinen ist ersichtlich, dass der Aufwand für das Erstellen der Dokumente einen wesentlichen Teil eingenommen hat.

Die Arbeitszeitverteilung innerhalb unseres Teams ist nahezu identisch ausgefallen. Der aufgewendete Zeitaufwand von knapp 270 Stunden liegt dabei über dem erwarteten Zeitaufwand von 8 ECTS-Punkten (entspricht ungefähr 240 Stunden pro Person). Das entspricht einer Abweichung von 12% pro Teammitglied gegenüber der geforderten Arbeitszeit. Der Grund für diese Differenz ist der optimistischen Zeitschätzung und dem persönlichen Ehrgeiz der Entwickler zuzuschreiben.

9.2 Pro Phase

In den nachfolgenden Graphen wurden die Arbeitszeiten pro Sprint ausgewertet. Zusätzlich führt der Graph die Vorbereitungen vor Sprint 1 und die Zeiten für den Endbericht auf. Bei der Zeiterfassung ist zu erwähnen, dass die Zeiten immer den entsprechenden Arbeitspaketen angerechnet wurden, welche pro Sprint festgelegt wurden, auch wenn gewisse Arbeiten eine Woche später erledigt wurden. Dadurch ist der reale Zeitaufwand pro Sprint ersichtlich.

9.2.1 Diagramm

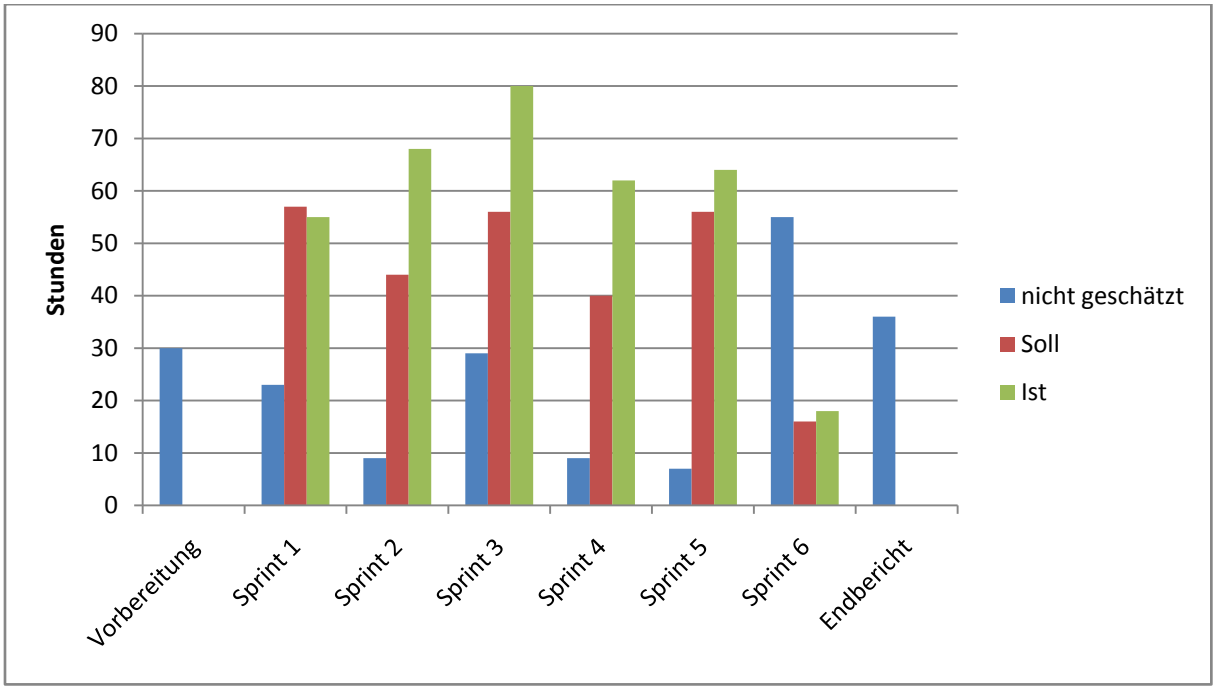


Abbildung 19: Aufwand pro Sprint

9.2.1.1 Analyse

Auf obigem Diagramm kann beobachtet werden, dass die geschätzten Zeiten für die Sprints 2, 3 und 4 sehr optimistisch ausgefallen waren. Ab Sprint 5 wurde versucht die Schätzungen etwas realistischer zu treffen, was sich auch bewährt hat. Sprint 6 hat einen hohen Anteil an „nicht geschätzter" Zeit, weil in diesem Sprint eine Zusatzleistung erbracht wurde. Es handelt sich dabei um die 2 Usterstories „Personen: Verteilung (Geschlecht, Alter, anderes) -> Diagramm" und „Aufenthaltsdauer: Verteilung (Geschlecht, Alter, anderes) -> Diagramm". Für die Vorbereitungen und den Endbericht wurden keine Schätzungen getroffen, da diese kaum abzuwägen waren.

9.3 Pro Arbeitspaket

Nachfolgend sind die Soll-/Ist-Zeiten pro Arbeitspaket (Userstories) aufgeführt. Für Arbeitspakete mit einem sehr geringen Zeitaufwand wurden keine Balken erstellt. Zu jedem Paket ist die entsprechende ID im Produktbacklog aufgeführt. Bei den letzten beiden (Zusatz-) Userstories bestehen keine Sollzeiten, da diese keine Schätzung erhielten.

9.3.1 Diagramm

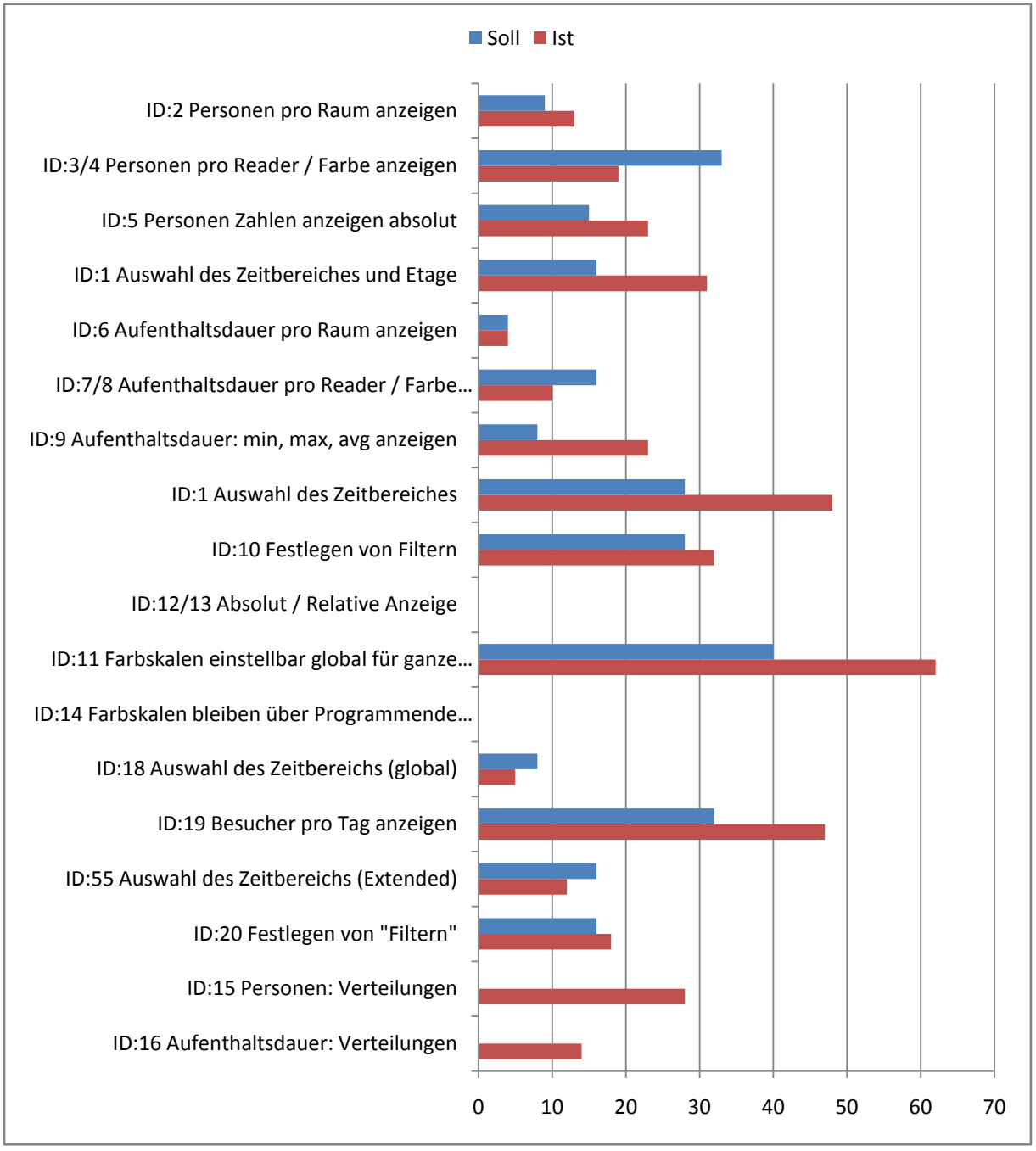


Abbildung 20: Aufwand pro Arbeitspaket

9.3.1.1 *Analyse*

Die Soll-Zeiten weichen ungefähr um 30% von dem Ist ab. Grund dafür lässt sich in den zu enggeplanten Schätzungen erkennen. In Zukunft wird versucht, ausgehend von diesem Ergebnis bessere Schätzungen zu führen.

9.4 Pro Person

Diese Auswertung zeigt die erfassten Zeiten pro Person mit der Aufteilung in einzelne Phasen.

9.4.1.1 Diagramm

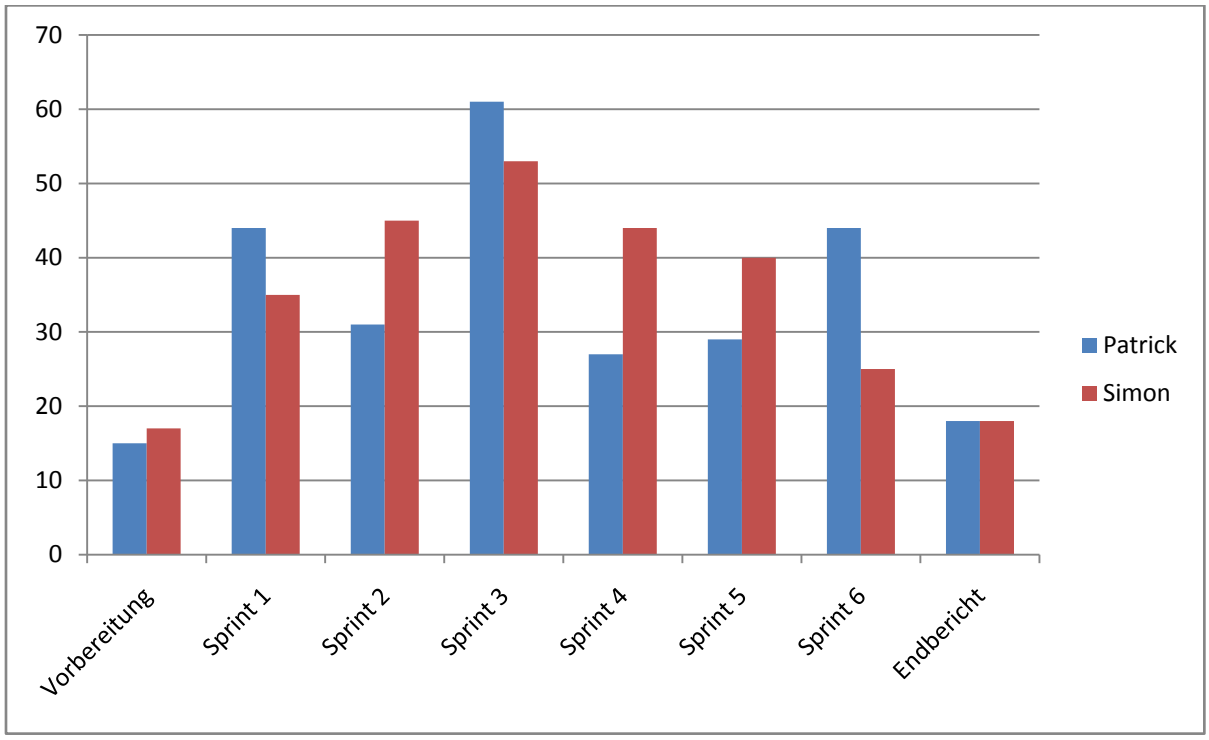


Abbildung 21: Aufwand der einzelnen Teammitglieder

9.4.1.2 Analyse

Auf den ersten Blick ist ersichtlich, dass in den Sprints unterschiedlich intensiv gearbeitet wurde. Durch Aufsummierung lässt sich erkennen, dass die Arbeitsaufteilung erstaunlich fair blieb: Simon verbuchte einen Zeitaufwand von 50.7% und Patrick 49.3%. Dies ist ein erfreuliches Ergebnis, da es zeigt, dass sich beide Beteiligten gut ergänzt haben.

9.5 Pro Disziplin

Infolgedessen, dass die ganze Studienarbeit auf vorgängigen Analysen und Prototypen basierte, konnten die üblichen Disziplinen auf folgende drei Primäraspekte reduziert werden: Analyse, Implementation und Dokumentation. Als erstes wird die Aufteilung in Phasen dargestellt. Anschliessend wird das Verhältnis der Disziplinen als Kuchendiagramm veranschaulicht.

9.5.1 Diagramm pro Sprint

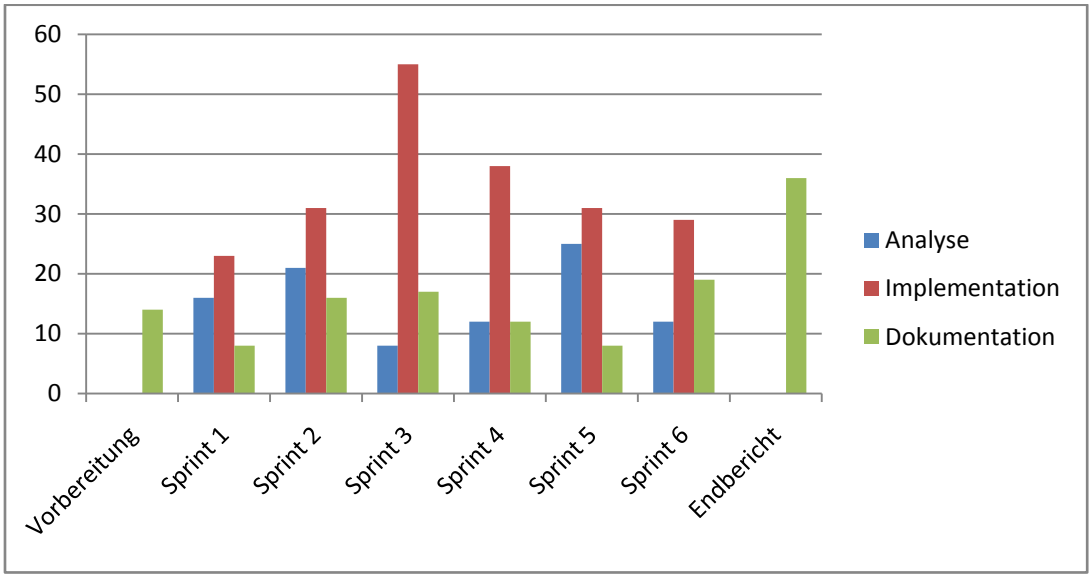


Abbildung 22: Aufwand pro Sprint

9.5.1.1 Analyse pro Sprint

In der Analyse pro Sprint ist zu erkennen, dass nicht immer zu gleichen Teilen an den Disziplinen gearbeitet wurde. Beispielsweise zeigte sich Sprint 3 (Möglichkeit zur Filterung) und Sprint 4 (Farbskalen mit einer Scrollbar) sehr implementationslastig, weil dort einige grundlegende Features realisiert wurden. Wie erwartet, wurde während der Vorbereitungsphase und dem Erstellen des Endberichts nur dokumentiert.

9.5.1.1.1 Diagramm Aufteilung

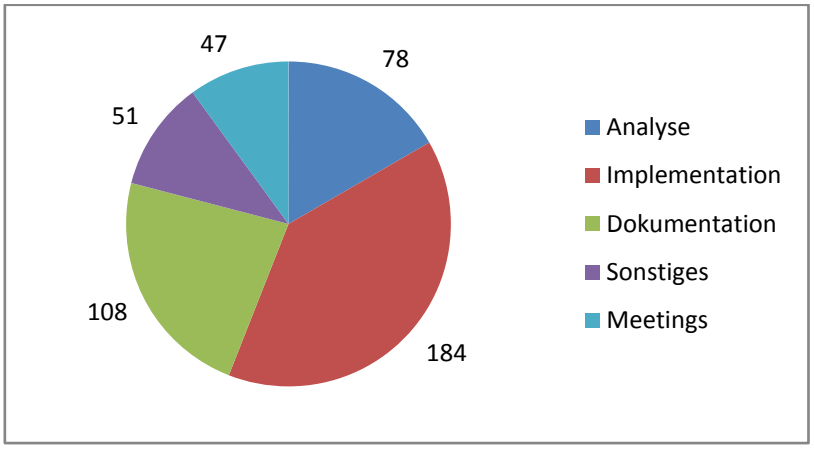


Abbildung 23: Zeitliche Aufteilung der Disziplinen

9.5.1.1.2 Analyse Aufteilung

Wie vermutet wies sich die Implementierung am Zeitaufwändigsten. Entgegen unserer Prognose nahm die Dokumentation wesentlich mehr Zeit in Anspruch als erwartet. Grund dafür war, dass gute Designanalysen und Beschreibungen aufwändig zu dokumentieren waren. Der zeitliche Aufwand der Meetings und von Sonstigem deckten sich mit unseren Vermutungen.

10 Risiko Management

Die folgende Risikobewertung hat keinerlei Anspruch auf Vollständigkeit, Ziel ist die Bewertung soll eine grobe Risikoeinschätzung erlauben. Grundlage der Berechnungen ist die Annahme, dass 100 CHF etwa 1 Mannarbeitsstunde entsprachen. Das heisst konkret, dass die Entwickler mit 100CHF/h entlöhnt werden. Folglich bedeutet dies, der monetäre Gesamtaufwand unserer Studienarbeit ergibt: 100CHF * 300*2 Stunden = 60'000 CHF

Risiko Bewertungen								
Risk ID	Risiko	Auswirkung	Massnahme	Mass-nahmenkosten in CHF	Max. Schaden in CHF	Wahrschein-lichkeit des Eintreffens	Gewichteter Schaden in CHF	Priorität
R01	Keine Kunden für dieses Projekt	Die Kosten werden kaum oder nicht gedeckt	Marktanalyse & gute Software	2'000	20'000	2%	400	Tief
R02	Personen erkranken (z.B. Schweinegrippe)	Wissen über Projekt und fehlender Arbeiter	Wiki um wissen zu speichern. Gegebenfalls gesunde Ernährung, Programmierpausen und Sport.	1'000	2'000	10%	200	Tief
R03	Falsche Terminplanung	Verschiebung und Verzögerung	Gute Planung und Puffer von Zeit einberechnen	1'000	5'000	30%	1'500	Mittel
R04	Zu komplexes GUI für den Anwender, da viele Einstellungen möglich	Software wirkt unattraktiv	Prototypen bauen und immer wieder auswerten	8'000	15'000	20%	3'000	Hoch
R05	Zu wenig Kommunikation innerhalb des Teams	Fehlender Gedankenaustausch, Abkapselung	Wöchentliche oder sogar tägliche Sitzungen	8'000	2'000	5%	100	Tief
R06	Kompletter Rechnerabsturtz	Verlust des Codes	SVN, d.h. Subversionierung von Code	1'000	25'000	1%	250	Tief
R07	Verbuggte Software	Systemcrash oder unbekanntes Fehlverhalten	BugTracking und Tickets ermöglichen	5'000	15'000	15%	2'250	Hoch
	Totale Kosten			26'000				
	Total Rückstellungen						7'650	

10.1 Auswertung der Risikoanalyse

Der anfängliche Gesamtaufwand beträgt ca. 60'000 CHF. Mit einer totalen Rückstellung von 7'700 CHF beträgt das Risiko ca. 14%. Dies scheint in der Informatikbranche ein durchaus tolerierbarer Wert. Das Risiko entspricht einem Arbeitsrisiko von 77 Stunden, was zwei ganzen Entwicklungswochen entsprechen würde.

10.2 Detaillierte Risikoanalyse

Eine ausführliche Analyse und Aufzählungen von Risiken sind im Dokument Risikomanagement[13] ersichtlich.

10.3 Die 3 grössten Risiken

Aufgrund der vorherigen analysierten Risiken kann eine Rangliste der drei grössten Risiken aufgestellt werden:

Rang	Risikobeschreibung	Konkrete Qualitäts- und Gegenmassnahmen
1.	Komplexes GUI	• Ständige Verbesserung der Software durch Tests. • Feedback von PO einholen und umsetzten. (mind. 1mal pro Woche) • Möglichst optimierte Software entwickeln, das heisst der Benutzer soll auf einfache Art und Weise komplexe Einstellungen vornehmen können. • Genügend Zeit in ausführliche Designanalysen investieren.
2.	Bugs und Fehler in der Software	• Bugtracking nutzen. • Externe Eclipse-Plugins wie Bugtracker und Checkstyle anwenden. • „Codesmells“ suchen und bereinigen. • XP-Praktiken anwenden, besonderen Schwerpunkt auf Pairprogramming legen. • Code gegenlesen. • Tickets erstellen, besonders bei teamübergreifenden Problemen. • Bugs öffentlich und schnell kommunizieren. • Fehlermeldungen von Hudson möglichst vermeiden. • Nur das Einchecken, das auch präsentationsfähig ist, d.h. der Build könnte jederzeit an Kunden ausgeliefert werden. • Unittests verwenden
3.	Terminplanung	• Vorsichtige Planung, beim Scrummeetings. • Pessimistische Schätzungen abgeben, d.h. tendenziell mehr Zeiteinplanen als vermutet wird. • Mindestens einen halben Entwicklertag (8h) als Pufferzone einführen.

[13] CD-ROM: \4. Projekt Management\Risiken\Risikomanagement.xls

10.4 Resümee der Risiken

Rückblickend erwies sich die Zusammenarbeit innerhalb und ausserhalb unseres Teams als hervorragend. Scrum erlaubte ein freies aber doch strukturiertes Vorgehen. Dadurch konnten wir jederzeit die Zeitplanung auf neu erschienene Risiken reagieren und adaptieren. Durch dieses Vorgehen reduzierten sich die Risiken auf ein Minimum und wurden vortlaufend erfasst und bearbeitet. Folgendermassen entstanden keine grossen Überraschungen während und besonders am Ende unserer Studienarbeit. Jeder Bug wurde getrackt, gezielt erfasst und verarbeitet. Die Fehler, die noch nicht abgearbeitet werden konnten wurden aufgeführt und sind somit für weitere Projekte klar dokumentiert. Innerhalb der Arbeit erreichte der Unit-TestsCase eine Abdeckung von mehr als 95% des Codes. Dies stellte jederzeit eine sichere Umgebung dar.

Die beschriebenen drei Toprisiken wurden – in der Praxis - nie wirklich als Bedrohung wahrgenommen. Grund dafür war, dass sich die Vorkehrungen als stabil und ausreichend herausstellten.

11 Ergebnisse

11.1 Erreichte Ziele

Während der Studienarbeit wurden alle vorgegebenen Ziele erreicht, darüber hinaus realisierten wir noch zusätzliche Tasks (eine Gesamtübersicht ist im Kapitel 8.2.3 vorzufinden).

11.2 Offene / abgeschlossene Probleme

Nachfolgend ist ein zusammengestellter Ausschnitt aus unserer Todo-Liste und dem TicketTracking ersichtlich. Die „Open-Issues“ sind auch aufgeführt, werden aber noch am Ende dieses Kapitels ausführlicher behandelt.

Nr.	Sprint	Was muss gemacht werden (ToDo)	Priorität (1=min, 10=max)	Status
1	-	**Visualisierung - Refactoring der Klasse "WindowSettings "** Die Klasse "WindowSettings“ der Visualisierungsanwendung hat sich im Verlaufe der Entwicklungsarbeit zum "GarbageCollector“ entwickelt. Ein Refactoring wäre sehr zu empfehlen. Dies ist aufgrund der aktuellen Komplexität der Klasse ziemlich aufwendig.	9	open
2	2	**Visualisierung - Umgestaltung des "Besucherattribute"-Dockables** Das "Besucherattribute"-Dockables (Statistiken) benötigt im Moment unnötig viel Platz. Eine verbesserte Gestaltung des Dockables (z.B. zweispaltig oder eine Tabellenlösung) wäre aus platzgründen zu bevorzugen.	6	Fixed
3	2	**Einmittung von Strings** Beim Visualisieren von Strings, ist meistens der korrekte Darstellungs-Mittelpunkt gegeben. Daher muss anhand der Stringlänge korrekt gemittet werden (so in etwa "-StringLänge_inPixel/2").	3	Fixed
4	2	**Besucher pro Raum / Reader – Min/MaxWerte** In der aktuellen Lösung werden die Reader / Räume in einem Farbton zwischen Minimum und Maximum eingefärbt. Als "Grenzwerte" ist momentan folgendes eingestellt: Minimum = 0 Besucher Maximum = Total der erfassten Besucher Damit die Reader besser unterscheidbar werden wäre vielleicht folgendes sinnvoller:	6	Fixed

		Minimum = Anzahl der Besucher beim Reader / Raum mit den wenigsten Erfassungen Maximum = Anzahl der Besucher beim Reader / Raum mit den meisten Erfassungen So sollte die Abstufung sehr viel besser unterscheidbar werden.		
5	3	**Darstellung der Visualisierungsauswahl** • Die Auswahl der Visualisierungen ist nicht selbst erklärend. Evt. Beschriftung nötig. • Gruppierung der zusammengehörigen Buttons • Veränderung der Button-Reihenfolge: eigentlich wählt der Benutzer ja zuerst, ob er die Besucherzahlen oder Aufenthaltsdauer sehen möchte. Erst danach legt er fest, ob er diese Infos für die Reader oder den Raum sehen will.	7	Fixed
6	3	**Kontextinfos in Statuszeile** In der Statuszeile werden bei den Statistik-Visualisierungen falsche Infos angezeigt. • Sind die Besucherzahlen gewählt, so wird "Statistiken der Aufenthaltsdauer" angezeigt • Sind die Aufenthaltsdauern gewählt, so wird "Statistiken der Besucherzählung" angezeigt	10	Fixed
7	3	**Verbesserung der Zeitdarstellung** • Darstellung der Zeiten in 3 Zeilen (Avg, Min, Max) • mm:ss anstelle von hh:ss[[BR]] • Hinweis auf das verwendete Zeitformat (z.B. xx min : xx sek)	7	Fixed
8	3	**Besucher pro Raum / Reader - Total der Besucher** In der aktuellen Lösung der Visualisierung "Besucher pro Raum", bzw. "Besucher pro Reader" ist nicht ersichtlich, was das Total der erfassten Besucher ist. Dies wäre evtl. interessant zu wissen, damit die in der Visualisierung angezeigten Zahlen in ein Verhältnis gesetzt werden können.	8	
9	3	**Zeiten zwischen Readern im selben Raum** Bei den Aufenthaltsdauern pro Raum werden momentan die Zeiten zwischen zwei Readern desselben Raumes auch in die Statistik aufgenommen. Der Productowner wünscht jedoch, dass diese Zeiten ignoriert werden, also nur die	9	Fixed

		Zeiten, welche wirklich innerhalb des Readers waren, für die Berechnung einbeziehen.		
10	3	**Kontextwechsel** Im Dockable gibt es momentan Daten und Visualisierung. Falls jetzt 2 Fenster, und bei einem ändert die Visualisierung, während beim anderen Daten ausgewählt ist. Sieht die Kontext-Verknüpfung folgendermassen aus: Visualisierung -> auf Fenster 1 Daten -> auf Fenster 2	3	Fixed
11	3	**Korrekte Sortierung der FilterAttribute** • Sortierung der DB für "Kinder, Senioren Jugendliche..." zu "Kinder, Jugendliche, Senioren...". D.h. eine Sortierung nach Alter. • Möglicherweise muss dadurch DB angepasst werden. • Andernfalls anhand von PD die gewünschte Sortierung erreichen.	6	Fixed
12	3	**Filterdockable mit Scrollbar** Filterdockable soll eine Scrollbar erhalten, damit bei vielen Filtern die Übersicht beibehalten wird.	3	Fixed
13	3	**Checkboxen per Default selektiert** Die Checkboxen für die Filterauswahl sollte per Default auf selektiert(X) anstelle von deselektiert ( ) sein.	8	Fixed
14	3	**Gruppen in eine Tabelle** Zusätzlich sollen noch Gruppenlisten mit Checkboxes hinzufügen werden.	7	Fixed
15	4	**Auswahl der Zeitspanne** bei der automatischen Zeitanpassung entsteht folgendes Problem: • Ausgangslage: Falls selber Tag gewählt Start: 22:59 end: 23:10 • Problem: es wird nun "Start" auf 23:59 mit "End" auf 00:10 von selben Tag gesetzt was nicht korrekt wäre!	8	Fixed
16	4	**Nutzung der Fenstertitel** Nutzen der Fenstertitel der Visualisierungsfenster für Informationen, z.B. "Naturama, 1. Obergeschoss" wäre	7	Fixed

		nützlich.		
17	4	**Unerwünschte Rücksetzung des Filters** Wenn ein Filter eingestellt ist und dann in dasselbe Visualisierungsfenster klickt, wird der Filter zurückgesetzt. Das passiert nicht, wenn man in ein anderes Visualisierungsfenster klickt und dann wieder in das erste.	6	Fixed
18	4	**Zeitüberprüfung** Beim Wechsel auf dasselbe Datum, muss zusätzlich auch eine Zeitüberprüfung stattfinden. Sonst könnte unter Umständen falsche Zeitwerte beim Filter gesetzt werden.	5	Fixed
19	5	**Slider** Die Slider sehen im Moment ziemlich hässlich aus (nur die Slider, nicht die gesamte Farbauswahl!). Kann man da optisch was rausholen?	4	Fixed
20	5	**Zeit mittels "Pfeilen"** Aktuell ist es so, dass das Datum / Zeit per direkter Eingabe oder Pfeiltasten verändert werden kann. Gerade auf die Tastatur sollte man (z.B. mit einem Tooltipp) vielleicht hinweisen. Eine denkbare Erweiterung wäre die Verwendung von Pfeil-Buttons hinter den Textboxen für das Erhöhen, bzw. Verringern der Feldwerte (siehe Anhang).	6	Fixed
21	5	**Aktualisierung der Farbeinstellungen** Übernehmen der Farbwerte ohne Neustart: • zunächst bleiben die Fenster gleich • beim Hineinklicken in eines der Fenster, wird die Farbe gewechselt • allerdings nicht rechts in der Einstellanzeige	5	Fixed
22	5	**Subfenstertitel ist immer gleich** Der Titel der Visualisierungsfenster ist immer gleich. Hier könnte auch stehen: • Aufenthaltsdauer in Minuten:Sekunden • Besucherzahlen • Besucherzahlen Prozent	2	open
23	5	**dynamischer Titel**	6	Fixed

		Der Titel vom Dockable soll automatisch der Anfang/End-Grenzen angepasst werden.		
24	5	**Sliderfarben** In den Sliderfarben werden bisher nur die total gesättigten akzeptiert im Slider. Grund ist, die Interpolation zw. den Farben geht momentan nicht über Sättigung.	8	Fixed
25	6	**Tooltipp auf Graphen** Tooltipp auf dem Graphen: "Alle Besucher: (Datum Zeit, Anzahl)": Keine Klammerung. Zeit ist überflüssig/falsch.	7	Fixed
26	6	**Grafikfehler bei JFreeChart** Auf einigen Rechner gibt es einen Grafikfehler beim Bildaufbau der JFreeChart. Möglicherweise muss nochmaliges „repaint“ geben.	3	open
27	6	**Anpassungen im Slider/-modell** • Das Slidermodell soll die minRecordedValue und maxRecordedValue im Modell gehandelt werden und nicht im JSlider. • Auch sollen die einzelnen Positionen klarer benannt werden: 1. value 2. value+extent 3. minPossibleValue 4. maxPossibleValue 5. minRecordedValue 6. maxRecordedValue	2	open
28	6	**Visualisierung - Ausblenden von GUI-Elementen** Aktuell werden bei den "globalen Statistiken" die nicht benötigten GUI-Elemente nur deaktiviert. Schöner werde ein komplettes Ausblenden dieser Komponenten.	2	open
29	6	**Anzeige der Aufenthaltsdauer** Der DrawString soll angepasst werden, damit dieser besser strukturiert und somit besser lesbar wird. Konkret heisst das, dass Tabs oder Leerzeichen eingefügt werden sollen.	3	Fixed
30	6	**Ausblenden der Etagenwahl** Im MuseumChooserDockable sollen bei der Museumsvisualisierung die Etagenwahl und Museumszustand ausgeblendet werden.	6	open

11.3 Begründung

Nr	Sprint	Begründung / Beschreibung
1	-	**Visualisierung - Refactoring der Klasse "WindowSettings "** Die Klasse „WindowSettings“ beinhaltet seit dem ersten Sprint schon eine unnötige Ansammlung an Funktionen, Methoden und Variablen. **Das Problem** liegt darin, dass diese Klasse eine zentrale Schnittstelle zwischen der Darstellung im Panel und der Einstellungen im Dockable darstellt. Nicht jede Visualisierung benötigt aber auch das ganze Spektrum dieser Klasse. So zum Beispiel bietet sie Methoden an, die von der Partnergruppe „Wege“ benötigt werden, die von uns aber nie benutzt wurden. Hier wäre ein Refactoring mit Vererbung angebracht. Da diese Klasse die gemeinsame Schnittstelle darstellt, ist dies nicht ganz trivial.
22	5	**Subfenstertitel ist immer gleich** Der Titel der Visualisierungsfenster ist bleibt immer gleich. Obwohl diese Anzeige sich gut für eine zusätzliche Informationsausgabe gut eignen würde. **Das Problem** befindet sich in der Dockable-Library. Denn diese unterstützt keine dynamische Änderung des Dockabletitels.
26	6	**Grafikfehler bei JFreeChart** Auf einem Rechner ist ein Grafikfehler beim Bildaufbau der JFreeChart aufgekommen. **Problem** scheint hier - nach ersten Analysen - eine veraltete Grafikkarte zu sein. Da diese Erkenntnis erst wenige Tage vor Semesterende aufgetreten ist, konnte dieses Problem nicht behoben werden.
27	6	**Anpassungen im Slider/-modell** Hierbei handelt es sich um interne Code-Richtlinien die nicht sauber implementiert wurden. **Das Problem** liegt daran, dass noch Umbenennungen im Code gemacht werden müssen. Das heisst, der Quellcode deckt sich noch nicht mit der in der Dokumentation beschriebenen Bezeichnungen. In der Klasse JSlider müssen folgende Umbenennungen gemacht werden, damit die Namen mit der Dokumentation übereinstimmen: 1. value 2. value+extent 3. minPossibleValue 4. maxPossibleValue 5. minRecordedValue 6. maxRecordedValue Für dieses Refactoring wurde bis anhin keine Zeit gefunden, da diese keine hohe Priorität aufweist. Aus zeitlichen Gründen ist diese Anpassung noch nicht geschehen.
28	6	**Visualisierung - Ausblenden von GUI-Elementen** Aktuell werden bei den "globalen Statistiken" die nicht benötigten GUI-Elemente nur deaktiviert. Schöner wäre ein komplettes Ausblenden dieser Komponenten. **Problem** befindet sich in dem Verhalten der Dockable-Library. Das heisst, die Dockables verhalten sich nicht so wie erwartet. Beispielsweise wird beim Ausblenden von einzelnen grafischen Komponenten diese zwar ausgeblendet, beim Anzeigen hingegen erscheint das Dockable nicht mehr. Lösungsansatz wäre hier, dass die Komponenten nicht unsichtbar, sondern nur deaktiviert werden – was auch der aktuellen Implementation entspricht.
30	6	**Ausblenden der Etagenwahl** Diese Anpassung ist eigentlich gar nicht notwendig, da – im Normalfall – die gegebene Komponente unsichtbar geschaltet wird. Momentan ist dies jedoch noch nicht möglich (siehe Nr. 28). Ein ausblenden der Textinhalte scheint hier nicht

angebracht, da dies nicht das ursprüngliche Problem lösen würde, sondern nur eine Umgehung des eigentlichen Problems darstellt.

12 Einleitung und Übersicht

In diesem Kapitel wird eine Übersicht der 6 Sprints geboten und eine Einleitung über das vom Projektleiter erstellte Problemdomain, auf welchem die in dieser Arbeit erstellten Artefakte aufbauen.

12.1 Problemdomain

Das Problemdomain wurde bereits vorweg erstellt und basiert auf folgendem Domainmodell:

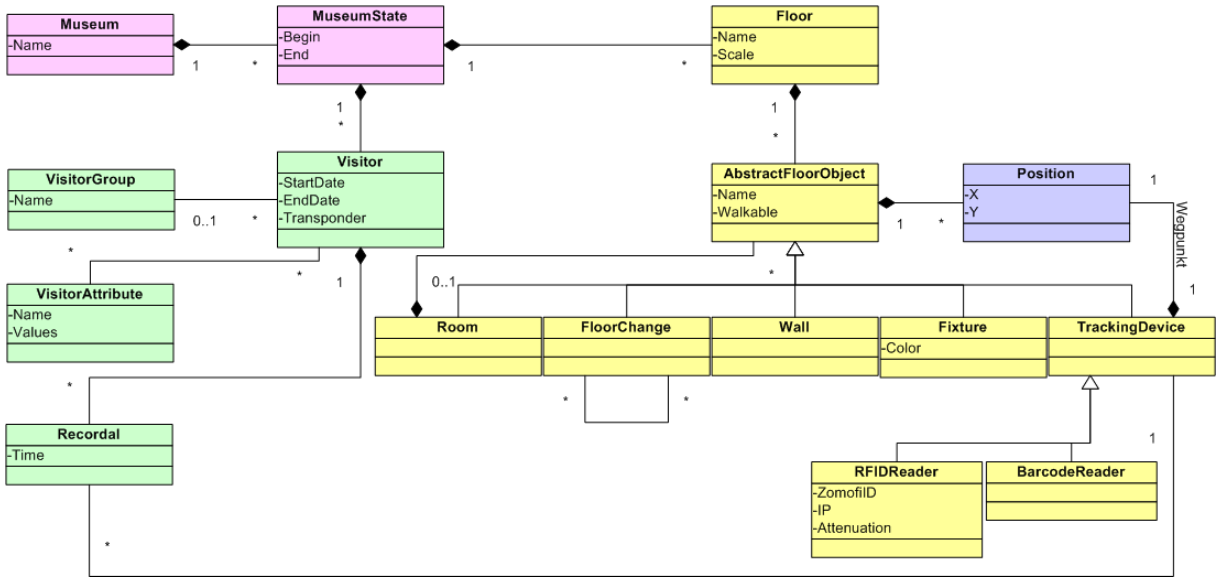


Abbildung 24: Aktuelles Problemdomain

In der Tabelle werden die einzelnen Konzepte genauer beschrieben:

Konzept	Beschreibung
Museum	Repräsentiert ein Museum, welches aus einer Menge von Museumszuständen definiert ist.
MuseumState	Ein Museumszustand bezeichnet eine Ausstellung im Museum und beinhaltet die Museumsstruktur (z.B. die Ausstellungsobjekte). Nach Abschluss einer Ausstellung kann für die nächste Ausstellung ein neuer Museumszustand definiert werden. Die erfassten Besucher pro Ausstellung werden mit dem Zustand assoziiert.
Visitor	Repräsentiert einen Besucher, der mit einem RFID Tag ausgestattet wurde und somit von einem RFID Reader erfasst werden kann.
VisitorGroup	Besucher können zu Gruppen wie zum Beispiel Schulklassen oder Familien zusammengefasst werden.
VisitorAttribute	Die Besucher bei der Tag Ausgabe mit bestimmten Attributen erfasst, beispielsweise demographische Daten wie Alter und Geschlecht.
Recordal	Jede Erfassung in einem RFID Reader wird in einem Recordal festgehalten.
Floor	Eine bestimme Etage im Museum
AbstractFloorObject	Fasst eine Menge von Objekten zusammen, welche sich auf einer Etage befinden können.

Room	Raum auf der Etage.
FloorChange	Steht für einen Etagenwechselpunkt, was beispielsweise einem Treppenhaus oder Lift entspricht.
Wall	Wand auf der Etage.
Fixture	Ein Ausstellungsobjekt auf der Etage.
TrackingDevice	Kapselt die möglichen Erfassungsgeräte.
RFIDReader	Erfassen die RFID Tags der Besucher über einige Meter hinweg.
BarcodeReader	Macht ein RFID Tag an der Kasse aktiv für Erfassungen. Am Ende eines Besuchs wird der Tag wieder deaktiviert.

12.2 Übersicht pro Sprint

In jedem Sprint wurden bestimmte Arbeitspakete (Userstories) umgesetzt, welche in der folgenden Tabelle pro Sprint kurz zusammengefasst wurden.

Sprint Nummer	Beschreibung
1	• Einarbeitung in den bestehenden Code • Besucheranzahl pro (RFID) Reader bestimmen und darstellen • Besucheranzahl pro Raum bestimmen und darstellen • Visuelle Unterstützung der Besucheranzahl mit Farbe
2	• Aufenthaltsdauer pro Reader bestimmen und darstellen • Aufenthaltsdauer pro Raum bestimmen und darstellen • Visuelle Unterstützung der Aufenthaltsdauer (mit Farbskalierung)
3	• Filterung nach Zeit, Besucherattribute und Besuchergruppen • Besucheranzahl in Prozent bestimmen und darstellen
4	• Farbskalen (für die visuelle Unterstützung) einstellbar für den Benutzer • Farbskalen über Programmende hinweg speichern
5	• Besucheranzahl pro Tag des gesamten Museum bestimmen und darstellen • Auswahl des dargestellten Zeitbereiches
6	• Besucheranzahl mit bestimmten Attribut bestimmen und darstellen • Verteilungen als Kuchen- und Balkendiagramme pro Reader und Raum

12.3 Packageübersicht

Es werden nur die Packages aufgeführt, welche uns im Laufe der Arbeit betroffen haben. Der Prefix *ch.hsr.ifs.visivis* wurde bei der Auflistung weggelassen.

Packagename	Beschreibung
shared.pd.*	Enthält die Problemdomain Klassen
shared.ui.helper	Enthält Hilfsklassen für die Visualisierungen
shared.ui.settings.*	Enthält die Einstellungspanels der Anwendung
visual.pd.statistic	Enthält alle Statistikklassen, welche in dieser Arbeit umgesetzt wurden
visual.ui.dockables.*	Enthält die Seitenpanels am rechten Rand des Applikationsfensters um Änderungen an der dargestellten Visualisierung vorzunehmen
visual.ui.floor	Enthält die Etagenvisualisierung
visual.ui.museum	Enthält die Museumsvisualisierung

13 Sprint 1

13.1 Änderungsgeschichte

Datum	Änderung
13.10.2009	Erstellung der Wikiseite.
18.10.2009	Hinzufügen von Algorithmen und deren Analysen.
02.11.2009	Einfärbung und div. Klassendiagramme erstellt.
02.11.2009	Erstellung der Testdokumentation zum Algorithmus.
14.12.2009	Hinzufügen in Gesamtbericht.

13.2 Beschreibung

Der erste Sprint legte den Fokus auf die Einarbeitung des bestehenden Codes, Kennenlernen der Entwicklungsumgebung und Visualisierung der Besucheranzahl pro Raum und (RFID) Reader. Die berechnete Besucheranzahl soll durch Untermalung mit Farben visuell unterstützt und jeweils in der Mitte des Raumes bzw. Readers dargestellt werden.

13.3 Zentrierungsalgorithmus

Das Finden einer Stelle im Polygon, welche am meisten Platz für die Darstellung eines Wertes bietet ist nicht so trivial, wie es auf den ersten Blick den Anschein macht. In den folgenden Unterkapiteln werden deshalb Algorithmen für das Finden einer solchen Stelle begutachtet und anschliessend entschieden, welche sich für eine Implementierung eignen.

13.3.1 Analyse

Der Algorithmus hat die Aufgabe die bestmögliche Position in einem Polygon zu finden, so dass eine Anzeige (z.B. Text) platziert werden kann. Bestmöglich bedeutet, dass sich die Anzeige innerhalb des Polygons befindet und am meisten Platz einnehmen kann. Würde die Anzeige durch einen schlechten Algorithmus beispielsweise ausserhalb des Polygons platziert, könnte ein Betrachter nur schwierig eine Verbindung zwischen dem dargestellten Wert und dem zugehörigen Objekt herzustellen. Im schlimmsten Fall würde die Anzeige einem anderen Objekt zugeordnet, als sie von der Visualisierung her geplant wäre.

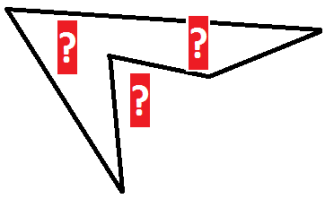


Abbildung 25: Polygon

Analytisch betrachtet ist die Stelle in einem beliebigen (konvexen/konkaven) Polygon gesucht, welche die grösstmögliche Fläche für ein waagerecht ausgerichtetes Rechteck bietet. Als Rechteck ist hier beispielsweise der Platz gemeint, welcher ein Textobjekt einnimmt. Im Folgenden werden verschiedene Algorithmen analysiert.

13.3.1.1 Ansatz: Der Inkreis

Der Inkreis eines Polygons ist der Kreis, der alle Seiten eines Polygons in ihrem Inneren berührt. Dieser tangiert die Strecken zwischen den Eckpunkten und nicht deren Verlängerungen. Gleichzeitig ist er der Kreis, welcher die grösste Fläche abdeckt. Der Algorithmus hat allerdings die Einschränkung, nur mit konvexen Polygonen umgehen zu können.

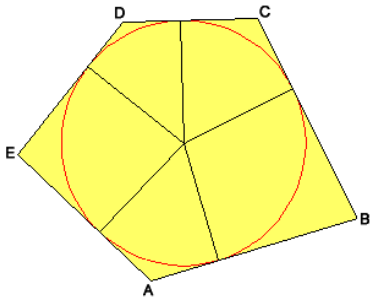


Abbildung 26: Einfaches (konvexes) Polygon mit Inkreis[14]

13.3.1.2 Ansatz: Straight Skeleton

In der Geometrie ist ein Straight Skeleton eine Methode, welche die topographische Struktur eines Skeletons beschreibt (Höhenlinie). Dieser Algorithmus wäre einer der besten Ansätze für das Platzierungsproblem, denn der Punkt mit der höchsten Topographie wäre gleichzeitig derjenige mit dem grössten Abstand zu den Rändern des Polygons. Allerdings, ist das Straight Skeleton äussert komplex zu implementieren. Aus diesem Grund wurde das Straight Skeleton nicht weiter betrachtet.

Beispiel Topographie:

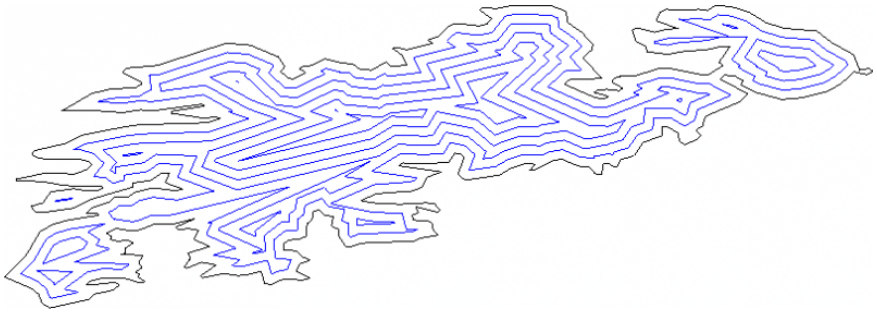


Abbildung 27: Topographische Analyse eines Polygons[15]

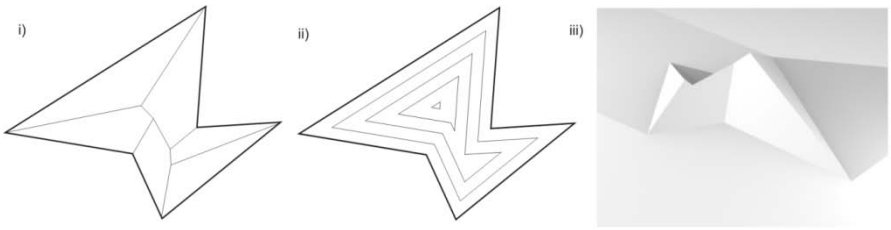


Abbildung 28: Drei verschiedene Visualisierungen des Straight Skeleton[16]

13.3.1.3 Ansatz Schwerpunkt

„Im Sinne der klassischen Mechanik ist der Schwerpunkt der Punkt, an dem die Masse des Körpers die gleiche Wirkung auf andere Körper hätte, wenn sie in diesem Punkt vereint wäre. Umgekehrt kann

[14] http://de.wikipedia.org/wiki/Inkreis, letzter Zugriff 28.11.09
[15] http://en.wikipedia.org/wiki/Straight_skeleton, letzter Zugriff 28.11.09
[16] http://en.wikipedia.org/wiki/Straight_skeleton, letzter Zugriff 28.11.09

man die Gravitation, die auf alle Massenpunkte des Körpers wirkt, durch eine einzige Kraft darstellen, die im Schwerpunkt angreift.“ [17]

Nachfolgend wird beschrieben, wie der Schwerpunkt in einem Polygon berechnet werden kann.

13.3.1.3.1 Ausgangslage

Gegeben sei ein nicht überschlagenes und geschlossenes Polygon mit N Eckpunkten. Der nullte Eckpunkt (x0, y0) und der N-te Eckpunkt (xN, yN) sind hierbei identisch. Gesucht wird der Schwerpunkt M(cx, cy) im Polygon.

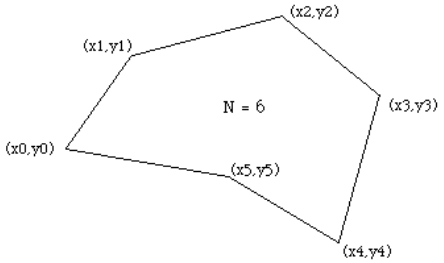


Abbildung 29: Polygon mit N=6 Punkte

13.3.1.3.2 Algorithmus

Für die Berechnung wird die Fläche des Polygons benötigt:

$$A = \frac{1}{2} \sum_{i=0}^{N-1} (x_i\, y_{i+1} - x_{i+1}\, y_i)$$

Diese wird durch die Aufsummierung von Dreiecken / Quadrate erreicht, da sich jedes Polygon in Rechtecke / Dreiecke unterteilen lässt.

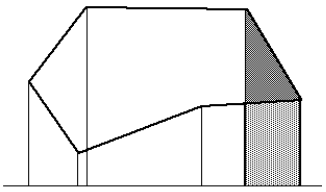


Abbildung 30: Aufteilung eines Polygons

Das Gravitationszentrum (Schwerpunkt) wird anhand der folgenden Formel berechnet:

$$c_x = \frac{1}{6A} \sum_{i=0}^{N-1} (x_i + x_{i+1})(x_i\, y_{i+1} - x_{i+1} y_i) \qquad c_y = \frac{1}{6A} \sum_{i=0}^{N-1} (y_i + y_{i+1})(x_i\, y_{i+1} - x_{i+1} y_i)$$

13.3.1.3.3 Probleme

Einzig bei einer Überschneidung des Polygons, schlägt dieser Algorithmus fehl:

[17] http://de.wikipedia.org/wiki/Schwerpunkt#Physikalischer_Schwerpunkt, letzter Zugriff 28.11.09

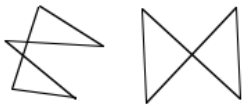


Abbildung 31: Überschneidung im Polygon

13.3.1.4 Ansatz Mittelpunkt

Durch die Aufsummierung aller X- und Y-Koordinaten der Eckpunkte im Polygon wird der Mittelpunkt bestimmt.

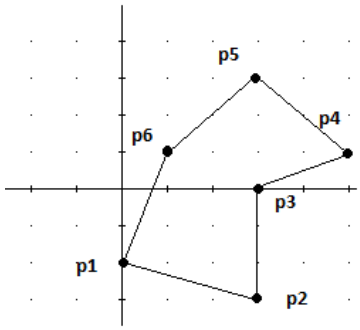


Abbildung 32: Geschlossenes Polygon[18]

13.3.1.4.1 Ausgangslage

Benötigt werden alle Eckpunkte des Polygons (im Bild wären dies die Punkte p1 bis p6). Der Mittelpunkt wird durch die Aufsummierung aller Koordinaten und dessen Berechnung des Mittelwerts gesucht.

13.3.1.4.2 Algorithmus

Hier werden die Punkte in ihrer jeweiligen Koordinaten aufsummiert und durch ihre Anzahl Punkte dividiert.

$$x = \frac{1}{m}\sum_{j=1}^{m} p_j(x) \qquad \mathbf{y} = \frac{1}{m}\sum_{j=1}^{m} p_j(\mathbf{y})$$

13.3.1.4.3 Probleme

Falls ein extrem verzogenes Polygon gegeben ist, oder viele Eckpunkt an einer Stelle konzentriert sind, schlägt dieser Algorithmus fehl. Eine Anhäufung von Punkten zieht den Mittelpunkt an, was zu ungeeigneten Resultaten führen kann.

13.3.1.5 Pro und Kontra der Algorithmen

Durch eine Gegenüberstellung von Pro und Kontra wird versucht einen oder mehrere Algorithmen zu definieren die implementiert werden sollen:

Algorithmus	Pro	Kontra	Entscheidung
Inkreis	Sehr gute Positionierung in einem konvexen Polygon.	Nur für konvexe Polygone geeignet.	Nicht implementiert, da zu spezifisch
Straight-Skeleton	Es wird womöglich die	Zu komplex.	Nicht

[18] http://www.analyzemath.com/, letzter Zugriff 16.12.09

	optimalste Stelle gefunden, wo ein Textobjekt platziert werden könnte.		implementiert, da zu komplex
Schwerpunkt	Liefert sehr gute Punkt im konvexen Polygon und hat eine gute Laufzeit.	Möglicherweise schlechte Resultate bei konkaven Polygonen.	Implementiert, da einfach umzusetzen mit ansprechenden Resultaten
Mittelpunkt	Schnell und leicht zu berechnen.	Zeigt schlechte Resultate bei Polygonen mit Anhäufungen von Eckpunkten, welche den Mittelpunkt zu sich ziehen.	Implementiert, da sehr einfach umzusetzen

13.3.2 Design

Da jeder Reader oder Raum als ein Polygon repräsentiert wird, kann auch nach einem optimalen Punkt innerhalb dieses Polygons gesucht werden. Aus der Analyse ging hervor, dass folgende Algorithmen umgesetzt werden sollen:

- Schwerpunkt-Algorithmus
- Mittelpunkt aller Ecken

13.3.2.1 Strategypattern für den Zentrierungsalgorithmus

Den optimalen Mittelpunkt in einem beliebigen Polygon zu finden, ist kein triviales Problem. Wie so oft in komplexeren Problemstellungen existieren mehrere mögliche Lösungsansätze. Aus diesem Grund wurde das Strategypattern verwendet, um eine einfache Auszutauschungen zu erlauben. Nachfolgend ist das Klassendiagramm mit den beiden Implementierten Strategien abgebildet:

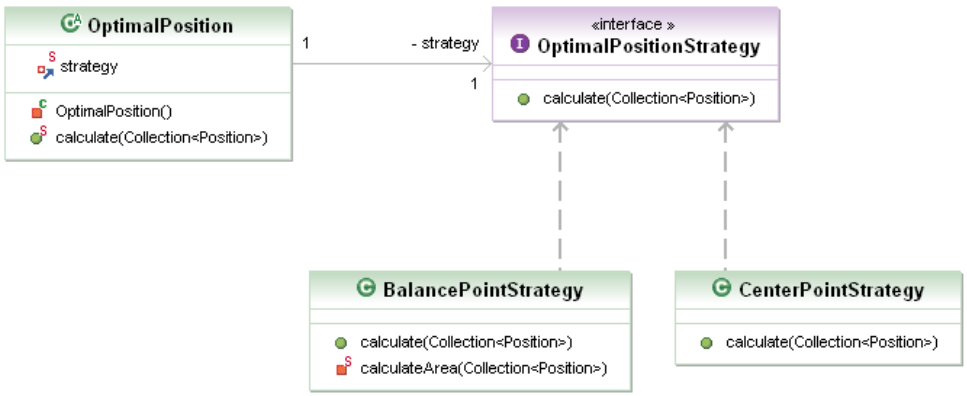


Abbildung 33: Strategypattern für die Zentrierung

13.3.3 Tests

Durch Unit Tests wurden die umgesetzten Algorithmen getestet. In den folgenden Berechnungen wird gezeigt, welche Tests ausgeführt wurden. Die Vergleichswerte der jeweiligen Algorithmen wurden alle von Hand ausgerechnet um einen Vergleichswert zu erhalten. Exemplarisch wird für jeden Algorithmus jeweils ein Beispiel illustriert. Extremfälle wie konkave Polygone wurden bewusst

nicht in die Testfälle aufgenommen, da die erste Version des Algorithmus auf diese Sonderfälle nicht eingehen soll.

13.3.3.1 Schwerpunkt-Algorithmus

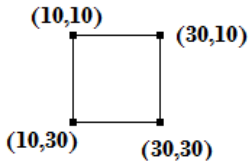
Fläche: $A = \frac{1}{2} + \sum_{i=0}^{N-1}(x_i * y_{i+1} - x_{i+1} * y_i)$

x-Koordinate = $c_x = \frac{1}{6A}\sum_{i=0}^{N-1}(x_i * x_{i+1})(x_i * y_{i+1} - x_{i+1} * y_i)$

y-Koordinate = $c_y = \frac{1}{6A}\sum_{i=0}^{N-1}(y_i * y_{i+1})(x_i * y_{i+1} - x_{i+1} * y_i)$

Beispiel an einem Polygon mit vier Eckpunkten:

```
polygon0 = new ArrayList<Position>();
polygon0.add(new Position(10, 10));
polygon0.add(new Position(30, 10));
polygon0.add(new Position(30, 30));
polygon0.add(new Position(10, 30));
```



13.3.3.1.1 Validierung der Berechnung:

$A = 400.0$

x-Koordinate:
für i=0 gilt (10+30)*(10*10-30*10) = -8000
für i=1 (30+30)*(30*30-30*10) = 36000
für i=2 (30+10)*(30*30-10*30) = 24000
für i=3 (10+10)*(10*10-10*30) = -4000

$$c_x = \frac{1}{6A}(-8000 + 36000 + 24000 - 4000) = \frac{1}{6A}(48000) = 20.0$$

y-Koordinate:
für i=0 gilt (10+10)*(10*10-30*10) = -4000
für i=1 (10+30)*(30*30-30*10) = 24000
für i=2 (30+30)*(30*30-10*30) = 36000
für i=3 (30+10)*(10*10-10*30) = -8000

$$c_y = \frac{1}{6A}(-4000 + 24000 + 36000 - 8000) = \frac{1}{6A}(48000) = 20.0$$

Koordinate auf Integer gerundet ist folglich: $M(x, y) = (20,20)$

13.3.3.2 Mittelpunkt-Algorithmus

x-Koordinate = $y = \frac{1}{m}\sum_{j=1}^{m} P_j(y)$
y-Koordinate = $x = \frac{1}{m}\sum_{j=1}^{m} P_j(x)$

Beispiel an einem Polygon mit 5 Eckpunkten:

```
polygon5 = new ArrayList<Position>();
polygon5.add(new Position(3, 3));
polygon5.add(new Position(4, 1));
polygon5.add(new Position(5, 1));
polygon5.add(new Position(10, 3));
polygon5.add(new Position(10, 5));
polygon5.add(new Position(5, 7));
polygon5.add(new Position(4, 7));
polygon5.add(new Position(3, 5));
```

13.3.3.2.1 Validierung der Berechnung:

x-Koordinate = $x = \frac{1}{8}(3 + 4 + 5 + 10 + 10 + 5 + 4 + 3) = \frac{1}{8}(44) = 5.5$

y-Koordinate = $y = \frac{1}{8}(3 + 1 + 1 + 3 + 5 + 7 + 7 + 5) = \frac{1}{8}(32) = 4.0$

Die Koordinate auf Integer gerundet ist folglich: $M(x, y) = (6, 4)$

13.4 Datensammlung

In diesem Sprint wurde der StatistikManager entwickelt. Er kapselt alle statistisch relevanten Berechnungen und stellt die berechneten Werte der Visualisierung zur Verfügung. Aktuell wurde die Anzahl Besucher pro Raum und Reader berechnet, weitere relevante Werte wie die Aufenthaltsdauer werden in weiteren Sprints folgen.

13.4.1 Analyse

Als erstes musste aus dem gegebenen Problemdomain die Menge an Konzepten ausgewählt werden, welche für die Statistiken von Belangen sind. Nachfolgend werden einige Ausschnitte der wesentlichsten Konzepte und deren Attribute dargestellt.

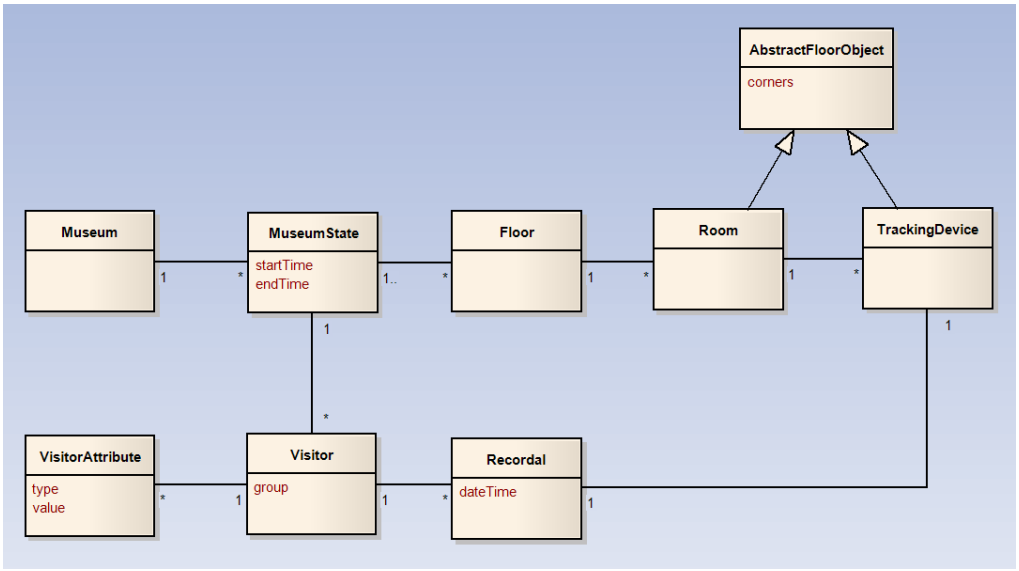


Abbildung 34: Übersicht der Problemdomain

13.4.1.1 Überblick der der Konzepte

Konzept	Beschreibung
Museum	Das Museum ist die oberste Stufe der Museumshierarchie und besteht aus Museumzuständen.
MuseumState	Mit Museumszustand wird die laufende Ausstellung im Museum bezeichnet, welche von der Museumsleitung von Zeit zu Zeit umgestellt wird. Eine neue Ausstellung wird durch einen neuen Museumszustand abgebildet. Alle registrierten Besucher in der Zeit während der Ausstellung werden mit dem entsprechenden Zustand assoziiert.
Visitor	Bezeichnet einen Besucher, welcher an der Kasse mit einem RFID Tag ausgestattet wurde. Die Kasse erfasst dabei bestimme Attribute des Besuchers, wie zum Beispiel demographische Daten (Alter, Geschlecht, etc.). Die Erfassungen, welche das RFID Tag bei den RFID Readern auslöst, werden mit dem Visitor assoziiert.
Recordal	Entspricht einer Erfassung zu einem bestimmten Zeitpunkt, welche bei einem RFID Reader ausgelöst wurde.
Floor	Eine Etage besteht aus mehreren AbstractFloorObjects.
AbstractFloorObject	Ein AbstractFloorObject ist alles, was sich auf einer Etage befindet. Dies sind unter anderem Räume, Reader, Wände, Ausstellungsobjekte. Da all diese Objekte in der Visualisierung als Polygone dargestellt werden, führt ein AbstractFloorObject eine Liste von Eckpunkten. Für die Statistiken spielen hauptsächlich die Räume und Reader eine wesentliche Rolle.
Room	In einem Raum befindet sich eine Menge von RFID Readern.
TrackingDevice	Bezeichnet einen RFID Reader, dessen Empfangsbereich die RFID Tags der Besucher registriert.

Wie aus den dargestellten Assoziationen abzulesen ist, besteht keine direkte Verbindung von Visitor und TrackingDevice. Um nun die Anzahl Besucher pro TrackingDevice und Raum zu bestimmen, müssen diese zuerst mit den Besuchern in Relation gesetzt werden.

13.4.2 Design

Die Analyse hat ergeben, dass zuerst eine Verbindung zwischen den Readern und den erfassten Besuchern im Museumszustand hergestellt werden muss. Nach Gelingen dieses Schrittes ist die Menge der erfassten Besucher pro Reader bekannt. Aus dieser Menge ist die Anzahl Besucher leicht zu bestimmen. Um die Besucheranzahl pro Raum zu berechnen, wird auf die im vorherigen Schritt bestimmten Besuchermengen pro Reader zurückgegriffen. Konkret bedeutet dies, dass zu einem Raum die Besuchermengen auf dessen Reader zusammenfasst werden. Dadurch wird die Anzahl der Besucher pro Raum bestimmt.

13.4.2.1 Klassendiagramm

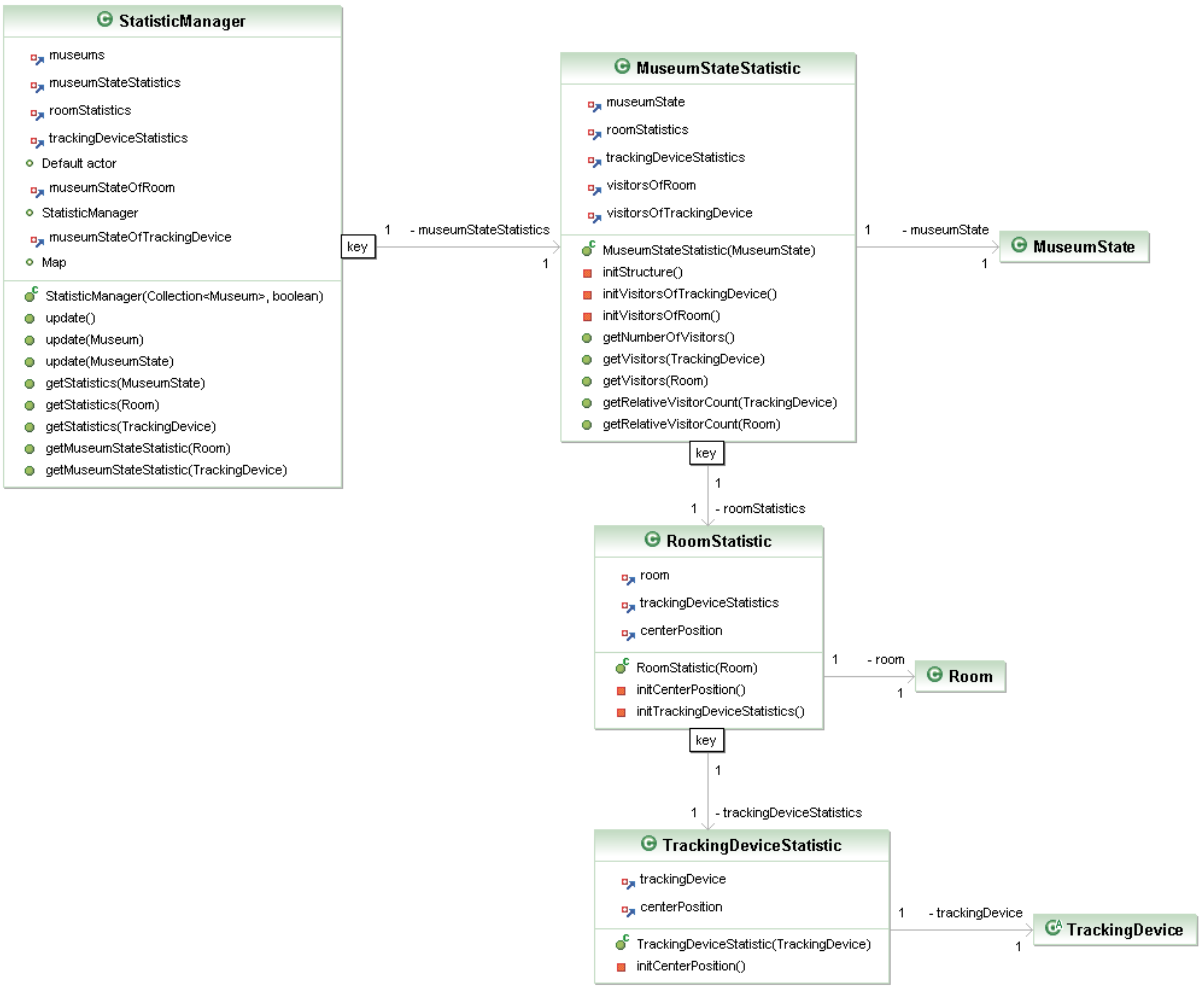


Abbildung 35: Klassendiagramm für die Statistiken

13.4.2.2 Beschreibung des StatisticManagers

Der StatisticManager wird beim Laden der Applikation erstellt und an alle Fenster weitergereicht. Er bietet ein Interface zum dynamischen Berechnen der statistischen Daten. Ein Fenster, welches im Besitz des StatisticManagers ist, kann selber bestimmen wann und welche Daten berechnet werden sollen. Dies ermöglicht es nur die Teile zu berechnen, welche in der gewählten Visualisierung sichtbar sind.

Schnittstelle	Beschreibung
StatisticManager(Collection<Museum>, boolean update)	Der Konstruktor wird mit der Menge aller Museen aufgerufen. Über das Flag *update* kann angegeben werden, ob die *update()* Methode bereits im Konstruktor aufgerufen werden soll.
update() **update(Museum museum)** **update(MuseumState museumState)**	Die Methode *update()* wird in der Regel im Konstruktor implizit aufgerufen. Sie iteriert durch alle Museen, welche im Konstruktor angegeben wurden. Es ist zusätzlich möglich, die statistischen Daten nicht über alle Museen, sondern spezifisch nur über ein Museum oder einen Zustand berechnen zu lassen.
getStatistics(MuseumState museumState)	Liefert zu einem Problemdomain-Objekt die

getStatistics(Room) **getStatistics(TrackingDevice trackingDevice)**	berechneten Statistikdaten, gekapselt in einer Statistikklasse. Auf die einzelnen Statistikklassen wird im nächsten Kapitel genauer eingegangen.
getMuseumStateStatistic(Room room) **getMuseumStateStatistic(TrackingDevice trackingDevice)**	Da die Assoziationen im Problemdomain unzyklisch gehalten sind, kennt ein Room zwar seine TrackingDevices, die Umkehrrichtung ist jedoch nicht gegeben. Da die MuseumStateStatistic als Information Expert alle Besucherdaten enthält, muss die Visualisierung beim Iterieren über Reader und Räume auf die MuseumStateStatistic zurückgreifen können. Diese Methoden liefern zu einem Room oder TrackingDevice die passende MuseumStateStatistic zurück.

13.4.2.3 Beschreibung Statistikklassen

Die Statistikklassen erweitern eine Problemdomain-Klasse um weitere Felder und Methoden. In zukünftigen Sprints werden noch weitere Felder/Methoden hinzugefügt wie beispielsweise die Aufenthaltsdauer. Die MuseumStateStatistic ist der Information Expert bezüglich aller Besucherdaten (siehe Problemdomain Konzepte). Aus diesem Grund führt sie die berechneten Mengen der Besucher pro TrackingDevie und Raum. Es muss allerdings geprüft werden, ob die Besucherdaten nicht direkt in die entsprechenden Statistikklassen abgefüllt werden sollen. Eine FloorStatistic wurde nicht erstellt, da bis dato keine statistischen Daten bezüglich einer Etage verlangt sind.

13.5 Einfärbung

Bei der Einfärbung geht es darum, einen Zahlenwert visuell durch eine Farbe zu unterstützen. Beispielsweise könnte ein tiefer Wert rot und ein hoher Wert grün untermalt werden. Dadurch kann sich ein Benutzer schneller einen Überblick der dargestellten Werte verschaffen, ohne diesen exakt lesen zu müssen.

13.5.1 Analyse

Für eine geeignete Einfärbung wurden zwei Modelle begutachtet. Das Erste ist eine stufenweise Einfärbung mit mehreren Farbwerten und die Zweite ein kontinuierlicher Farbverlauf zwischen zwei Farben.

13.5.1.1 Stufenweise Einfärbung

Dieses Färbungsmodell wurde in den alten Lösungen der Software verwendet. Der Benutzer kann die Einfärbung anhand von fixen Zahlwerten vorgeben. Es ist momentan unklar, ob die Anzahl der Farben veränderbar oder wie im Bild fix auf fünf festgelegt werden soll.

Abbildung 36: Stufenweise Einfärbung

13.5.1.1.1 Vorteile

- Die Farbabdeckung kann vom Benutzer exakt bestimmt werden.

13.5.1.1.2 Nachteile

- Eventuell zu viele Einstellungsmöglichkeiten.
- Gefahr bei zu vielen unterschiedlichen Farben die Übersicht zu verlieren.

13.5.1.2 Kontinuierliche Einfärbung

Der Benutzer kann hier die Anfangs- und Endfarbe, sowie den entsprechenden minimalen (im Bild 100) bzw. maximalen Zahlenwert (im Bild 300) festlegen. Dies bedeutet, dass die Zahlenwerte unterhalb des minimal gewählten Wertes die Anfangsfarbe und alle Werte über dem maximal gewählten Wert die Endfarbe annehmen. Dazwischenliegende Werte erhalten die entsprechende Mischfarbe im Farbverlauf.

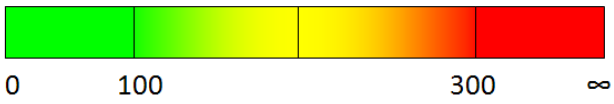


Abbildung 37: Kontinuierliche Einfärbung

13.5.1.2.1 Vorteile

- Weniger Einstellungen notwendig.
- Übersichtsverlust durch zu viele unterschiedliche Farben ist deutlich kleiner.

13.5.1.2.2 Nachteile

- Eventuell möchte der Benutzer eine Abstufung über mehr als zwei Farben einstellen.

13.5.1.3 Entscheidung

Die kontinuierliche Einfärbung scheint intuitiver und stellt für den Benutzer einen geringeren Einstellungsaufwand dar. Aus diesem Grund fiel die Entscheidung auf das zweite Einfärbungsmodell. Eventuell wird es in zukünftigen Sprints möglich sein, mehrere Färbungspunkte als nur das Minimum und Maximum anzugeben.

13.5.2 Design

Die Berechnung der Mischfarbe ist als statische Funktion im UIHelper angesiedelt:

public static Color ***getMixedColor(****Color color1, Color color2, double factor****)***

Zur Berechnung werden die Farben von RGB in das sogenannte HSB Farbschema (hue, saturation, brighness) umgewandelt. In diesem Schema ist es sehr einfach Farbverläufe zu berechnen. Ein Verlauf von rot zu grün ist beispielsweise durch einfaches erhöhen bzw. reduzieren des Farbtones (hue) zu erreichen. Auf den nachfolgenden Bildern sind die RGB und HSB Werte der Farben Rot, Gelb und Grün ersichtlich.

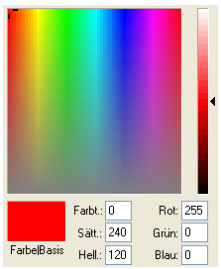


Abbildung 38: Rote Farbauswahl

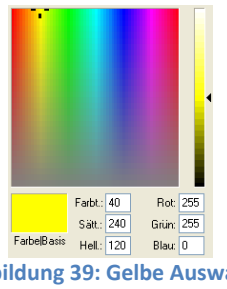


Abbildung 39: Gelbe Auswahl

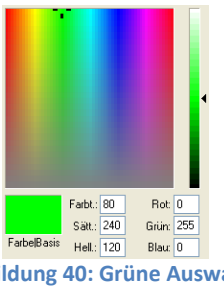


Abbildung 40: Grüne Auswahl

13.5.3 Offene Punkte

- Wir die Farbe Rot als erste und die Farbe Grün als zweite Mischfarbe angegeben, geht der Farbverlauf von Rot über Gelb nach Grün, also nicht direkt von Rot nach Grün. In der Visualisierung ist dieser Umweg über Gelb willkommen, eventuell wäre jedoch in zukünftigen Sprints eine direkte Mischung zweier Farben erwünscht.

13.6 Visualisierung

In der Visualisierung müssen die berechneten Werte sinnvoll dargestellt werden. Im aktuellen Sprint betrifft dies nur die Besucherzählung. Die Umschaltung zwischen den Raum und Reader Auswertungen findet momentan noch über einen Menüeintrag statt.

13.6.1 Analyse

In der Visualisierung soll ersichtlich sein, wie viele Besucher pro Reader/Raum erfasst wurden. Die Zahlen sollen dabei passend platziert und farbig untermalt werden. Diese beiden Teilprobleme wurden in den Kapiteln „Zentrierungsalgorithmus“ und „Einfärbung“ diskutiert und das Berechnen der Besucherzahl im Kapitel „Datensammlung“.

13.6.2 Design

Der Ablauf für die Visualisierung der Reader läuft analog zur der der Räume ab. Vom StatisticManager werden die relevanten Werte erhalten und entsprechend dargestellt. Auf den nachfolgenden Bildern sind die Reader Auswertung auf der linken und die Raum Auswertung auf der rechten Seite dargestellt. Die Einfärbung ist leicht transparent, damit darunter liegende Objekte und die Raumstruktur sichtbar bleiben. Räume welche kein TrackingDevice enthalten, wurden nicht eingefärbt.

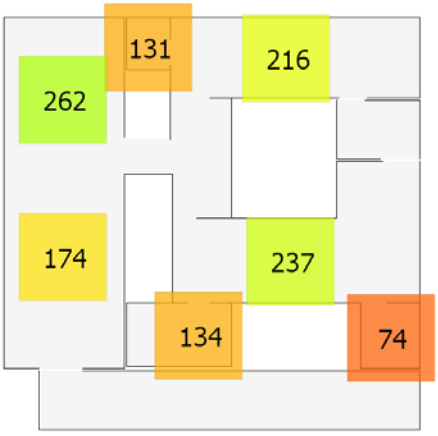


Abbildung 41: Besucher pro Reader

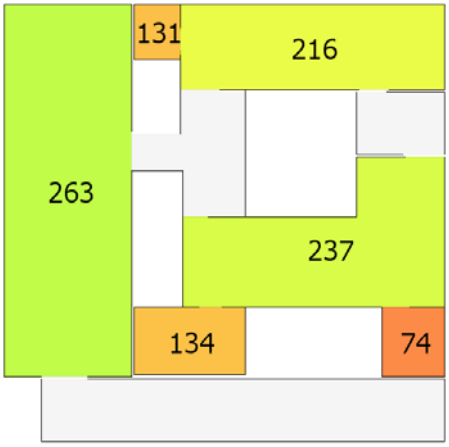


Abbildung 42: Besucher pro Raum

13.6.3 Offene Punkte

- Die Schriftgrösse ist momentan fix gewählt, das heisst sie zoomt mit, wenn der Ausschnitt verkleinert/vergrössert wird. Eventuell macht es jedoch Sinn, die Schriftgrösse unabhängig vom Zoomfaktor zu halten.
- Das Minimum und Maximum für die Einfärbung ist momentan auf 0 bis 'Total erfasste Besucher' festgelegt. In einem Museum mit hohen Besucherzahlen pro Reader oder Raum führt dies zu einer unzureichenden Einfärbung. Grund dafür ist, dass sich in der Regel die meisten Zählungen in der Nähe des Totals befinden und somit alle die Farbe Grün annähmen. Ein Vorschlag dieses Problem zu lösen wäre den Farbverlauf auf folgende Schlüsselwerte festzulegen: kleinste Zählung (z.B. 120 Besucher) bis grösste Zählung (z.B. 260 Besucher).
- Das Total an Besuchern ist in der Visualisierung nicht ersichtlich. Somit können die dargestellten Zahlen vom Betrachter aus nicht ins Verhältnis zur Gesamtmenge gebracht werden. Im nächsten Sprint wird aus diesem Grund ein Übersichtspanel mit den wichtigsten Schlüsseldaten implementiert.

14 Sprint 2

14.1 Änderungsgeschichte

Datum	Änderung
12.11.2009	Erstellung des Dokumentes.
16.11.2009	Darstellungen und Visualisierungsmöglichkeiten notiert.
16.11.2009	Hinzufügen von Beschreibungen der Statistikklassen
18.11.2009	Visualisierungen erneuert
15.12.2009	In Bericht hinzugefügt.

14.2 Beschreibung

Ziel im Sprint2 war es, die Aufenthaltsdauer pro RFID Reader zu bestimmen und die Etage / MuseumState wechseln zu können. Eine Filterung nach Zeit konnte in diesem Sprint leider nicht umgesetzt werden, da eine erste Analyse unerwartete Schwierigkeiten aufdeckte. Die Umsetzungsverschiebung dieses Tasks, in einen nächsten Sprint, zeigte sich auch dadurch sinnvoll, dass das Filtern als Ganzes (nach Zeit, Besucherattributen und Gruppen) in einem einzigen Sprint umgesetzt werden konnte. Im Gegenzug wurde der zukünftige Task „Berechnen und Darstellen der Aufenthaltsdauer pro Raum“ bereits in diesem Sprint umgesetzt.

14.3 Statistikklassen

Die Statistikklassen haben sich im Vergleich zum vorherigen Sprint stark verändert. Damals ist davon ausgegangen worden, dass sich jedes Fenster eine fixe Menge an berechneten Daten teilt und diese über den StatistikManager abrufbar ist. Nach ersten Analysen hat sich jedoch gezeigt, dass es eine gemeinsame Datenmenge zwischen den Fenstern gar nicht gibt. Denn sobald in einem Fenster eine Filtereinstellung getroffen wird, basiert diese auf völlig anderen Werten als die restlichen Fenster. Das Resultat dieser Erkenntnisse war, dass es den StatistikManager gar nicht benötigt, und jedes Fenster selber eine Instanz mit all den berechneten Daten direkt führt.

14.3.1 Analyse

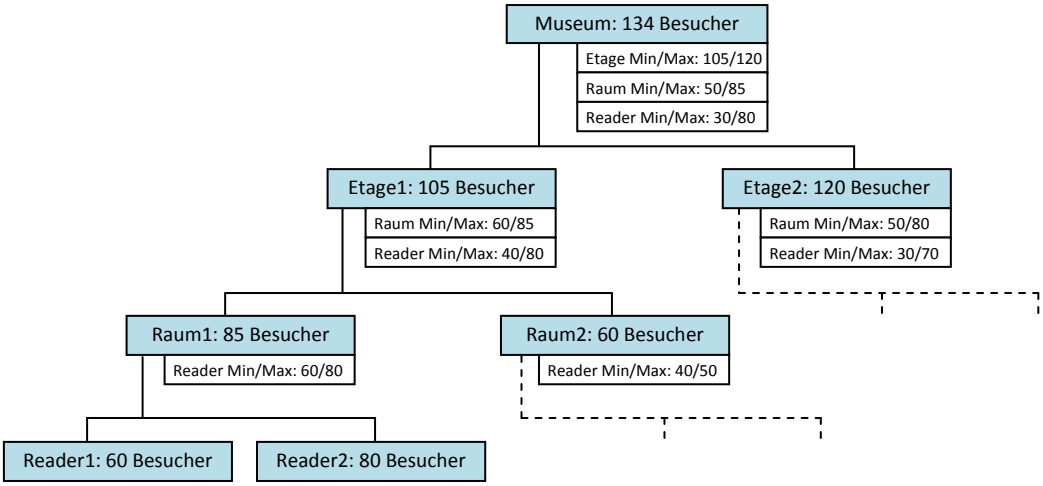
Ziel der Analyse war es herauszufinden, wie Erfassungen zeitlich gefiltert werden, relative Werte berechnet und die Aufenthaltsdauer bestimmt.

14.3.1.1 Filterung

Das aktuelle Problemdomain verfügt nicht über die Assoziationen, welche für die Statistiken benötigt werden. Es ist zum Beispiel nicht möglich, zu einem Reader direkt dessen erfasste Besucher zu erfragen. Alle Besucher werden in einer Liste auf der Stufe Museumszustand gehalten. Diese Liste muss zuerst aufwendig durch Iterieren mit den Readern verknüpft werden, um die gewünschte Verbindung herzustellen. Bei der Filterung wird dieser Prozess zusätzlich komplexer und es stellte sich die Frage, ob nicht von Beginn an mit SQL oder einem ORM gearbeitet werden soll. Mit SQL oder einem ORM entsprächen diese komplizierten Vernetzungen lediglich einer Query, welche mit zusätzlichen WHERE Bedingungen im selben Schritt Filterung erlauben würde. Diskussionen darüber, wie das bestehende Problemdomain zu erweitern wäre um eine Filterung zu realisieren, führten allerdings zum Ergebnis, dass eine Lösung nicht über SQL oder einen ORM realisiert werden sollte. Der Aufwand das bestehende Problemdomain mit einem ORM zu integrieren oder direkt SQL Abfragen zu erlauben schien grösser, als der Aufwand das Problemdomain für unsere Zwecke anzupassen.

14.3.1.2 Relative Werte

Die relativen Werte werden einerseits für die Einfärbung und andererseits bei der Anzeige in Prozent benötigt. In dem gewählten Einfärbungsmodell entspricht die minimalste Erfassung beispielsweise der Farbe Rot (0%) und die maximalste Erfassung der Farbe Grün (100%). Es muss somit bestimmt werden können, welches diese minimalsten bzw. maximalsten Erfassungen sind. Dabei soll jede Ebene in der Museumshierarchie die Minima und Maxima seiner darunterliegenden Ebenen kennen.



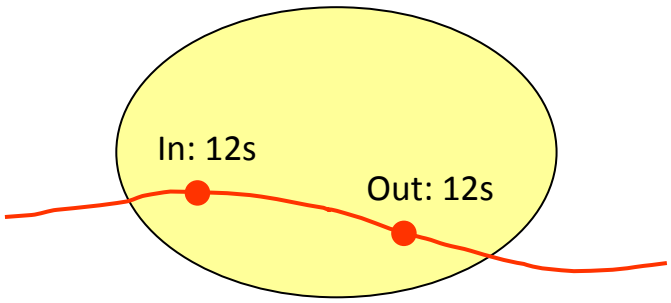
Die Minima und Maxima werden dabei rekursiv an die nächste höhere Ebene hochpropagiert, bis die höchste Ebene, das Museum, alle Schlüsselwerte kennt.

14.3.1.3 Aufenthaltsdauer

Für die korrekte Berechnung der Aufenthaltsdauer müssen mehrere Situationen berücksichtigt werden. Nachfolgend sind diese genauer analysiert worden:

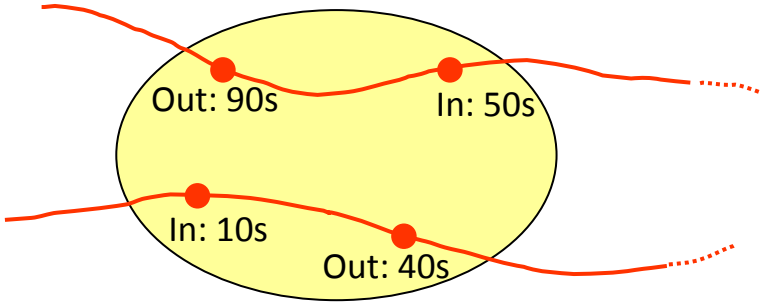
14.3.1.3.1 Besucher mit einer Aufenthaltsdauer von 0 Sekunden

Es kann passieren, dass ein Besucher nur ein einziges Mal in einem Reader erfasst wurde. Die Datenbereinigung macht daraus zwei künstliche Erfassungen, jedoch mit derselben erfassten Zeit. Die daraus resultierende Aufenthaltsdauer von 0 Sekunden wird nicht für die Berechnung berücksichtigt. Auf diese Weise wird die durchschnittliche Aufenthaltsdauer durch solche Null-Werte nicht verfälscht.



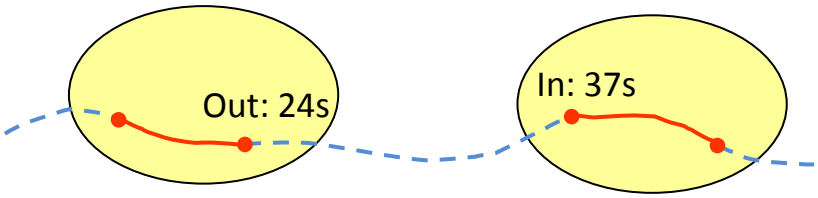
14.3.1.3.2 Besucher wird mehrmals im selben Reader erfasst

Wird derselbe Besucher in einem Reader mehrmals erfasst, wird die Aufenthaltsdauer durch die Aufsummierung dessen gemessenen Aufenthaltsdauer berechnet. Das heisst für die Auswertung spielt es keine Rolle, ob sich der Besucher einmal 70 Sekunden, oder in zwei Teilen 30 und 40 Sekunden im Reader aufgehalten hat.



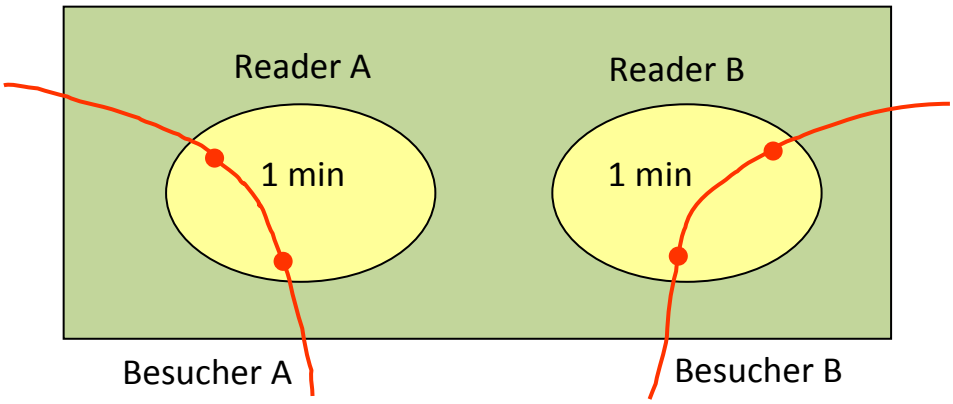
14.3.1.3.3 Zeit von einem Reader zum nächsten

Über die Zeit, welche ein Besucher von einem Reader zum Nächsten benötigt, kann keine genaue Aussage gemacht werden. Aus diesem Grund wird diese Dauer ignoriert und auch nicht versucht der Aufenthaltsdauer des umschliessenden Raumes anzurechnen. In die Berechnung fliessen nur die tatsächlich erfassten Aufenthaltsdauer mit ein.



14.3.1.3.4 Zeit pro Raum / Etage / Museum

Für die Berechnung der Aufenthaltsdauer pro Raum / Etage / Museum kann nicht einfach auf die berechneten Zeiten der Reader zurückgegriffen werden. Beispielsweise kann es sein, dass sich ein Besucher A eine Minute lang im Reader A aufhält, den Raum verlässt und sich ein Besucher B ebenfalls nur eine Minute in Reader B aufhält. Die resultierende durchschnittliche Aufenthaltsdauer im Raum ist folglich eine Minute, und nicht etwa zwei Minuten. Es muss auf Grund dessen für jede Ebene die Aufenthaltsdauer einzeln berechnet werden.



14.3.2 Design

Einstiegspunkt für die statistischen Daten ist neu die MuseumStateStatistic Klasse. Ausgehend vom momentan gesetzten MuseumState in WindowSettings erstellt sich jedes Statistikfenster eine Instanz dieser Klasse, welche alle statistischen Berechnungen vornimmt.

14.3.2.1 Klassendiagramm

Abbildung 43: Klassendiagramm über StatisticUnit

14.3.2.2 Beschreibung der Statistikklassen

Die Struktur des Museums wurde mit folgenden Klassen abgebildet, wobei hier die oberste Ebene der Museumsstruktur die MuseumStateStatistic darstellt. Sie besitzt eine Menge von FloorStatistic, welche aus einer Menge von RoomStatistic besteht. Zuletzt enthalten die RoomStatistic die Menge der TrackingDeviceStatistic, welche die unterste Schicht der Museumsstruktur ausmachen.

14.3.2.2.1 StatisticUnit

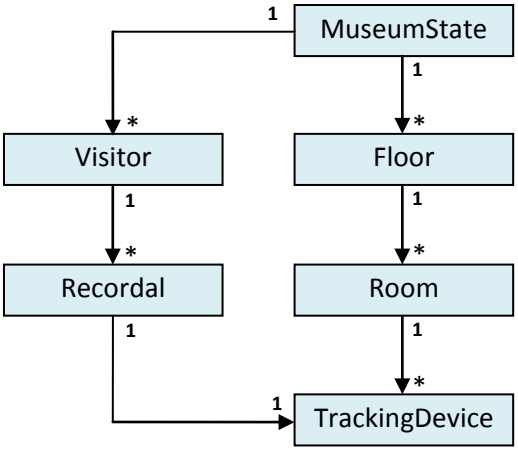
Alle Statistikklassen haben gemeinsame Eigenschaften, welche in der Klasse StatisticUnit zusammengefasst wurden. Über die Schnittstellen der StatisticUnit kann die Anzahl der Besucher, Angaben zur Aufenthaltsdauer und die Minima / Maxima der darunterliegenden StatisticUnits abgefragt werden.

14.3.2.2.2 MuseumStateStatistic

Ausgehend von einem gegebenen MuseumState berechnet die MuseumStateStatistic alle statistischen Daten. Der genaue Ablauf wird nachfolgend erklärt.

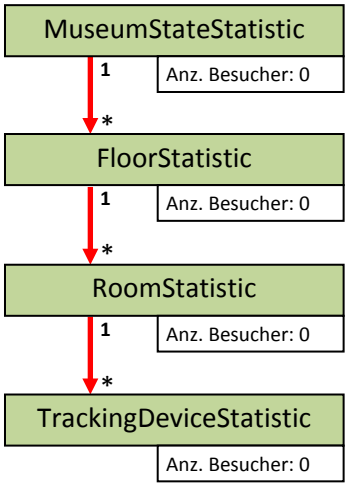
Ausgangslage Problem Domain:

Wie bereits erwähnt besteht ein MuseumState aus beliebig vielen Floors, die Floors aus beliebig vielen Rooms und die Rooms aus mehreren TrackingDevices. Die MuseumState besitzt als einzige Klasse eine Liste aller Besucher. Ein Besucher enthält die von ihm erzeugten Messdaten (Erfassungen). Es geht nun darum, diese Erfassungen mit den unterschiedlichen Ebenen des Museums zu verknüpfen, um statistische Aussagen machen zu können.



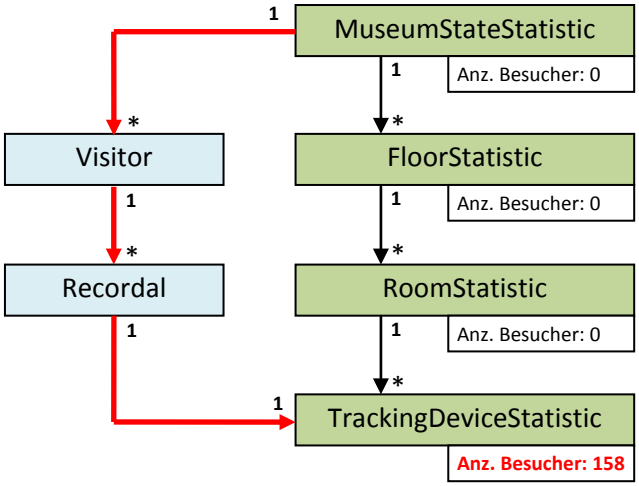
14.3.2.2.2.1 Initialisierung der Museums Struktur

Als erstes wird bei der Initialisierung die Gesamte Museums-Struktur erfasst. Das heisst, ausgehend vom gegebenen MuseumState wird über alle Etagen, Räume und Reader iteriert und leere StatisticUnits erstellt. Leer deshalb, weil diese noch mit keinen Besucherinformationen verknüpft sind.



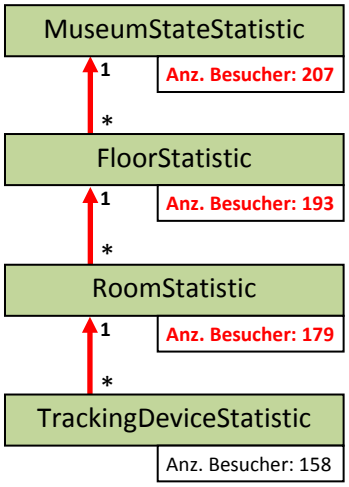
14.3.2.2.2.2 Ablaufen der Besucher Erfassungen

In einem zweiten Schritt werden über alle Besucher und deren Erfassungen iteriert. Die leeren TrackingDeviceStatistics, welche im ersten Schritt bereits erstellt wurden, können nun mit den Informationen aus den Erfassungen abgefüllt werden. Nach diesem Schritt wissen alle TrackingDevice-Statistics, welche Besucher sie erfasst haben und wie lange sich diese in ihnen aufhielten.



14.3.2.2.2.3 Hochpropagieren der Daten

Das Hochpropagieren der Daten aus den TrackingDeviceStatistics, welche die unterste Ebene der Museumsstruktur darstellen, erfolgt rekursiv. Ein Raum füllt somit seine Besucherdaten (Aufenthaltsdauer Zeiten und Minima / Maxima) basierend von seinen Readern (TrackingDevice), die Etage von seinen Räumen und schliesslich das Museum von seinen Etagen.



14.3.2.2.2.4 BemerkungBesucherzählung

Die Besucheranzahl der erstellten MuseumStateStatistic kann kleiner sein als die Anzahl Besucher in der Liste des zugrundeliegenden MuseumStates. Grund dafür ist, dass der MuseumState auch Besucher enthält, welche gar keine Erfassungen besitzen, und somit in der Statistik nicht berücksichtigt werden.

14.3.2.2.3 TrackingDeviceStatistic, RoomStatistic, FloorStatistic

Die restlichen Statistikklassen unterscheiden sich nur minim. Es wäre ebenfalls möglich die StatisticUnit-Klasse mit generischen Parametern zu versehen und diese drei Klassen zu entfernen. Es wäre jedoch im Allgemeinen umständlicher mit einem StatisticUnit<Room, TrackingDevice> zu arbeiten, als mit einer RoomStatistic.

14.3.3 Qualitätskontrolle

14.3.3.1 Unit Tests

Um die Korrektheit der berechneten Daten sicherzustellen, wurde ein geeignetes Testmuseum erstellt. Nachfolgend wird die Struktur dieses Testmuseums genauer erklärt, um einen Eindruck der Testabdeckung zu bieten. Das Testmuseum wird von Sprint zu Sprint erweitert, um die Testbarkeit neuer Funktionen zu reflektieren.

14.3.3.1.1 Testmuseum

Das Testmuseum besteht aus vier Etagen, wobei jede ihre Eigenheiten besitzt, welche eine mögliche Schwachstelle im Datensammelprozess ausleuchtet. Die erste Etage entspricht einer „gewöhnlichen“ Etage mit ein paar Besuchern, welche versuchen alle in der Analyse aufgedeckten Besonderheiten abzudecken. Die zweite Etage besitzt zwar Räume und Reader, enthält jedoch keine Besucher. Die dritte Etage ist etwas degeneriert, das heisst sie enthält Räume ohne Reader, Räume mit Readern aber nur einem Besucher. Diese und weitere Eigenheiten werden mit entsprechenden Testcases erfolgreich abgedeckt.

14.3.3.2 Leistung

Das Erfassen der statistischen Daten soll möglichst performant sein. Für den Benutzer ist es sehr unangenehm mehrere Minuten auf sein Ergebnis warten zu müssen. Es wurde deshalb viel Wert darauf gelegt, die Berechnung der statistischen Daten zu optimieren. Das Erstellen einer MuseumStateStatistic läuft in *O(Anz. Etagen + Anz. Räume + Anz. Reader + Anz. Besucher + Anz. Erfassungen)* ab, was einem linearen Wachstum entspricht. Messungen festigten diese theoretische Kalkulation:

14.3.3.2.1 Messungen

Hardware: Intel Core Duo 2.66GHz, 4GB Ram

Anzahl Datensätze: über 12‘000

Zeit für das Erstellen der MuseumStateStatistic-Klasse: ca. 20ms

14.4 Visualisierungen

14.4.1 Analyse

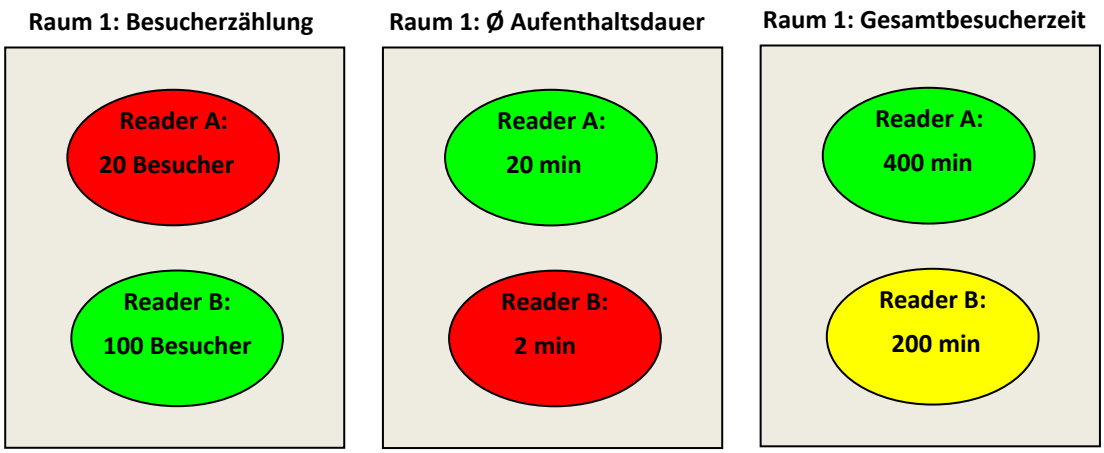
Nach der Review-Sitzung wurde das Einfärbungsmodell (minimale Erfassung in rot mit Farbverlauf bis zur maximalen Erfassung in grün) für gut geheissen.

14.4.1.1 Gesamtübersicht

Zusätzlich wurde das Total der Besucher auf Stufe "ganzes Museum" und "pro Etage" realisiert. Die Alternative, dass diese Anzeige direkt im Fensterpanel angezeigt wird, ist verworfen worden, da sie auf der Seite in einem Seitenpanel (Dockable) als genügend übersichtlich erachtet wurde.

14.4.1.2 Visualisation: Gesamtbesucherzeit

Die Visualisierungen der Besucheranzahl und durchschnittliche Aufenthaltsdauer liefern im Einzelnen nicht immer aussagekräftige Werte. Folgende Ausganglage soll betrachtet werden:



Beim Begutachten der Visualisierung der Besucherzählung macht Reader A gegenüber Reader B den Anschein „schlechter besucht“ zu sein. Wird hingegen auf die Visualisierung Aufenthaltsdauer gewechselt, ist das Resultat der Betrachtung gerade verdreht. Um nun eine aussagekräftigere Visualisierung anzubieten, wurde die Gesamtbesucherzeit eingeführt. Sie verbindet beide Visualisierungen miteinander, um auf einen Blick zu sehen, welche Reader oder Räume „am Intensivsten“ besucht wurden. Mathematisch gesehen werden alle gemessenen Aufenthaltsdauern pro Raum / Reader aufsummiert.

14.4.2 Design

Da ein Design ständigen Umstrukturierungen und Anpassungen unterlegen ist, zeigt dieses Kapitel die bis dato aktuellste Version der Visualisierungen. Hauptsächlich neu in diesem Sprint sind die Visualisierung der Aufenthaltsdauer und der Gesamtbesuchszeit.

14.4.2.1 Aufenthaltsdauer

Angezeigt wird die durchschnittliche, die geringste und höchste Aufenthaltsdauer der Besucher pro Reader oder Raum.

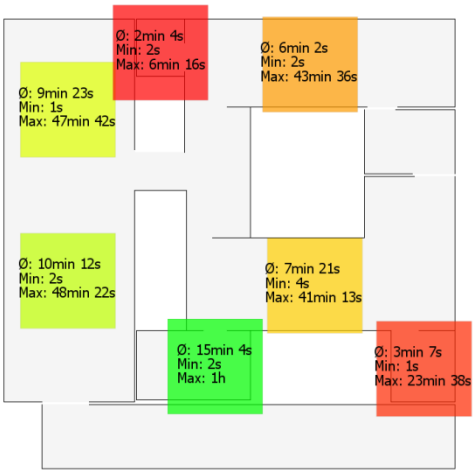


Abbildung 44: Aufenthaltsdauer pro Reader

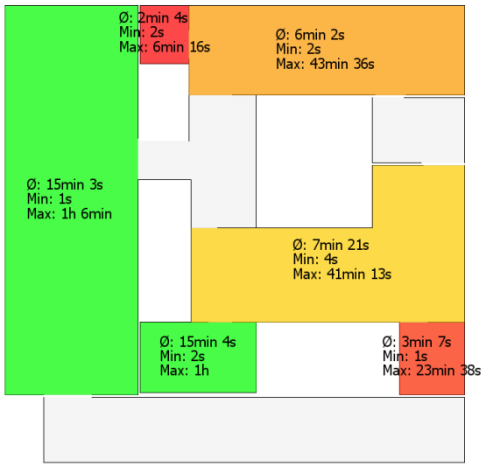


Abbildung 45: Aufenthaltsdauer pro Raum

14.4.2.2 Gesamtbesucherzeit

Angezeigt wird die total investierte Zeit pro Reader / Raum. Bei der Darstellung pro Raum wird nun klar ersichtlich, dass der Raum mit der Sonderausstellung im Vergleich zu den Nachbarräumen am Intensivsten besucht wurde.

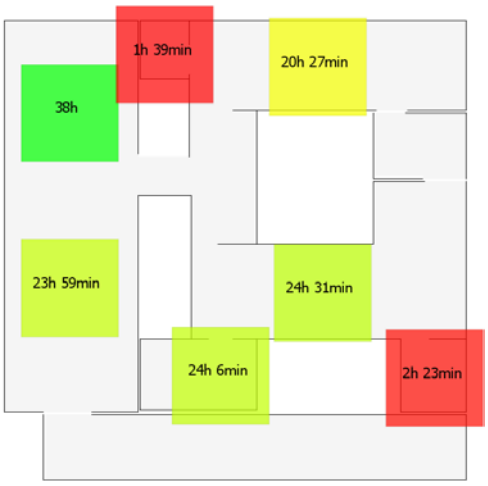


Abbildung 46: Verweilzeit pro Reader

Abbildung 47: Verweilzeit pro Raum

14.4.2.3 Daten-Tab

Das Datentab ist zuständig für die Auswahl der gewünschten Daten. Hier werden grundsätzliche Einstellungen (wie beispielsweise das Museum, Museumszustand und Etage) ausgewählt. In einem weiteren Dockable wird dem Benutzer ein Gesamtüberblick über die vorhandenen Museums/Etagen-Werte geboten. Im nächsten Sprint wird ein weiteres Dockable dazukommen, welches eine Filterung über Zeit, Besucherattribute und Gruppen zulässt.

14.4.2.3.1 Auswahl Museum / Museumszustand / Etage

Die Auswahl der einzelnen Museen, der Zustand und die Etage sind über mehrere Comboboxes auswählbar. Bei jeder Selektion werden automatisch alle auszuwählenden Werte neu geladen.

Abbildung 48: MuseumsChooser

14.4.2.3.2 Gesamtübersicht

Die Gesamtübersicht ermöglicht auf einen Blick alle wichtigen Grund- und Vergleichswerte. Eingeteilt wurde es in die Punkte im Museum und der aktuell ausgewählten Etage.

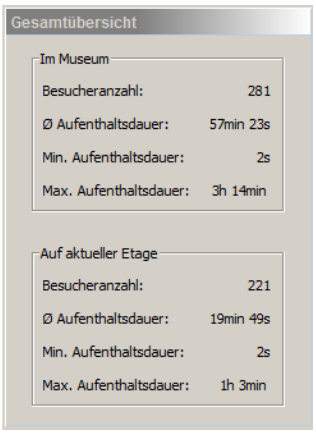


Abbildung 49: Übersicht der Daten

14.4.2.4 Visualisierungs-Tab

Das Visualisierungs-Tab ist das zweite Tab, welches anhand der vorherigen Einstellungen (siehe Kapitel Datentab) die gewünschte Visualisierung der Besucherzählung, der Aufenthaltsdauer oder der Gesamtbesucherzeit darstellt.

Die aktuelle Version enthält zwei grundsätzlich verschiedene Anzeigevarianten:

Abbildung 50: Visualisierungsart

14.4.2.4.1 Visualisierungsarten

Die Visualisierungen beziehen sich auf die Art der Datenauswertung.

Besucherzählung: Zeigt für jeden Raum / Reader die Anzahl Besucher an.

Aufenthaltsdauer: Durchschnittliche Zeit, die ein Besucher im Raum / Reader verbracht hat.

Gesamtbesuchszeit: Die für den Raum / Reader investierte (oder verbrachte) Gesamtzeit über alle Besucher.

14.4.2.4.2 Pro Raum / Reader

„Pro Raum / Reader“ zeigen die entsprechenden Visualisierungen pro Reader oder Raum.

Reader: Auswahl der Berechnungen auf Reader bezogen.

Raum: Alle Darstellungen werden auf Ebene Raum ausgeführt. (konkrete Beispiele sind im jeweiligen Kapitel ersichtlich)

14.4.3 Offene Punkte

Folgende Punkte sind bis dato noch nicht vollständig abgeklärt oder erarbeitet worden:

- Die Frage, ob evtl. ein verbesserter Algorithmus zur Einmittung benötigt wird
- Die Minima und Maxima der Aufenthaltsdauer sind wenig aussagekräftig. Evtl. wird in späteren Sprints die Möglichkeit erarbeitet, statt nur der Schlüsselwerte von Beginn an eine Verteilung der Aufenthaltsdauer pro Raum / Reader darzustellen.

15 Sprint 3

15.1 Änderungsgeschichte

Datum	Änderung
17.11.2009	Dokument erstellt.
20.11.2009	Beschreibungen der Museumsstatistic-Klasse hinzugefügt.
20.11.2009	Filterung der Attribute analysiert.
01.12.2009	Feedback-Anpassungen.
15.12.2009	Übernahme in Bericht-Dokument.

15.2 Beschreibung

Hauptziel des Sprints 3 war, die Filterung nach Zeit, Besucher Attributen und Gruppen fertigzustellen. Zusätzlich wurde das Anzeigen der Besucheranzahl in Prozent, welche eigentlich für den nächsten Task markiert war realisiert.

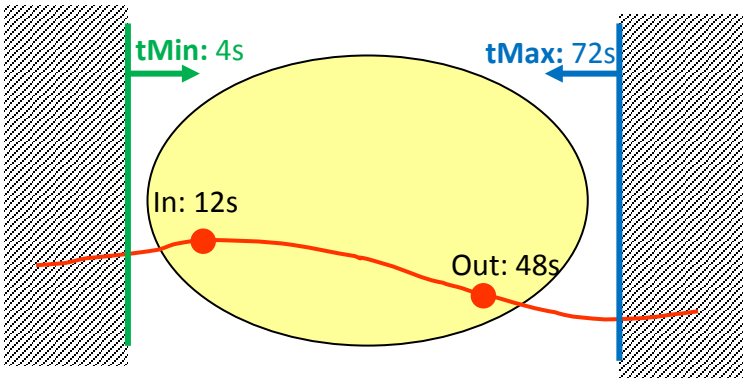
15.3 Filterung

15.3.1 Analyse

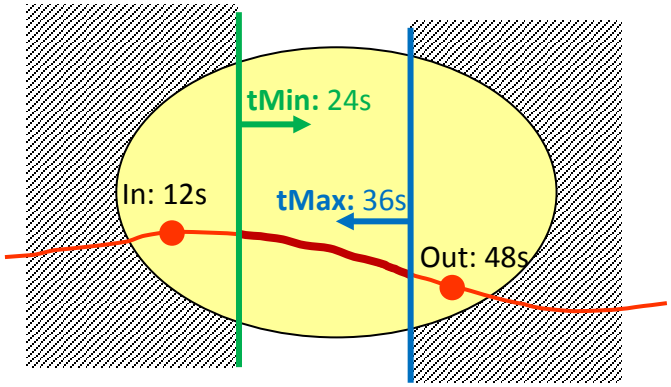
Mit welcher Architektur die Filterung gestaltet werden soll, wurde bereits im Sprint 2 analysiert. Aufbauend darauf werden hier die einzelnen Aspekte der Filterung genauer diskutiert.

15.3.1.1 Filterung nach Zeit

Aufgrund der Filterung nach Zeit wird vom Benutzer das Zeitintervall [tMin..tMax] gewählt und Erfassungen dementsprechend weggefiltert. Im Generellen bedeutet dies, dass Erfassungen, welche vor tMin oder nach tMax stattfanden, nicht mehr in die Zählung mit einfliessen.



Es gibt jedoch zwei symmetrische Spezialfälle (und deren Kombination davon), welche wie folgt berücksichtigt wurden:



Ist die gewählte untere oder obere Grenze des Zeitfilters genau innerhalb einer Erfassung, wird die erfasste Zeit an die Grenzen angepasst. Im dargestellten Beispiel würde somit die Besuchszeit von 48-12=36 Sekunden auf 36-24=12 Sekunden gekürzt. Liegen beide Grenzen exakt aufeinander, wird die Aufenthaltsdauer von 0 Sekunden nicht mehr berücksichtigt (siehe Sprint 2 unter „Besucher mit einer Aufenthaltsdauer von 0 Sekunden“).

15.3.1.2 Filterung nach Attributen

Den Besuchern wurden bei der Erfassung unterschiedliche Attribute zugeordnet. Diese wären beispielsweise demographische Eigenschaften wie Alter und Geschlecht, oder weitere Eigenschaften (z.B. mit / ohne Audioguide). Ein Attribut besteht aus zwei Teilen, einem Attributtyp und einer Menge von Attributwerten. Der Attributtyp beschreibt diese Werte, so besteht zum Beispiel der Typ „Alter“ aus den Werten {Kinder, Jugendliche, Erwachsene, Senioren}. Ein Besucher kann zu einem Typ nur genau einen Wert besitzen. Beispielsweise könnte ein gewöhnlicher Besucher folgende Attribute besitzen:

Geschlecht: *„Männlich“*
Alter: *„Erwachsen“*
Audioguide: *„mit Audioguide“*

15.3.1.3 Filterung nach Gruppen

Ein Besucher kann genau einer Gruppe angehören. Beispiele für Gruppen wären „Klasse 4g Wetzikon“, „Familie Müller“, etc. Der Filter soll es ermöglichen, bestimmte Gruppen weg zu filtern. Werden alle Gruppen bis auf eine herausgefiltert, kann die Auswertung isoliert auf diese Gruppe begutachtet werden.

15.3.2 Design

15.3.2.1 Klassendiagram

Abbildung 51: Beschreibung der Filterklasse

15.3.2.2 Beschreibung der Klassen

15.3.2.2.1 Filter

Die Filterklasse bietet ein einfaches Interface für das Festlegen, Abfragen, Prüfen und Zurücksetzten einer Filterung.

15.3.2.2.1.1 Festlegen und Abfragen der Filterung

Funktion	Beschreibung
void setMinDate(Date minDate); **Date getMinDate();**	Setzt die untere Grenze für die Filterung nach Zeit. Diese kann auch *null* sein, um nach unten offene Intervalle zu unterstützen.
void setMaxDate(Date maxDate); **Date getMaxDate();**	Setzt analog die obere Grenze für die Filterung nach Zeit.
void add(VisitorAttributeTypeValue excludedAttributeValue); **void remove(VisitorAttributeTypeValue excludedAttributeValue);** **Collection< VisitorAttributeTypeValue> getExcludedAttributeValues();**	Legt die weg zu filternden Attribute fest. Mit dem Getter kann die Liste der Attribute abgefragt werden, welche herausgefiltert werden.
void add(VisitorGroup excludedGroup); **void remove(VisitorGroup excludedGroup);** **Collection< VisitorGroup> getExcludedGroupes();**	Legt die weg zu filternden Besuchergruppen fest. Mit dem Getter wird die Liste der herauszufilternden Gruppen zurückgeliefert.

15.3.2.2.1.2 Filterprüfung

Funktion	Beschreibung
boolean passes(Visitor visitor);	Nur wenn der Besucher *visitor* keiner der im Filter angegeben Attributwerte oder Gruppen zugeordnet ist, wird *true* zurückgeliefert. *true* bedeutet, dass der Besucher in der Auswertung berücksichtig wird.
boolean passes(Date in, Date out);	Nur wenn sich das Zeitintervall von *in* bis *out* vollständig ausserhalb des festgelegten Filterintervalls befindet, wird *false* zurückgegeben. *false* bedeutet, dass die Erfassung nicht in der Auswertung berücksichtigt wird.

15.3.2.2.1.3 Weitere Funktionen

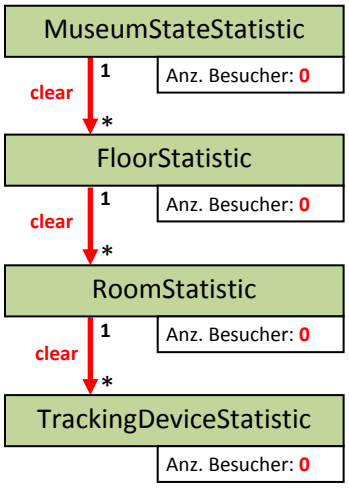
Funktion	Beschreibung
Date trimToMinDate(Date in); **Date trimToMaxDate(Date out);**	Das zurückgegebene Datum ist entweder *minDate* des Filters oder *in*, falls *in* grösser ist als *minDate*. Die Funktion liefert als Folge stets ein Datum, welches grösser oder gleich dem *minDate* des Filters entspricht. Das analoge gilt für *trimToMaxDate*. Diese Funktion wird im Spezialfall benötigt, welcher im Kapitel „Filterung nach Zeit“ besprochen wurde.
void reset(); **void resetTime();** **void resetExcludedAttributes();** **void resetExcludedGroups();**	Setzt den Filter entweder vollständig, oder teilweise zurück.

15.3.2.2.2 MuseumStateStatistic

Die MuseumSateStatistic besitzt neu eine Instanz der Klasse Filter. Über diese Instanz kann der Filter nach Belieben angepasst werden. Nach den gewünschten Einstellungen am Filter kann die Datensammlung neu angestossen werden. Der Ablauf wird nachfolgend genauer beschrieben.

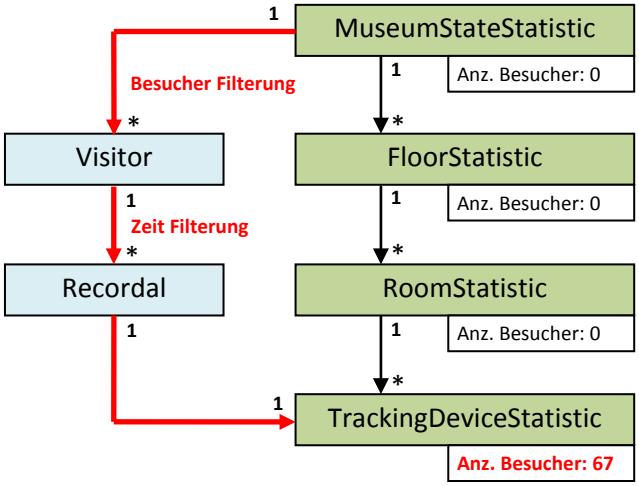
15.3.2.2.2.1 Zurücksetzten der vorherigen Datensammlung

Bevor eine neue Datensammlung stattfinden kann, müssen alle bisher gesammelten Daten zurückgesetzt werden. Da sich die Museumsstruktur nie verändert, werden die bestehenden Statistikklassen Instanzen recycelt, indem die bisherigen Werte zurückgesetzt werden.



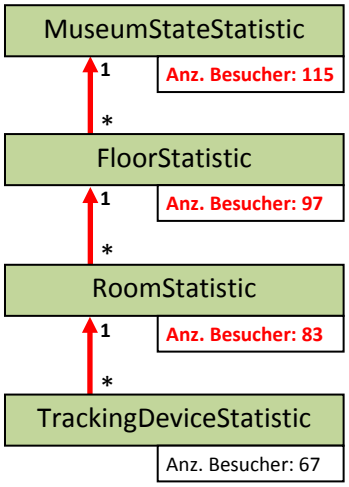
15.3.2.2.2.2 Ablaufen der Besucher Erfassungen

Es werden nur diejenigen Besucher betrachtet, welche nicht aufgrund deren Attributen und Gruppen weggefiltert wurden. Von den ungefilterten Besuchern werden weiter nur die Erfassungen berücksichtigt, welche sich im gewählten Zeitintervall des Filters befinden.



15.3.2.2.2.3 Hochpropagieren der Daten

Am Hochpropagieren der Daten hat sich gegenüber Sprint 2 nichts geändert.



15.4 Visualisierungen

15.4.1 Analyse

In diesem Sprint wurden alle Filter Paper Prototypes mit dem Productowner besprochen und gemeinsam verfeinert. Nachfolgend sind die Skizzen ersichtlich, welche wir vorgelegt und besprochen haben.

15.4.1.1 Filterung nach Zeit

Die bisherige Eingabemaske der Zeit hat sich als gut erwiesen und wurde beinahe unverändert übernommen, Grund dafür war die gute visuelle Übersicht. Optional wäre eine Validierung der Uhrzeit/Datum wünschenswert. Auch eine Auswahlmöglichkeit anhand eines „Datepickers“ würde die Bedienung stark verbessern.

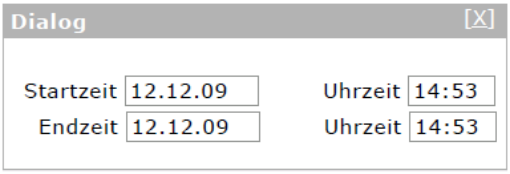


Abbildung 52: Prototype vom Datumpicker

15.4.1.2 Filterung nach Attributen

Jeder Attributtyp führt eine fixe Liste seiner Attributwerte. Für die Anzeige der Attribute wurde die Kombination von Titleborder und Checkboxen als am Geeignetsten befunden.

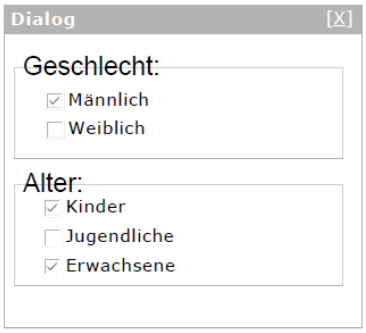


Abbildung 53: Prototype von Attributauswahl

15.4.1.3 Filterung nach Gruppen

Die Anzahl der Gruppen kann mit der Zeit zunehmen. Aus diesem Grund wird statt den einzelnen Checkboxen eine Liste mit Scrollbar verwendet. Wobei das Verhalten analog zur Attributauswahl ist.

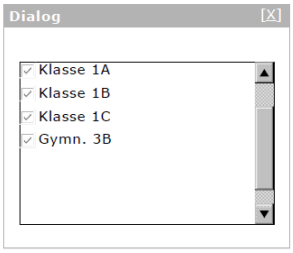


Abbildung 54: Prototype von Gruppenauswahl

15.4.1.4 Visualisierung der Besucherzählung: Absolute / Relative Werte

Bei der Umschaltung zwischen absoluten zu relativen Werten ändert sich nur die angezeigte Zahl von einem absoluten Wert in einen Prozentwert. Für die mathematische Berechnung dieser Prozentangaben gilt folgende Formel:

$$\text{prozentuale Anzahl} = \frac{absolute\ Anzahl}{maximale\ Anzahl\ im\ Museum}$$

15.4.2 Design

15.4.2.1 Filterung nach Zeit

Die Zeitauswahl wird über jeweils zwei Elemente gesteuert: Das Datum und die Uhrzeit. Für die Realisierung des Zeitintervalls, kann der Benutzer jeweils das Start- und Enddatum inklusive Uhrzeit auswählen.

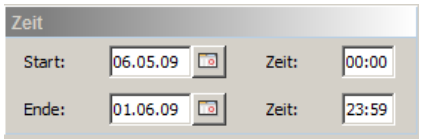


Abbildung 55: Implementierte Version von Datumauswahl

Um die Datumseingabe zu vereinfachen wurde eine externe Bibliothek eingesetzt, die eine kalenderartige Auswahl der Tage ermöglicht (siehe JCalendar).

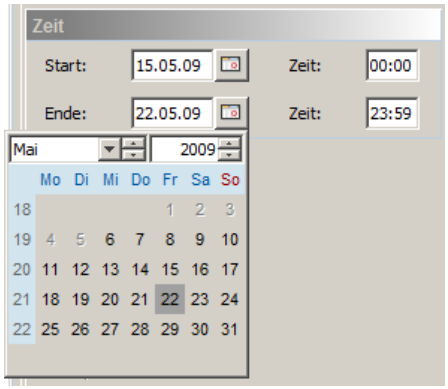


Abbildung 56: Datumauswahl mit erweiterten Eigenschaften

Die Auswahl der Uhrzeit wurde durch ein FormattedTextField implementiert, was eine Validierung der Eingabe ermöglicht. (Default ist HH:mm, was dem Intervall von 00:00 bis 23:59 entspricht.)

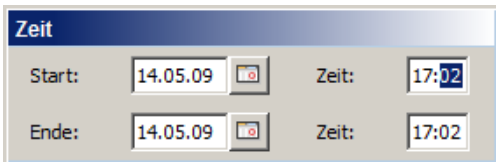


Abbildung 57: Verwendung von FormattedTextFields

15.4.2.1.1 Externe Library: JCalendar

JCalendar ist eine Bibliothek, die eine Datumsauswahl anhand eines grafischen Picking ermöglicht. JCalendar besteht aus verschiedenen anderen Klassen, so zum Beispiel dem JDayChooser, JMonthChooser und dem JYearChooser. All diese Klassen können individuell angesteuert und editiert werden. In der Applikation wird hauptsächlich JCalendar mit der Picking-Möglichkeit verwendet, sodass ein Datum auf eine einfache Art und Weise selektiert werden kann.

Die Library ist frei verfügbar und es werden keine Lizenzen benötigt. Für weitere Details ist folgender Verweis auf die dazugehörige Homepage[19] hilfreich.

15.4.2.1.2 Verhalten der automatischen Zeitanpassung

Die Filterung nach Zeit benötigen zwei Eingabefelder, einerseits das für den Start und andererseits für die Endzeit. Diese Werte dürfen dabei keine ungültigen Zeitintervalle zulassen. D.h. die ausgewählte untere Zeitschranke muss immer früher (oder gleich) der oberen Zeitschranke sein.

Um dieses Verhalten zu realisieren, wurde bei einer Zeitüberlappung folgendes angewendet.

Legende:

MinDate:	Datum der ersten Erfassung des Museums mit der Zeit 00:00:00
MaxDate:	Datum der letzten Erfassung des Museum mit der Zeit 23:59:00
SelectedMinDate:	Durch den Benutzer bestimmtes Startdatum (beschreibt untere Zeitschranke)
SelectedMaxDate:	Durch den Benutzer bestimmtes Enddatum (beschreibt obere Zeitschranke)

[19] http://www.toedter.com/en/jcalendar/api/com/toedter/calendar/package-summary.html, letzter Zugriff 19.Nov 2009

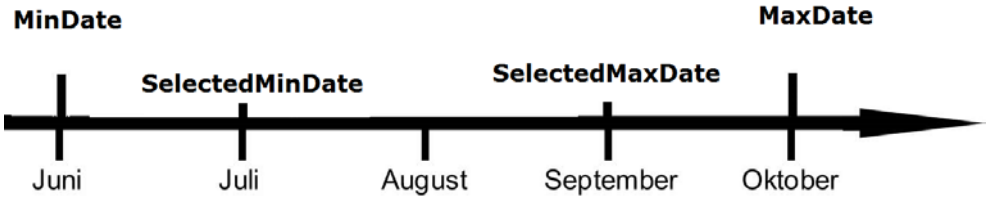


Abbildung 58: Gültige Zeitintervall

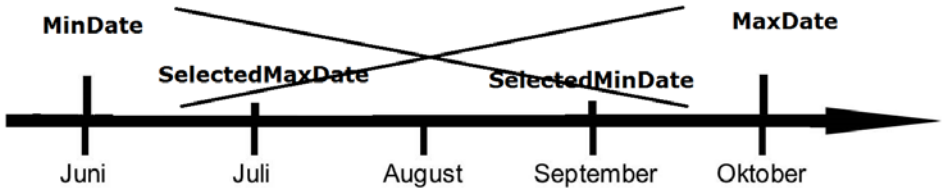
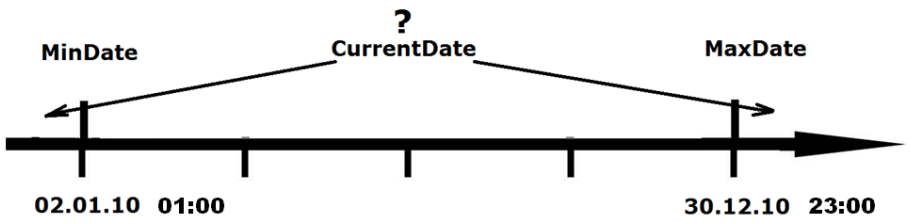


Abbildung 59: Ungültiges Zeitintervall

Bei dieser Darstellung gibt es eine Überlappung, was ein ungültiges Zeitintervall zur Folge hat (SelectedMaxDate < SelectedMinDate). Solche Sonderfälle werden nachfolgend beschrieben und ihre Behandlung detailliert erläutert.

Bei einer Überlappung wie in Abbildung 59: Ungültiges Zeitintervall zu sehen ist, wird beim Setzen von SelectedMaxDate der Wert des SelectedMinDate automatisch auf SelectedMaxDate gesetzt. Damit werden die ungültigen Intervalle abgefangen. Analog verhält sich die Applikation, falls der Benutzer SelectedMinDate nach dem SelectedMaxDate setzten würde.

Für den Fall, dass der Benutzer eine Filtergrenze CurrentDate über die Datumsgrenzen MinDate und MaxDate hinweg selektiert, gilt folgendes Verhalten:



CurrentDate		Beschreibung	Aktion / Verhalten
01.01.10	12:00	Neues Datum ist vor MinDate.	CurrentDate = MinDate
02.01.10	***01:00***	Exakt dieselbe Zeit und Datum.	Keine Anpassung
02.01.10	***00:50***	Derselbe Tag, aber die Zeit ist vor Min-Zeit.	CurrentDate = MinDate
31.12.10	12:00	Neues Datum ist nach MaxDate.	CurrentDate = MaxDate
30.12.10	***23:00***	Exakt dieselbe Zeit und Datum.	Keine Anpassung
30.12.10	***23:50***	Derselbe Tag, aber die Zeit ist nach Max-Zeit.	CurrentDate = MaxDate

15.4.2.2 Filterung nach Attributen

Da die Besucherattribute jeweils von der Datenbank geladen werden und somit erst zur Laufzeit bekannt sind, werden diese dynamisch in das Seitenpanel geladen. Beim De-/Selektieren einer

Checkbox, wird automatisch eine Filteranfrage gesendet, was eine Aktualisierung der dargestellten Visualisierung auslöst.

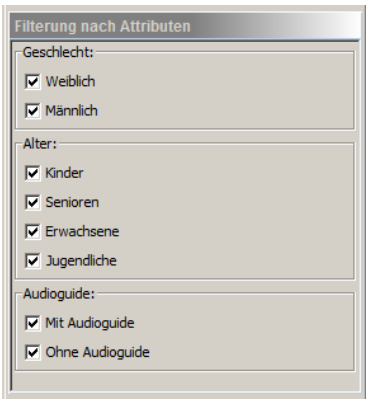


Abbildung 60: Filterung nach Attributen

15.4.2.3 Filterung nach Gruppen

Die verschiedenen Gruppen werden in einer Liste dargestellt, welche in ihrer ersten Spalte eine Checkbox beinhaltet und in der zweiten den Gruppennamen. Durch die Checkboxen können beliebige Gruppen an oder abgewählt werden.

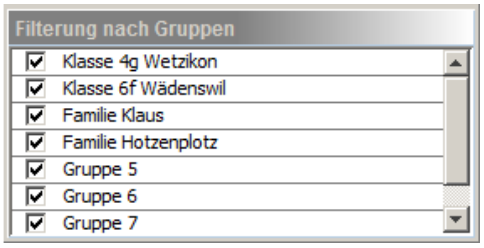
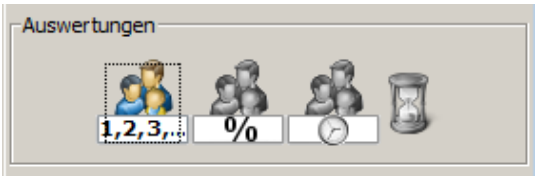


Abbildung 61: Gruppenfilterung mit Test-Werten

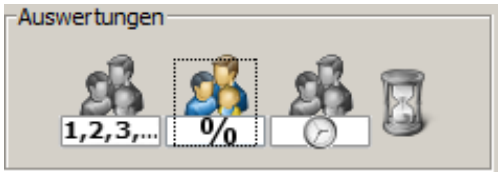
15.4.2.4 Absolute / Relative Werte

Wie auf den nachfolgenden Bildern zu sehen ist, ändert sich die farbige Untermalung des Readers nicht. Die Anzeige in relativen Zahlen wurde als eigene Visualisierung zu den bisherigen drei hinzugefügt.

Absolute Darstellung:



Prozentuale Darstellung:



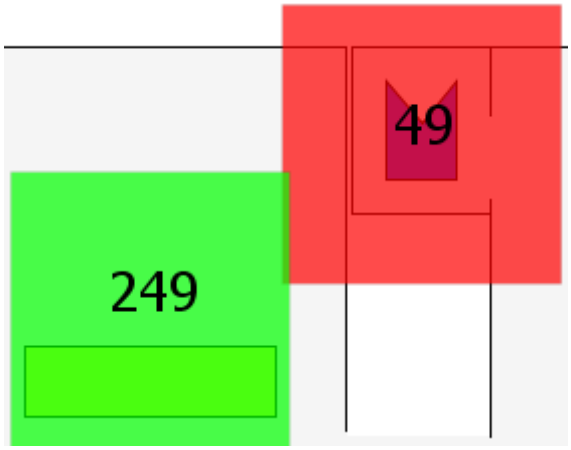


Abbildung 62: Absolute Darstellung

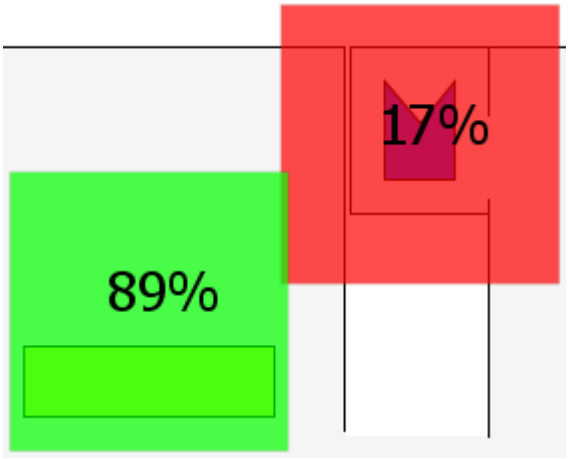


Abbildung 63: Relative Darstellung

15.4.3 Offene Punkte

- Evtl. sollte die Schrift mit weiss umrandet werden (schwarze Schrift auf dunklem Hintergrund ist schlecht lesbar)
- Evtl. sollte der Filtervorgang in einem eigenen Thread auslagert werden (für den Fall, dass der Filtervorgang bei sehr vielen Daten länger als 20ms dauert)
- Gruppen: evtl. eine Checkbox mit „ohne Gruppe“ / „alle Gruppen“ hinzufügen

16 Sprint 4

16.1 Änderungsgeschichte

Datum	Änderung
27.11.2009	Erstinitialisierung des Dokumentes.
28.11.2009	Analyse des Sliders & Farbauswahl hinzugefügt.
11.12.2009	„Extended“ Version der Diagramme hinzugefügt.
11.12.2009	Aufgrund von Feedback div. Kapitel verbessert.
15.11.2009	Umkopieren in den Bericht.

16.2 Beschreibung

Hauptziel des Sprint 4 war es, dass der Benutzer die Farbskalen einstellen kann. Dies wurde bisher vom System fix eingestellt und konnte nicht angepasst werden. Zusätzlich wurde analysiert, ob eine verbesserte Auswahl des Zeitbereiches Sinn macht und mit welcher Priorität diese umgesetzt werden sollte.

16.3 Auswahl des Zeitbereiches („extended“)

In diesem Sprint wurde die Analyse einer erweiterten Version zur Zeitauswahl vorgenommen, da die Möglichkeit besteht, dass diese im nächsten Sprint umgesetzt wird.

Ziel war es, durch ein Diagramm zusätzliche Informationen zu erhalten, welche in der aktuellen Realisierung nicht oder nur schwer ersichtlich sind. Beispielsweise hat der Benutzer keine Übersicht über die bisher gesammelten Daten pro Tag und kann sich somit kein Bild machen, wann genau viele oder interessante Zeitbereiche verfügbar wären.

Damit frühzeitige Prototypen möglich sind, wurde die Analyse in 3 Abstufungen vorgenommen, welche jeweils aufeinander aufbauen: einer Basic-, Advanced- und Professional-Version. Ein endgültiges Design oder eine Implementation dieser Analyse wird je nach Priorität vom Productowner in einem späteren Sprint realisiert.

16.3.1 Analyse

16.3.1.1 Bisherige Lösung

In der bisherigen Lösung ist nur der gültige Bereich sichtbar, jedoch nicht an welchem Tag viele, wenige oder gar keine Besucher das Museum besuchten. Der Benutzer muss nach dem „Try-And-Error“-Prinzip nach interessanten Zeitbereichen suchen, was nicht besonders benutzerfreundlich ist.

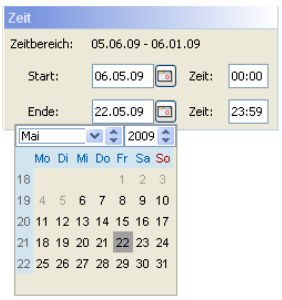


Abbildung 64: Bisherige Lösung

16.3.1.2 Basic Lösung

Ziel ist es, durch ein Diagramm zusätzliche Besucherinformationen zu erhalten, die sonst mühsam zusammengesucht werden müssten.

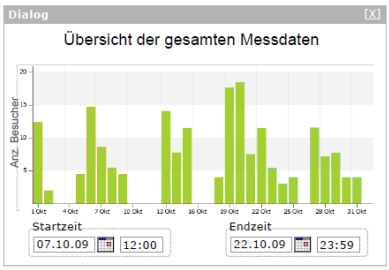


Abbildung 65: Erweiterte Lösung mit einem zusätzlichen Diagramm

- Besucher pro Stunde / Tag direkt ersichtlich.
- Verteilung der Besucher als Diagramm.
- Visualisierte Unterstützung der Datumsauswahl.
- Benutzer erkennt sofort wann wie viele Besucher erfasst wurden.

16.3.1.3 Advanced Version

Das aktuell gewählte Datum mit Uhrzeit wird zusätzlich visuell hervorgehoben. Das Diagramm soll interaktiv gehalten werden, das heisst, dass bei einem Klick auf einen einzelnen Balken der Tag ausgewählt wird.

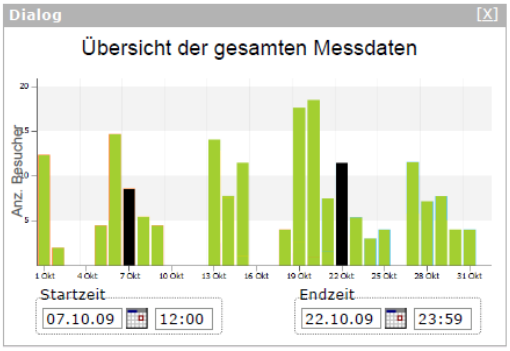


Abbildung 66: Übersicht von mehreren Tagen

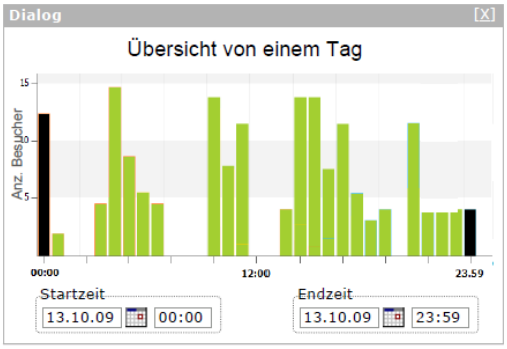


Abbildung 67: Auf einen Tag "gezoomt"

Durch einen Klick auf einen einzelnen Tag, wird in „Tagesübersicht“ gewechselt.

16.3.1.3.1 Zusätzliche Features

- Farbliche Untermalung der Datumsauswahl.
- Per Mouse-Click wird von Datums- in Tagesauswahl „gezoomt“.

16.3.1.4 Professional Lösung

Auswahl des Zeitbereiches ist nun zusätzlich per internen Slider möglich und die Anzeige unterstützt Tooltipps.

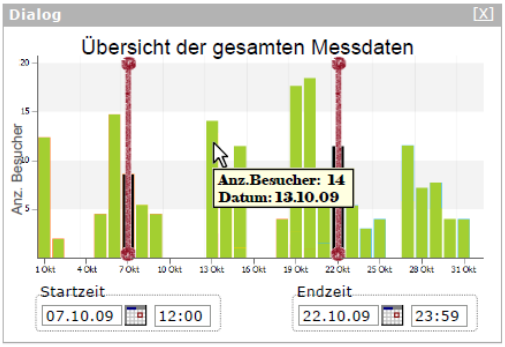


Abbildung 68: Professionelle Lösungsvariante

16.3.1.4.1 Zusätzliche Features

- Zusätzliche Auswahl des Zeitbereiches durch interne Slider möglich.
- Tooltipp im Diagramm bei den jeweiligen Balken.

16.3.1.5 Problemfall

Da die Anwendung in einem produktiven Einsatz sicherlich länger als nur einige Monate laufen wird, könnte es vorkommen, dass sich Datenmengen über Jahre hinweg ansammeln. Dies könnte zu einem Problem mit der grafischen Darstellung führen, da nicht jedes Diagramm für eine solche Datenmenge geeignet ist.

16.3.1.5.1 Bibliothekswahl

Um diesem Problem entgegenzuwirken, muss eine geeignete Diagramm-Bibliothek eruiert werden, die auch mit grossen Wertemengen (ca. 100‘000 Datenwerte) umgehen kann.

16.3.1.5.2 Bündelung der Messdaten

Eine alternative Lösung wäre, dass nicht jeder einzelne Punkt verwendet werden soll, sondern dass die Anzahl der dargestellten Punkte im Diagramm mit der Anzahl Messwerte skaliert. Mögliches Vorgehen wäre, dass nach folgendem Prinzip gehandelt wird: Je mehr Messwerte vorhanden sind, desto mehr Werte werden zusammengefasst / gebündelt.

16.3.1.5.3 Abwägung

Primär soll die Bibliothekswahl unteranderem mit Hinblick auf die Verwendung von grossen Datenmengen getroffen werden. Falls das gewählte Diagramm (oder deren Bibliothek) trotzdem nicht mit dieser Anzahl an Werten umgehen kann, soll eine Bündelung der Daten stattfinden.

16.3.1.6 Auswertung der Analyse

Da die Diagramme sehr starke Ähnlichkeiten mit den globalen Statistiken aufweisen, welche die Aufgabe des Sprints 5 sind, werden zuerst diese realisiert. Folglich wird der Task bis zu diesem Zeitpunkt bei einer ersten Analyse belassen.

16.4 Farbauswahl

In der Farbauswahl wird die Farbe für die minimale bzw. maximale Erfassung festgelegt. In den Visualisierungen werden die Messwerte durch einen Farbverlauf zwischen diesen Farben visuell unterstützt. Aktuell wird eine minimale Erfassung mit roter und eine maximale Erfassung grüner Farbe visualisiert. Zusätzlich ist verlangt, dass die Einstellungen über Programmende hinaus persistent bleiben. Dies wird bereits durch die Text-Ressource „visual.properties“ realisiert.

16.4.1 Analyse

Bis anhin wurde die Einfärbung standardmässig von Rot über Gelb nach Grün benutzt. Die Farbe Gelb ergibt sich dabei nur, weil sie sich auf dem Lichtspektrum genau in der Mitte von Rot und Grün befindet. Eine andere Möglichkeit des Farbverlaufs wäre die Mischung über die Farbpigmente. Die Mischung von Rot und Grün ergäbe dabei Braun. Beide Varianten werden nachfolgend analysiert und diskutiert.

16.4.1.1 Farbverlauf mittels Lichtspektrum

In diesem Farbenmodell ist nur der Farbton (als Zahl im Bild erkennbar) entscheidend. Die Mischung aus der Farbe Gelb (60) und Blau (240) würde folglich über Grün (120) und Türkis (180) führen.

16.4.1.1.1 Vorteile

- Nur saubere Verläufe sind möglich (vermeidet unschöne Mischfarben).
- Weniger Einstellmöglichkeiten sind notwendig (nur 2 Farben sind anzugeben).

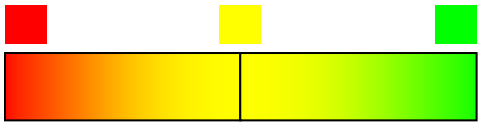
16.4.1.1.2 Nachteile

- Ein Farbverlauf ist nur mit gesättigten Farben möglich (z.B. kein Schwarz-Weiss Verlauf).

16.4.1.2 Farbverlauf mittels Farbpigmente

In diesem Farbenmodell wird mit den Pigmenten der Farben gearbeitet. Mathematisch betrachtet werden die Rot / Grün / Blau Werte zweier Farben gemittelt.

Farbverlauf über 2 Farben.



Farbverlauf über 3 Farben.

16.4.1.2.1 Vorteile

- Der Farbverlauf entspricht der direkten Mischung zweier Farben.
- Jeder Farbverlauf ist möglich (auch von Schwarz nach Weiss).
- Schwarz/Weiss ermöglicht guten Graustufendruck (mit Blick auf spätere Tasks).

16.4.1.2.2 Nachteile

- Es sind auch unschöne Farbverläufe (oft über Braungemisch) möglich.
- Es sollten mindestens 3 Farben angegeben werden (z.B. von Rot über Gelb zu Grün).

16.4.1.3 Auswertung der Analyse

Da der Verlauf über das Lichtspektrum bereits seit Anfang des Projektes benutzt wird und die zweite Variante keine nennenswerten Vorteile bringt, wird das bestehende Farbmodell beibehalten.

16.4.2 Design

Über das Einstellungsfenster kann die untere Farbe (für die minimale Erfassung) und obere (für die maximale Erfassung) geändert werden. Der resultierende Farbverlauf ist direkt sichtbar. Möchte der Benutzer nur den Farbverlauf umkehren, kann dies mittels „Farben tauschen“ erreicht werden. Für das Auswählen einer Farbe wurde der JColorChooser im Swing-Package benutzt.

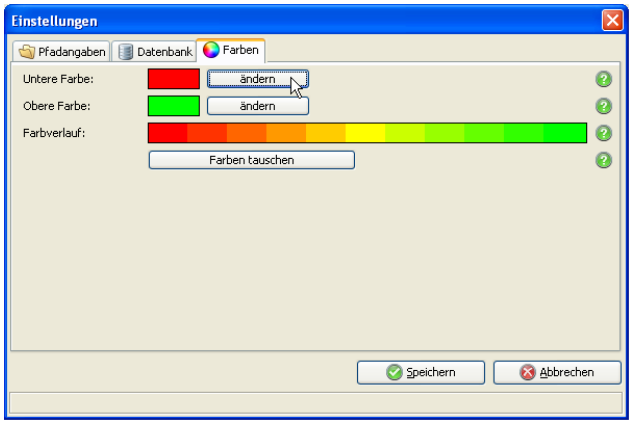


Abbildung 69: Aktuelle Farbauswahl

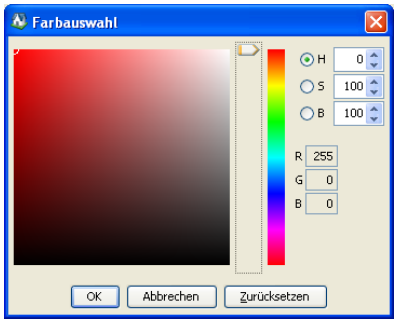


Abbildung 70: JColorChooser

16.4.3 Offene Punkte

- Beim Ändern der Farben wird vom Einstellungspanel jeweils ein Neustart der Anwendung verlangt, welcher prinzipiell nicht nötig wäre. Dies ist momentan so realisiert, da in demselben Panel auch einige Datenbankeinstellungen ausgeführt werden können, welche ein Neustart der Anwendung verlangt.

16.5 Farbskalen

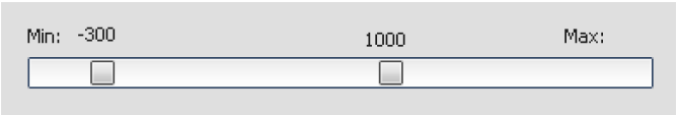
Bei den Farbskalen wird die im vorherigen Kapitel besprochene Einfärbung eingesetzt. Standardmässig entspricht die gewählte untere Farbe der minimalen Erfassung (z.B. Rot: 43 Besucher) und die obere Farbe der maximalen Erfassung (z.B. Grün: 234 Besucher). Dem Benutzer soll nun ermöglicht werden, diese Grenzen anzupassen (z.B. Rot: 0 Besucher und Grün: 300 Besucher).

16.5.1 Analyse

Bei der Realisierung der Farbskalierung wird ein Doppel-Slider verwendet, das heisst ein Slider mit zwei Buttons (sogenannte „Thumbs“). Dieser soll dem Benutzer die Möglichkeit bieten, seine gewünschte Farbskalierung durch ziehen der Grenzen einfach anzupassen. Nachfolgend werden unterschiedliche Varianten solcher Doppel-Slider analysiert.

16.5.1.1 Standard Doppel-Slider

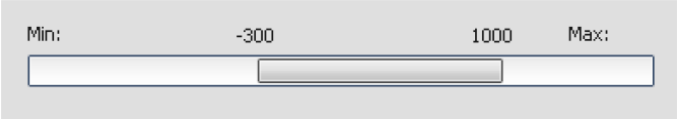
Ausgangslage und Grundlage der Analyse ist ein Slider mit zwei Thumbs. Das heisst ein Thumb für die untere und ein Thumb für die obere Grenze.



Doppelthumb mit unterer (-300) und oberer (1000) Grenze.

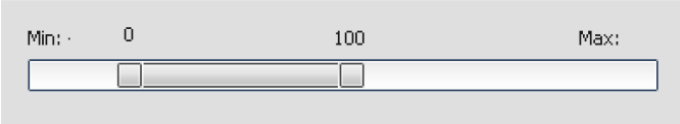
16.5.1.2 Verbundener Doppel-Slider

Bei dieser Variante ist die Fläche zwischen den Thumbs ausgefüllt. Folglich wäre beim Klicken auf die Fläche eine simultane Verschiebung beider Slidergrenzen möglich.



Doppelthumb mit verschmolzener Zwischenfläche.

Eine optische Verbesserung bietet die Version, bei der die Thumbs trotzdem noch sichtbar bleiben. Dadurch kann der Benutzer einfacher erkennen, dass die jeweiligen Enden einzeln verschiebbar bleiben.



Doppel-Slider mit zusätzlicher Visualisierung der Thumbs.

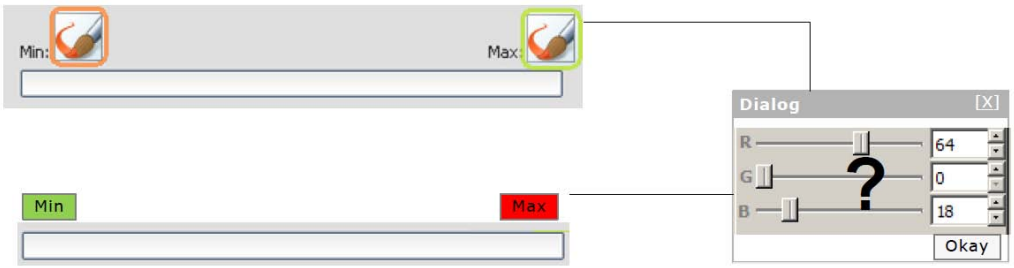
16.5.1.3 Farbverlauf Doppel-Slider

Eine weitere modifizierte Variante bietet der Doppel-Slider mit visueller Aufwertung durch einen Farbverlauf. Die Farbinterpolation findet zwischen dem unteren Thumb (A) und dem oberen Thumb (B) statt. Jeglicher Wert unter (A) wird mit der ersten Farbe dargestellt (hier Schwarz) und jeder Wert über (B) wird durch die zweite Farbe dargestellt (hier Gelb).

Doppel-Slider mit farblichem Verlauf der gewählten Farben.

16.5.1.4 Direkte Farbauswahl am Slider

Anstatt über das Einstellungspanel könnten die Farben direkt am Slider gewählt werden.



Beispielhafte Darstellung des Problems: Slider (ohne Thumbs) mit Anzeige durch Knopfdruck, das ein Farbauswahl-Dialog öffnet.

16.5.1.5 Auswertung der Analyse

Die einwandfreiste Visualisierung bietet ein Doppel-Slider mit Farbverlauf, da bei diesem auf einen Blick mehrere Einstellungen sichtbar wären. Diese Variante ist jedoch mit einem deutlichen Mehraufwand verbunden, da die Visualisierung keiner Standard-Komponente entspricht. Daher wäre es sinnvoll, zuerst eine schlichtere Variante als Prototype zu realisieren. Falls im Verlauf des Sprints noch Zeit zur Verfügung stehen würde, kann immer noch die gewünschte erweiterte Version

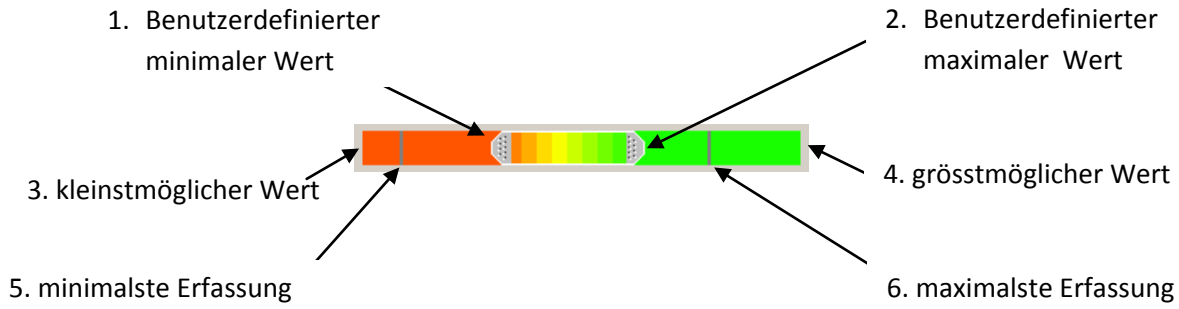
implementiert werden. Da der Benutzer in der Regel selten die Grenzfarben ändert, wird auf eine direkte Farbwahl am Slider verzichtet.

16.5.2 Design

Die Anforderungen an die Farbskalierung sind sehr spezifisch, demzufolge ist eine Benutzung eines gewöhnlichen JSliders von Java ungenügend. Folglich wurde für diese Anwendung ein eigener Slider entwickelt, welcher möglichst alle Anforderungen abdeckt.

Ausgehend von der Analyse sind folgende Anforderungen an den Slider bekannt: Der Doppel-Slider soll einen farblichen Verlauf enthalten, damit der Benutzer auf einfachste Weise seinen gewünschten Anzeigebereich einstellen und diesen auch zugleich visuell erkennen kann. Aus dem Farbeinstellungs-Panel sind die Minimal- und Maximalfarbe bereits bekannt (siehe Kapitel Farbauswahl). Die Applikation interpoliert zwischen diesen zwei Farben entlang des Lichtspektrums (Regenbogenverlauf). Im unteren Beispiel wurden die Minimalfarbe „Rot“ und „Grün“ als Maximalfarbe gewählt. Der Farbverlauf kann nun anhand des Sliders angepasst werden, wobei eine iterative Abstufung zwischen Rot und Grün stattfindet.

16.5.2.1 Beschreibung des Sliders

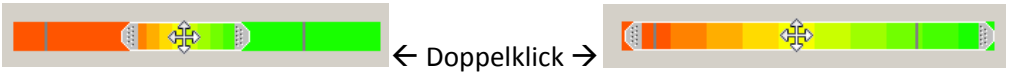


Position:	Beschreibung:
1. Benutzerdefinierter minimaler Wert	Der Slider besteht aus zwei verschiedenen Thumbs (Drag&Drop Knöpfe). Der linke Thumb legt dabei den Wert fest, bis zu welchem Punkt auf dem Slider rot eingefärbt werden soll.
2. Benutzerdefinierter maximaler Wert	Analog zum linken Thumb, legt der rechte Thumb den Wert fest, ab welchem grün eingefärbt wird.
3. kleinstmöglicher Wert	Der kleinstmögliche Wert ist prinzipiell immer 0: • Bei Zeitmessungen: 0s. • Bei Prozent: 0%. • Bei der Besucheranzahl: 0 Besucher Der Wert entspricht dem kleinstmöglichen zu wählenden Wert, dieser wird standardmässig immer auf 0 gesetzt.
4. grösstmöglicher Wert	Der grösstmögliche Wert wird wie folgt definiert: • Bei Zeitmessungen: Grösste gemessene Zeit. • Bei Prozent: 100%. • Bei der Besucheranzahl: Die Gesamtbesucheranzahl im Museum. Der Wert beschreibt einen grösstmöglichen zu wählenden Wert, dieser wird von der Anwendung selber festgelegt.
5. minimalste Erfassung	Der feine Strich im Slider unterstützt den Benutzer beim

	Einstellen seines Bereiches. Dieser zeigt die minimalste Datenerfassung an, das heisst es zeigt den ersten (oder kleinsten) Messwert an. Beispielsweise die kleinste erfasste Besucheranzahl eines Trackingdevices in der aktuellen Etage.
6. maximalste Erfassung	Analog zur ersten Datenerfassung, entspricht diese feine Linie der maximalsten Erfassung, also beispielsweise der längsten Aufenthaltsdauer einer Person auf der aktuellen Etage.

16.5.2.2 Verhalten des Sliders

Auf der Sliderfläche (d.h. zwischen kleinstmöglichem und grösstmöglichem Wert) kann der Slider durch Drag und Drop verschoben werden. Zusätzlich löst ein Doppelklick eine des Sliders auf den gesamten Bereich aus. Durch einen weiteren Doppelklickt, stellt sich wieder die vorherige Position her.



Analog zur Verschiebung des ganzen Bereiches, verändert sich der Mauszeiger über dem linken/rechten Tumb entsprechend.

Der Mauszeiger verändert sich sobald die Maus über einem Thumb liegt.

16.5.2.3 Aussehen des Sliders je nach gewählter Visualisierung

Je nach gewählter Visualisierung (absolute und relative Besucherzählung, Aufenthaltsdauer, Gesamtbesuchszeit) ändern sich die Schlüsselwerte des Sliders.

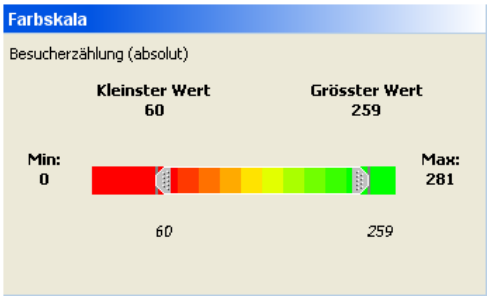


Abbildung 71: Besucherzählung (absolut)

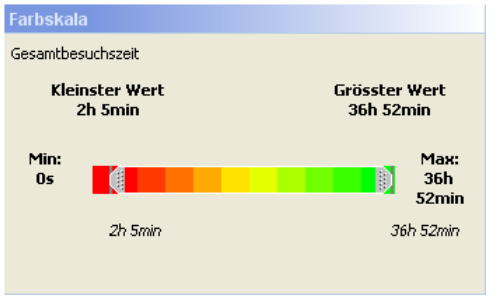


Abbildung 72: Gesamtbesuchszeit

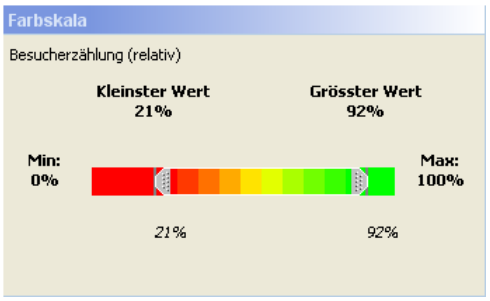


Abbildung 73: Besucherzählung (relativ)

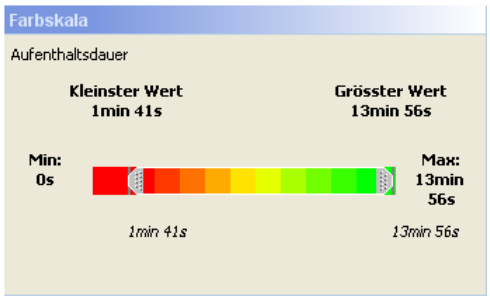


Abbildung 74: Aufenthaltsdauer (Verweildauer)

16.5.3 Offene Punkte

- Farbskalen müssen für alle Fenster übernehmen werden, damit nicht in jedem Fenster im Einzelnen die Farbeinstellungen angeglichen werden muss.

17 Sprint 5

17.1 Änderungsgeschichte

Datum	Änderung
07.12.2009	Ersterstellung.
08.12.2009	Analyse der JFreeChart hinzugefügt.
08.12.2009	Grosse Teile im Bereich des Problemdomains ausdokumentiert.
15.12.2009	Übernahme in Hauptbericht.

17.2 Beschreibung

Hauptziel im Sprint 5 war es, Statistiken über ein ganzes Museum darzustellen. Bis anhin wurden nur Auswertungen über eine einzelne Etage visualisiert. Wie bei der Etagenvisualisierung soll der Benutzer die Möglichkeit haben, den dargestellten Zeitbereich einzuschränken.

17.3 Museumsstatistik - Datensammlung

Für die Visualisierung einer Museumsauswertung müssen zusätzliche Werte berechnet werden. In den folgenden Kapiteln wird diskutiert, welche Werte dies sind und wie sie berechnet werden sollen.

17.3.1 Analyse

In den vorherigen Sprint Dokumentationen wurde die Klasse MuseumStateStatistic bereits genauer beschrieben. Zusammenfassend berechnet diese Klasse die Besucheranzahl und die Aufenthaltsdauer pro Tracking Device / Raum / Etage und dem Museumszustand selber. Da ein Museum aus beliebig vielen Museumszuständen besteht, stellt sich die Frage, ob eine Statistik über ein ganzes Museum auf der bereits existierenden Klasse MuseumStateStatistic aufbauen kann. In der nachfolgenden Analyse werden nun die Unterschiede zu einer aufbauenden Lösung und einem eigenständigen Ansatz diskutiert.

17.3.1.1 Anforderungen an die Datenberechnung

Von den Protoypen der Museumvisualisierungen her ist bekannt, dass die Anzahl Besucher pro Tag berechnet werden soll. Im Hinblick auf den nächsten Sprint darf nicht vergessen werden, dass die Anzahl Besucher pro Attribut (z.B. „männlich“) ebenfalls bekannt sein soll. Eine weitere Anforderung ist die zeitliche Eingrenzung der Daten auf einen vom Benutzer festgelegten Zeitbereich.

Die Anforderungen noch einmal kurz zusammengefasst:

- Berechnen der gesamten Anzahl Besucher pro Tag.
- Berechnen der Anzahl Besucher pro Attribut pro Tag (im nächsten Sprint).
- Berechnungen beschränken auf ein festgelegtes Zeitintervall.

Die Klasse MuseumStateStatistic berechnet nur die Anzahl Besucher pro Einheit und nicht wie gefordert pro Tag. Eine Einheit steht hier für ein beliebiges Tracking Device, Raum oder Etage in (z.B. das Tracking Device mit der ID „7“ hat 46 Besucher erfasst). Zusätzlich wird die berechnete Besucherzahl nicht für jedes Attribut im Einzelnen sondern total geführt. Das heisst, es wäre nicht möglich abzufragen, wie viele „männliche“ Besucher im Zustand erfasst wurden. Einzig über eine geschickte Filterung pro Attribut und Zeitbereich könnten die geforderten Werte mit der MuseumStateStatistic berechnet werden. Dies wäre jedoch sehr ineffizient, da bei jeder

Filtereinstellung jeweils die gesamten Daten durchlaufen werden müssten. Zudem berechnet die Klasse weitere Werte, welche für die oben genannten Anforderungen keine Rolle spielen.

Fähigkeiten der bestehenden Klasse MuseumStateStatistic:

- Berechnen der gesamten Anzahl Besucher pro Tag (nur über Filterung).
- Berechnen der Anzahl Besucher mit bestimmtem Attribut pro Tag (nur über Filterung).
- Berechnungen beschränkt auf ein festgelegtes Zeitintervall.
- Filterung nach Attributen und Gruppen.
- Anzahl Besucher pro Einheit (Tracking Device / Raum / Etage / Zustand).
- Aufenthaltsdauer pro Einheit.
- Die Minima- / Maxima-Werte auf allen Ebenen.

Der Vergleich zwischen den gestellten Anforderungen und den verfügbaren Fähigkeiten der Klasse MuseumStateStatistic führt zum Schluss, dass ein eigenständiger Ansatz erarbeitet werden muss. Die MuseumStateStatistic bietet nur über Umwege die gewünschten Daten und berechnet Werte welche in der Museumsvisualisierung nicht benötigt werden. Ein eigenständiger Ansatz scheint effizienter und könnte die Anforderungen gezielter erfüllen.

17.3.1.2 Anforderungen an die Datenauswahl

In den Seitenpanels auf der rechten Seite der Applikation soll der Benutzer die Möglichkeit haben den Zeitbereich und zusätzliche Anzeigen pro Attribut (z.B. „mit Audioguide“) auszuwählen. Es muss somit gespeichert werden, welcher Zeitbereich und welche Daten bezüglich einer gewählten Menge von Attributen dargestellt werden soll.

Zusammengefasst werden folgende Fähigkeiten erwartet:

- Speichern eines Zeitbereiches
- Speichern einer Menge von Attributen

Dieses Interface wird von der Klasse Filter (ein Teil der MuseumStateStatistic) abgedeckt. Im Hinblick auf Wiederverwendung bestehender Seitenpanels, könnte die Filterklasse für die oben genannten Anforderungen recycelt werden. Wiederverwendet deshalb, weil im semantischen Sinne keine Daten weggefiltert, sondern die Filterklasse nur zum Speichern der Einstellungen verwendet würde. Mit Einstellungen ist damit gemeint, welcher Zeitbereich und welche Attribute als Kurven in der Museumsvisualisierung dargestellt werden sollen.

Fähigkeiten der bestehenden Klasse Filter:

- Speichern eines Zeitbereiches
- Speichern einer Menge von Attributen
- Speichern einer Menge von Gruppen
- Zurücksetzen des Filters
- Filterlogik

Der Vergleich zwischen den gestellten Anforderungen und den verfügbaren Fähigkeiten der Filterklasse führt zum Schluss, dass die Filterklasse für den gewünschten Zweck wiederverwendet werden kann.

17.3.2 Design

Die Analyse hatte ergeben, dass eine eigenständige Klasse für die Museumsstatistik erarbeitet werden muss und dass die Filterklasse in einem anderen Sinne Verwendung findet.

17.3.2.1 Klassendiagramm

Abbildung 75: Klassendiagramm der Filterung

17.3.2.2 MuseumStatistic

Ergebnis des Designs ist die MuseumStatistic-Klasse. Diese berechnet gezielt die Anzahl Besucher pro Tag, welche von der Visualisierung gewünscht wird. Die Aufteilung der Daten nach Attributen wird wie geplant erst im nächsten Sprint umgesetzt.

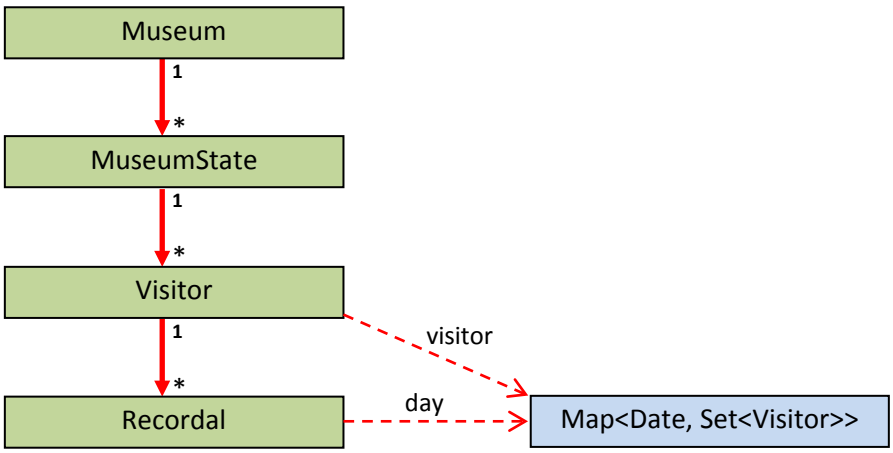
17.3.2.2.1 Ablauf der Datensammlung

Das Sammeln der Daten erfolgt durch eine einfache Iteration über alle Museumszustände des zugrundeliegenden Museums, welches im Konstruktor angegeben wird. Für jeden Zustand wird anschliessend über dessen Besucher und für jeden Besucher über dessen Erfassungen iteriert.

Aus den Erfassungen kann das Datum herausgelesen werden, an welchem sich der Besucher im Museum aufgehalten hatten. Über eine Hashmap aus Tag und Besuchermenge wird die resultierende Anzahl Besucher pro Tag gesammelt. Diese Hashmap trägt nach der Datensammlung das Resultat des Sammelprozesses. Sie beinhaltet alle notwendigen Werte für die Museumsvisualisierung.

Der Filter dient in dieser Klasse nur als Speicherobjekt des vom Benutzer gewünschten Zeitintervalls (und im kommenden Sprint die darzustellenden Attribute). Dies bedeutet, dass der ganze

Sammeldurchlauf pro MuseumStatistic nur im Konstruktor stattfindet. Eine Filterung in diesem Sinne existiert nicht.



17.3.3 Offene Punkte

- Da sich ein Besucher in der Regel nicht länger als ein Tag im Museum befindet, wäre es denkbar, dass für die Bestimmung des Besuchstages nur dessen erste Erfassung ausschlaggebend ist. Da sich jedoch ein Durchlaufen aller Erfassungen im Millisekunden-Bereich befindet und eine Mitternachtsausstellung durchaus denkbar wäre, wird nach wie vor über alle Erfassungen iteriert.

17.4 Museumsstatistik - Visualisierung

Museumsstatistiken erlauben einen Überblick der Besuchererfassungen über ein gesamtes Museum. In diesem Sprint wurde die Visualisierung implementiert, welche das zeitliche Besuchervorkommen anhand eines Diagrammes visualisiert.

17.4.1 Analyse

Während der Analyse wurde ersichtlich, dass die Diagramme mit Hilfe einer vorhandenen Library realisiert werden sollten. Grund dafür ist, dass eine eigene Diagramm-Library zu schreiben deutlich mehr Aufwand bedeuten würde, als eine bereits existierende Bibliothek auszuwählen und zu verwenden. Das Ziel dieser Analyse ist folglich das Auffinden einer umfangreichen, aber dennoch einfach anzusteuernden Diagrammbibliothek.

In Diskussionen mit dem Team stellte sich heraus, dass zu diesem Thema schon Analysen existieren. Infolgedessen wurde die Suche nach besseren Alternativen erspart. Es muss nur noch analysiert werden, ob die in früheren Arbeiten verwendete Bibliothek immer noch aktuell ist. Eine kurze Recherche zeigte, dass die damals verwendete Bibliothek „JFreeChart“ bis dato immer noch unterstützt und tatkräftig ausgebaut wurde.

Demzufolge wird primär auf die Literatur vorgängiger Analysen[20] verwiesen.

17.4.2 Anforderung

Folgende Punkte werden vom Diagramm verlangt:

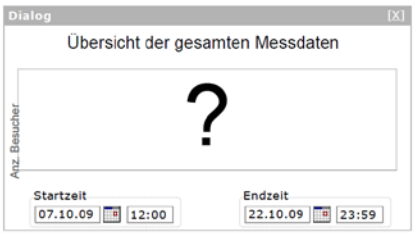
[20] *\07_HS_SA_Ferrari_Kaelin\Gruppe_1\03 Berichte\Bericht.doc S.91 ff*

- *Die Darstellung von Werten über eine beliebige Zeitachse*
- *Mehrere Serien von Messewerten müssen darstellbar sein (z.B. Kinder, Erwachsene & Senioren)*
- *Eine sinnvolle Skalierung bei vielen Messwerten*
- *Die wichtigsten Werte sollen klar ersichtlich sein*

17.4.2.1 Library JFreeChart

Die JFreeChart-Bibliothek bietet vielseitige Darstellungsmöglichkeiten an. In der nachfolgenden Designanalyse werden einige Diagramme vorgestellt.

Die primäre Fragestellung ist, welcher Diagrammtyp verwendet werden soll, um die Besucherzählungen pro Tag darzustellen.



Bildliche Darstellung des Problems.

17.4.2.1.1 DualAxisBar

Dieses Diagramm ist analog zu dem Balkendiagramm. Es bietet jedoch die Erweiterung, zwei verschiedenartige Balken gegenüberzustellen. Denkbar wäre auch ein Trippel- oder eine Mehr-Achsendarstellung.

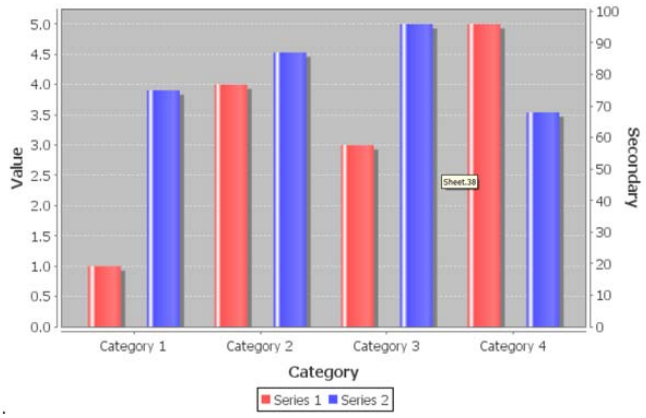


Abbildung 76: DualAxisBar

Vorteil	Nachteil
Balkendiagramme ermöglichen gute Übersicht der einzelnen Kategorien.	Bei der Auswahl von mehr als drei Kategorien werden alle Balken nebeneinander dargestellt, was auf Kosten der Übersicht geht.
Zwei oder mehrdimensionale Darstellungen sind möglich.	Bei vielen Messwerten wirkt das Diagramm überfüllt und einzelne Balken sind kaum zu erkennen.

17.4.2.1.2 TimeLine

Dieses Diagramm stellt einzelne oder mehrere Kurven entlang der zeitlichen Achse dar.

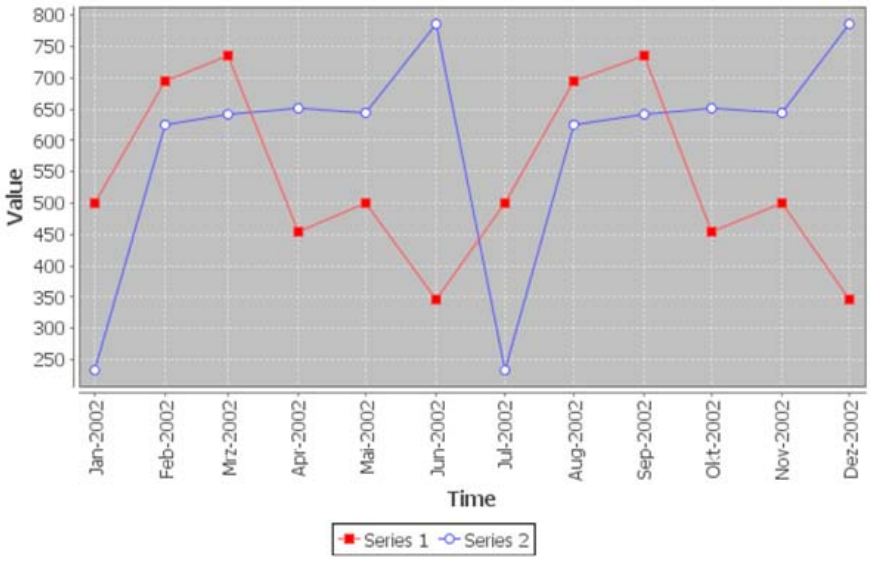


Abbildung 77: TimeLine

Vorteil	Nachteil
Auch bei kleinen Werten skaliert das Diagramm auf die gewünschte Grösse.	Falls einige Messwerte extrem kleine/grosse Werte liefern, wird eine geeignete Y-Achsen Skalierung schwierig.
Durch Punkte im Graphen sind alle Messwerte klar ersichtlich.	
Es können theoretisch beliebig viele Messwerte aufgenommen werden.	
Durch Überlagerung sind mehrere Kurven darstellbar.	

17.4.2.1.3 XYLines mit Spline-Render

Dieser Graph ist das Analogon zur TimeLine, nur mit dem Unterschied, dass hier zwischen den einzelnen Messwerten eine Spline gelegt wird.

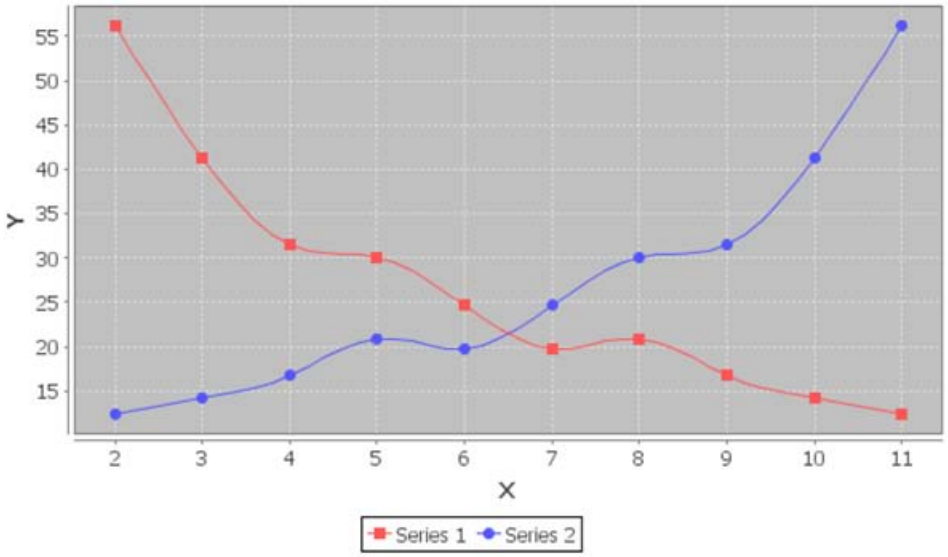


Abbildung 78: XYLines mit Spline-Render

Vorteil	Nachteil
(siehe „TimeLine“)	Kann das Aussehen der Peaks (extrem schnelle Steigung & Fall) zu stark abschwächen.
Visuell schöner gerundete Kurven. Diese wirken glatter und nicht mehr künstlich.	

17.4.2.2 Auswertung der Analyse

Bei den obigen Anforderungen an das Diagramm ist der Anspruch, dass mehrere Darstellungskriterien (wie z.B. „Männlich“, „Jugendlich“ usw.) hinzu geschaltet werden können von primärer Bedeutung. Deshalb liegt das Schwergewicht bei der Auswertung der Analyse auf diesem Kriterium.

Analysierte Chart	Bemerkung	Verwendung
DualAxisBar	Ungenügend, da für jede neu dazugeschaltene Darstellung ein Balken hinzugefügt wird.	Keine
TimeLine	***Erlaubt das Hinzufügen von mehreren Kriterien und stellt diese auch ansehnlich dar. Dieser Charttyp wird verwendet. Erfüllt alle Kriterien die an das Diagramm gestellt wurden.***	***Implementiert***
XYLines mit Spline-Render	Grafische Verfeinerung der TimeLine, wird jedoch nicht primär verfolgt. Implementierungen der Splines sind optional.	(Optional)

17.4.3 Design

17.4.3.1 Hauptpanel

Standardmässig wird das Attribut „alle Besucher“ geladen. Dies bedeutet, dass die Aufsummierung aller Besucher pro Tag dargestellt wird. Die XY-Achsen skalieren automatisch mit dem jeweils kleinsten und grössten Wert.

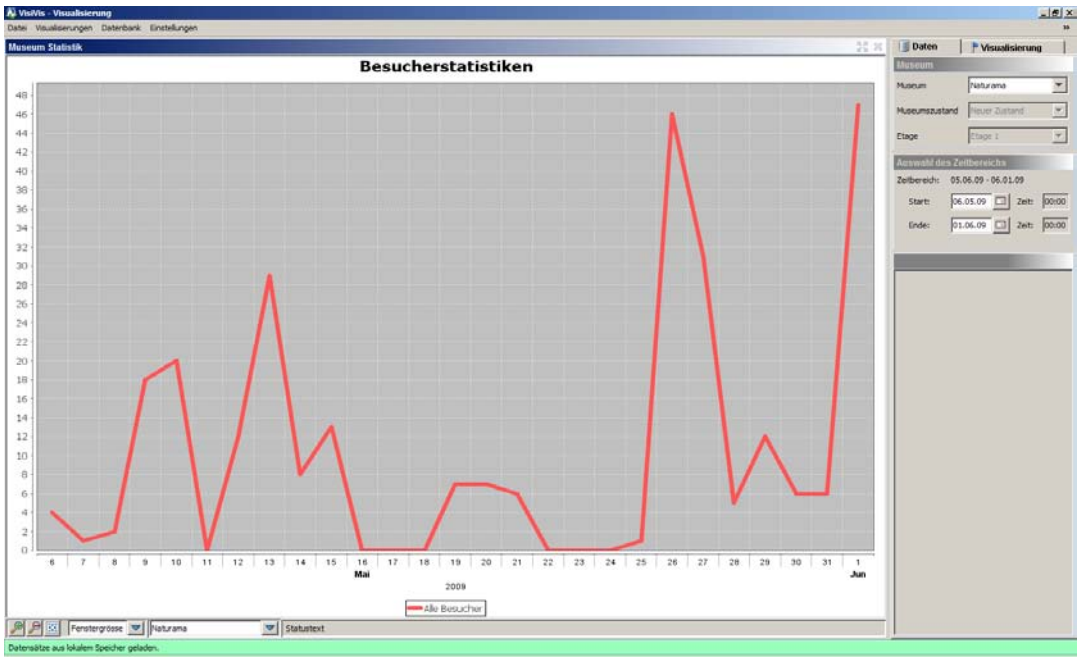
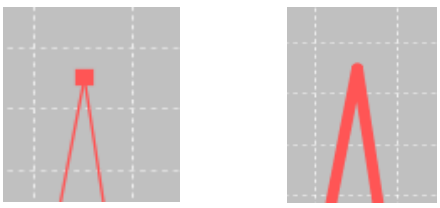


Abbildung 79: Bildauszug der Implementierung von JFreeChart

17.4.3.1.1 Neudesign der Graphen

Während diesem Sprint wurden einige Änderungen an der Graphendarstellung vorgenommen. Neu wird beim Mauszeiger ein Tooltipp angezeigt. Ebenfalls neu ist, dass die einzelnen Messwerte keine Markierungen mehr besitzen. Diese wurden ausgeschalten, da sie wenig zur Übersicht beitrug und somit ein entbehrliches Element darstellte. Zusätzlich wurde die Linienstärke erhöht.



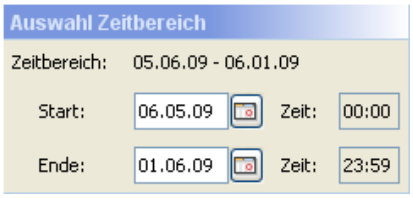
Erste Abbildung markiert den Messwert durch einen Block.
Rechtes Bild besitzt keine Markierung dafür eine dickere Linie.

17.4.3.2 Seitenpanels

Verwendet werden die schon bekannten Seitenpanels (Dockables) mit kleineren Anpassungen.

17.4.3.2.1 Auswahl des Zeitbereichs

Die einzige Abweichung vom Vorgänger ist, dass hier keine Uhrzeit gewählt werden kann. Grund ist, dass die Einheitenskalierung nur über Tage und nicht über Stunden geführt wird.



17.4.3.2.2 Auswahl des Museums

Das Museum ist durch das schon früher verwendete und bekannte MuseumChooser-Dockable auswählbar. Der Museumszustand und die Etage können nicht ausgewählt werden, da diese keinen Einfluss auf die Graphen haben. Die dargestellte Kurve ist unabhängig vom gewählten Zustand oder der Etage.

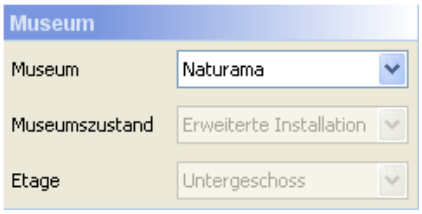


Abbildung 80: Auswahl des Museums auf Museums-Statistikebene

17.4.3.2.2.1 Offene Punkte

Wie in der „Abbildung 80: Auswahl des Museums auf Museums-Statistikebene“ ersichtlich, ist in diesem Menü einerseits der Museumszustand und andererseits die Etage sichtbar. Beide sind hier deaktiviert, da sie keinen Einfluss auf die dargestellte Statistik haben. Dies ist ein offener Task, denn das Dockable unterstützt das Ein-/Ausschalten einzelner Komponenten nur bedingt. (detaillierte Beschreibungen sind im Kapitel „Offene / abgeschlossene Probleme“ einsehbar).

17.4.3.3 Skalierung des Charts

Ein Problem in der Analyse war das Verhalten bei extrem kleinen oder grossen Zeitbereichen bezüglich der X- und Y-Achsen

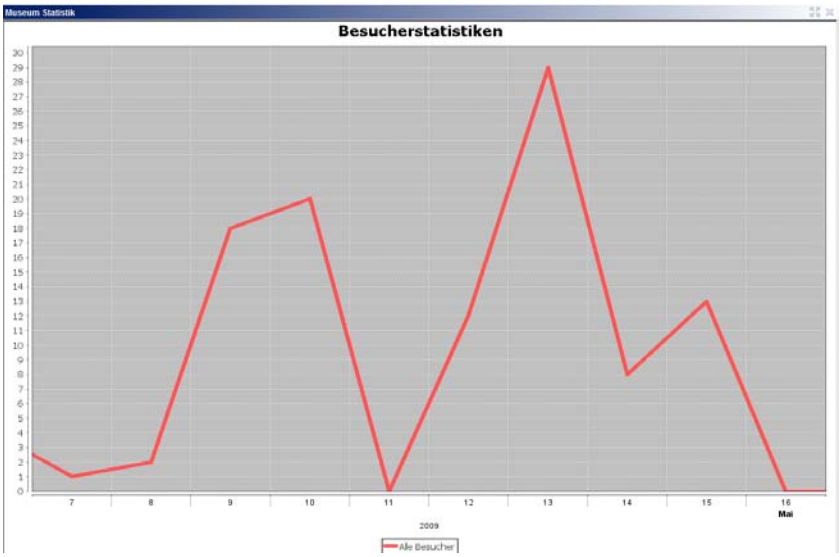


Abbildung 81: Skalierung der X-Achse über den Zeitbereich von 7.Mai bis 16.Mai

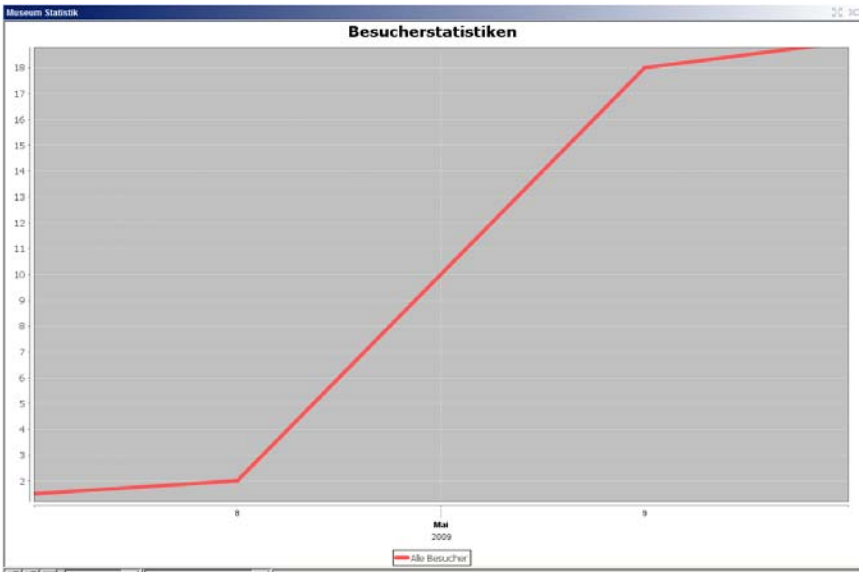


Abbildung 82: Skalierung der X-Achse über 2 Tage

17.4.3.3.1 Skalierung der X-Achse

Die Skalierung erfolgt über das JFreeChart. Das heisst die Library übernimmt bereits Aufgaben für die Skalierung bei Extremwerten. Die Bibliothek bietet auch die Einstellung der Achsenbeschriftung in Tage, Monate und Jahre an.

Die Tage skalieren in Abständen von einzelnen Tagen bis hin zu Wochen, oder wie im unteren Beispiel ersichtlich, jeweils mit 16 Tagen.

12 28 13 1 17 2 18 4 20 5 21 7 23 8 24 9 25 11 27 12 28 14
Jan Feb Mrz Apr Mai Jun Jul Aug Sep Okt Nov Dez
2009

Der Anzeigebereich mit Tagesskalierung von +16 Tagen und der eines Monats.

Falls nun der Zeitbereich über mehrere Jahre hinweg angezeigt werden soll, werden die Anzeigen dementsprechend zusammengefasst und angepasst. Im unteren Bild wird dies am Beispiel der Monate dargestellt.

: Mrz Jun Sep Dez Mrz Jun Sep Dez Mrz Jun Sep
2008 | 2009 | 2010

Zwecks Platzmangels wird nur noch jeder 3te Monat angezeigt.

17.4.3.3.2 Skalierung der Y-Achse

Das Verhalten der Y-Achse ist analog zu der Achse von X, nur dass hier keine Datumswerte verwendet werden müssen sondern ganzzahlige Werte.

17.4.4 Anmerkung zur Besucherzählung

Ein wichtiger Hinweis muss für die Badgeausgabe der Museumsbetreiber angebracht werden. In den Diagrammen wird lediglich die Anzahl der Badgeausgaben aufsummiert und nicht die tatsächliche Anzahl der Besucher. Grund dafür ist, dass nicht immer jedem Besucher ein Badge ausgehändigt wird. Folglich ist bei der Badgeausgabe unbedingt zu beachten, dass in gleichen zeitlichen Abständen und an möglichst randomisierte Personen die Badges vergeben werden. Ist zum Beispiel montags immer ein Kassier im Museum, der jedem Besucher ein Badge gibt und einer am Freitag der dies

vernachlässigt, macht die Statistik einen falschen Anschein. Sie sagt aus, dass montags das Museum viel intensiver besucht wurde als freitags, was nicht der Realität entspricht.

17.4.5 Offene Punkte

- Eventuell sollten auch einzelne Museumszustände auswählbar sein. Dadurch kann das Zeitintervall automatisch auf die Grenzen des gewählten Zustandes gesetzt werden.
- Der Standart-Tooltip der JFreeChart ist nicht intuitiv und muss noch angepasst werden. Dies bedeutet, dass unnötige Werte und Klammern entfernt werden sollten.
- Absolute Besucherzählungen sind nicht sehr aussagekräftig, da diese nur anzeigen wie viele Besucherbadges in diesem Intervall ausgehändigt wurden.

18 Sprint 6

18.1 Änderungsgeschichte

Datum	Änderung
10.12.2009	Erstellung.
14.12.2009	Klassendiagramme hinzugefügt.
14.12.2009	Analyse Kuchen- / Balkendiagramme.
15.12.2009	Zum Hauptbericht verschoben.

18.2 Beschreibung

Das Hauptziel im Sprint 6 bestand darin, bei der Statistik Visualisierung über ein ganzes Museum weitere Kurven pro Attribut darzustellen. Als Zusatzleistung wurden in diesem Sprint zwei weitere Tasks umgesetzt, welche das Anzeigen von Verteilungen (Kuchen- / Balkendiagram) pro Reader und Raum vorsahen.

18.3 Datensammlung

Bis anhin reichte es Besucherzählungen unabhängig von Attributen zu führen, das heisst die Besucherzahl pro Raum oder Tag genügte für die bisherigen Visualisierungen. Für die in diesem Sprint gestellten Aufgaben musste der Datensammelprozess jedoch erweitert werden. Konkret bedeutet dies, dass neuerdings die Besucherzählungen pro Attribut berechnet werden müssen. Auf diese Weise wird es möglich, beispielswiese die Anzahl „männlicher“ Besucher pro Raum oder Tag festzustellen.

18.3.1 Analyse

Ziel der Analyse ist es herauszufinden, welche bisherigen berechneten Werte neu pro Attribut bekannt sein müssen, um den Anforderung in diesem Sprint gerecht zu werden.

Bisherige Werte	Nutzen bei Aufteilung pro Attribut
Besucherzählung	Muss pro Attribut berechnet werden: **ja** Die Etagenvisualisierung benötigt für die Anzeige der Kuchen- bzw. Balkendiagramme die Besucherzahlen pro Attribut. In der Museumvisualisierung müssen ebenfalls Kurven angezeigt werden, welche sich auf Besucherzählung pro Attribut stützen.
Aufenthaltsdauer (Durchschnitts- und Gesamtbesucherzeit)	Muss pro Attribut berechnet werden: **ja** Die Kuchen- und Balkendiagramme stellen die durchschnittliche Aufenthaltsdauer und Gesamtbesucherzeit pro Attribut dar. Es ist somit ebenfalls notwendig diesen Wert pro Attribut zu berechnen.
Minima / Maxima von Besucherzählungen und Aufenthaltsdauern	Muss pro Attribut berechnet werden: **ja** Die Minima / Maxima Werte wurden bisher nur für die Einfärbung der RFID-Reader und Räume verwendet. So wurde beispielweise der RFID-Reader mit der kleinsten Erfassung rot und derjenige mit der grössten Erfassung grün eingefärbt. Für die Einfärbung müssten somit die Minima / Maxima nicht pro Attribut berechnet werden. In Kuchendiagrammen spielen diese Schlüsselwerte pro Attribut ebenfalls keine Rolle, da diese Diagramme die relativen Verhältnisse wiederspiegeln. Bei den Balkendiagrammen ist es jedoch unerlässlich, dass eine konsistente Skala verwendet wird, damit beim Vergleichen von unterschiedlichen Balkendiagrammen die Höhe des Balkens ausschlaggebend bleibt. Damit eine einheitliche Skala erzeugt wird, muss gewissermassen bekannt sein, welcher Reader die meisten „weiblichen“ Besucher erfasst hat.

Die oben durchgeführte Analyse hat ergeben, dass alle bisher berechneten Werte zusätzlich pro Attribut berechnet werden müssen. Für die Laufzeit der Statistikerstellung bedeutet dies, dass jede Erfassung unter Berücksichtigung von Attributen gezählt werden muss. Sie ist nun neu:

*O(Anz. Etagen + Anz. Räume + Anz. Reader + Anz. Besucher + Anz. Erfassungen * Anz. Attribute)*

18.3.2 Design

Im Design werden die Änderungen gegenüber den bisherigen Klassen erläutert und mit Klassendiagrammen illustriert. Als erstes wird ein Überblick der bisher erstellen Klassen gezeigt und erläutert.

18.3.2.1 Klassendiagramm – Überblick

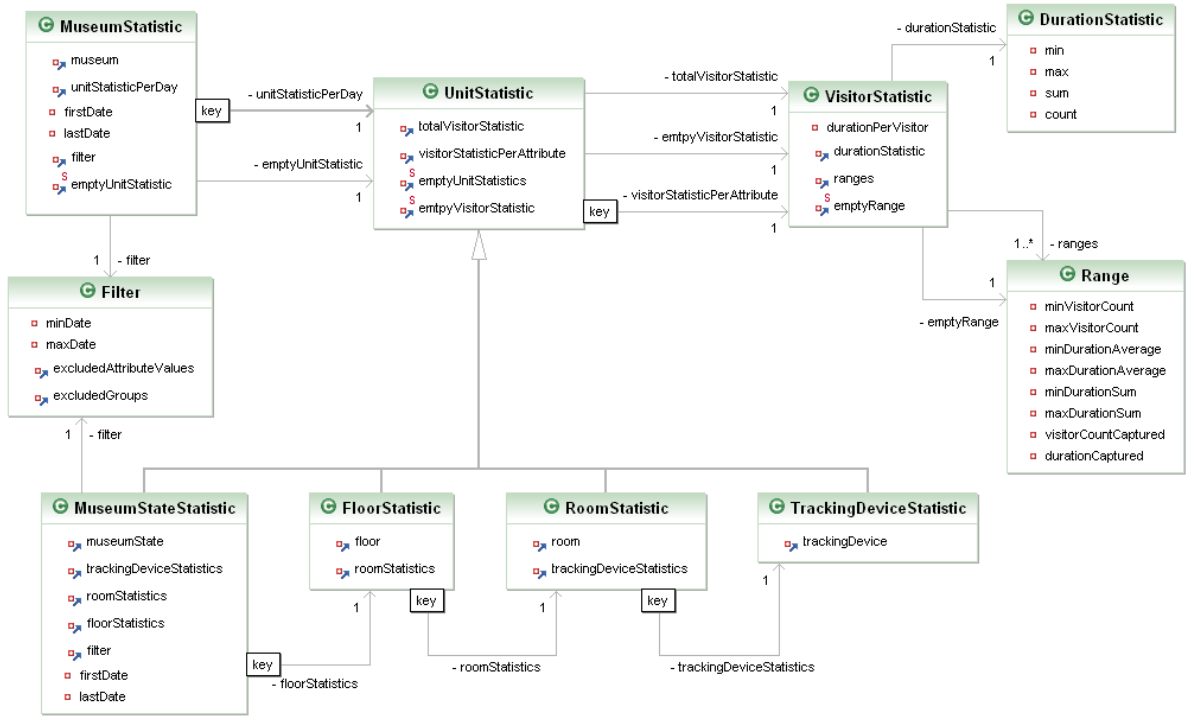


Abbildung 83: Überblick der Statistikklassen

Dargestellt sind alle Klassen, welche für die Berechnungen der statistischen Daten verantwortlich sind. Des Überblicks halber wurden nur die Felder (ohne Methoden und Konstruktoren) dargestellt.

Klasse	Verantwortlichkeit
MuseumStatistic	Berechnet die Anzahl Besucher pro Tag auf Ebene Museum (aufgeteilt in Total und pro Attribut).
Filter	Zuständig für die Entscheidung, ob ein Besucher oder eine Erfassung in der Auswertung berücksichtig wird.
DurationStatistic	Sammelt eine Menge von Zeiten und bestimmt dabei den Schnitt, die Summe, das Minimum und Maximum der gesammelten Werte.
Range	Kapselt die Minima und Maxima von Besucherzählungen, Aufenthaltsdauern und Gesamtbesucherzeiten.
VisitorStatistic	Führt eine Liste von Besuchern und bestimmt aus dieser Liste die Besucheranzahl, Aufenthaltsdauer, Gesamtbesucherzeit und Minima / Maxima.
UnitStatistic	Kapselt alle Besucherdaten getrennt in Total und pro Attribut.
MuseumStateStatistic	Führt und berechnet statistische Daten auf Ebene Museumszustand.
FloorStatistic	Führt und berechnet statistische Daten auf Ebene Etage.
RoomStatistic	Führt und berechnet statistische Daten auf Ebene Raum.
TrackingDeviceStatistic	Führt und berechnet statistische Daten auf Ebene RFID Reader.

18.3.2.2 Klassendiagramm - Etagenstatistik

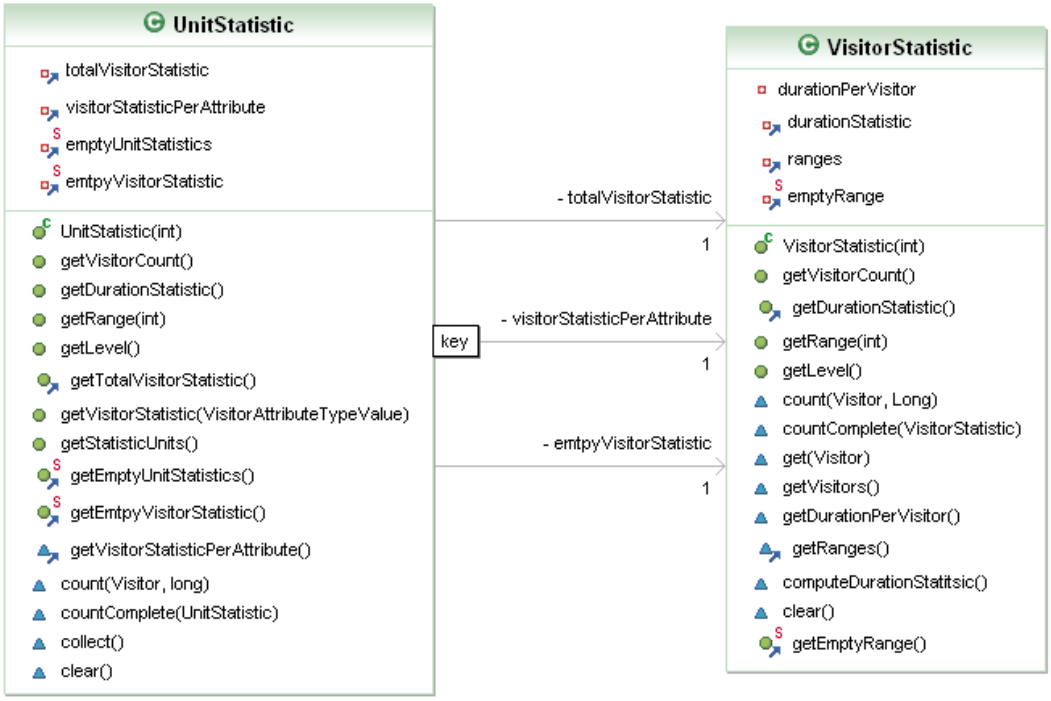


Abbildung 84: Einsicht in die Etagenstatistiken

18.3.2.3 UnitStatistic

Die UnitStatistic hat in diesem Sprint einige ihrer Verantwortungen an die neue Klasse VisitorStatistic abgegeben. Bis anhin waren die drei Werte Besucherzählung (*getVisitorCount*), Aufenthaltsdauer (*getDurationStatistic*) und Minima / Maxima (*getRange*) in der Klasse UnitStatistic direkt assoziiert. Die Klasse VisitorStatistic kapselt fortan diese drei Werte, wodurch es in der UnitStatistic möglich wird, Besucherdaten total oder pro Attribut zu führen.

Die Totalbesucherzahl, welche bis anhin unabhängig von Attributen berechnet wurde, wird ab sofort in der VisitorStatistic *totalVisitorStatistic* geführt. Um das bisherige Interface nicht zu stark zu verändern, bleiben die Getter *getVisitorCount, getDurationStatistic* und *getRange* in der Klasse *UnitStatistic* weiterhin verfügbar, delegieren jedoch nur an die *totalVisitorStatistic* weiter. Die Zählmethoden *count(visitor, duration)* und *countComplete(unitStatistic)* delegieren ebenfalls an eine VisitorStatistic weiter. Die eigentliche Zählung findet in der VisitorStatistic statt.

Die Map *visitorStatisticPerAttribute* führt Besucherdaten (VisitorStatistic) pro Attribut. Falls im Sammeln der Besucherdaten gewisse Attribute niemals aufgetreten sind, kann es vorkommen, dass bei *getVisitorStatistic(attribute)* ein Null-Pointer zurückgeliefert würde. Um eine Nullabfrage auf der Callerseite zu verhindern, wird in diesem Fall die statische VisitorStatistic *emptyVisitorStatistic* zurückgeliefert. Die *emptyVisitorStatistic* enthält alles 0 Werte.

18.3.2.4 VisitorStatistic

Die VisitorStatistic kapselt neu alle Besucherdaten. Diese wären die Anzahl Besucher und Aufenthaltsdauer pro Besucher. Beide Werte werden aus der Map<Visitor, long> *durationPerVisitor* gelesen. Dabei entspricht die Anzahl Besucher *durationPerVisitor.size()* und die Aufenthaltsdauer pro Besucher *durationPerVisitor.get(visitor)*.

18.3.2.4.1 Methode count(visitor, duration)

Durch die Methode *count(visitor, duration)* wird die Aufenthaltsdauer des Besuchers *visitor* berechnet. Da ein Besucher ein Tracking Device mehrmals besuchen kann, wird die Methode *count(visitor, duration)* ebenfalls mehrmals für denselben Besucher *visitor* aufgerufen. Dabei wird bei jedem Aufruf die bereits registrierte Aufenthaltsdauer des Besuchers *visitor* um *duration* erhöht. Nachfolgend wird der Ablauf in einem Sequenzdiagramm visualisiert:

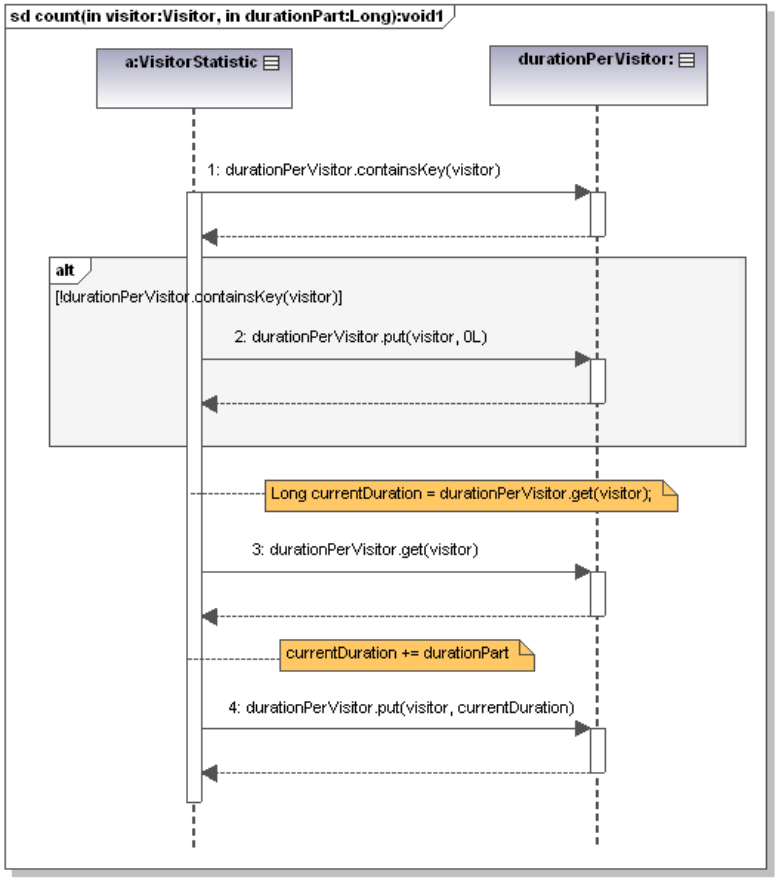


Abbildung 85: Sequenzdiagramm von count()

Beim Aufruf von *count(visitor, duration)* wird als erstes geprüft, ob für den Besucher *visitor* bereits eine Aufenthaltsdauer erfasst wurde. Falls nicht, wird der Besucher *visitor* in der Map *durationPerVisitor* mit einer Aufenthaltsdauer von 0 Millisekunden eingetragen. Nach diesem Punkt ist garantiert, dass *durationPerVisitor.get(visitor)* keinen Null-Pointer zurückliefert. Die erhaltene Besuchsdauer wird um *duration* erhöht und wieder in die Map zurückgeschrieben.

18.3.2.4.2 Methode countComplete(visitorStatistic)

Die Methode *countComplete(visitorStatistic)* verwertet im Gegensatz zu *count(visitor, duration)* eine komplette VisitorStatistic. Die Methode wird beispielweise von einer RoomStatistic verwendet, um die Besucherdaten der TrackingDeviceStatistcs in sich zu vereinen. Bei diesem Prozess werden ebenfalls die Minima / Maxima aktualisiert. Dies bedeutet, dass die RoomStatistic das TrackingDevice mit der geringsten Besucheranzahl oder grössten durchschnittlichen Aufenthaltsdauer kennt. Diese Schlüsselwerte werden von der Visualisierung für eine optimale Einfärbung verwendet.

18.3.2.4.3 Methode computeDurationStatistic()

Wurden mit *count(visitor, duration)* oder *countComplete(visitorStatistic)* alle Besucher verwertet, kann mit *computeDurationStatitsic()* eine Aufenthaltsdauer-Statistik erstellt werden. Konkret bedeutet dies, dass von den Aufenthaltsdauern in der Map *durationPerVisitor* der Durchschnitt gebildet wird. Für das Bestimmen dieses Wertes ist die Klasse DurationStatistic verantwortlich. Im nachfolgenden Sequenzdiagramm wird der Ablauf illustriert:

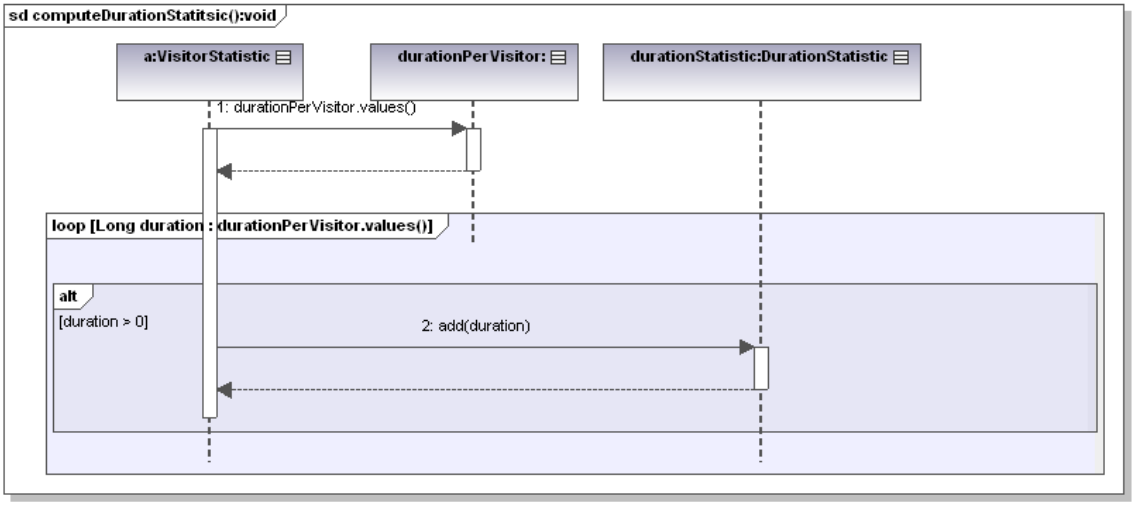


Abbildung 86: Sequenzdiagramm von computeDurationStatistic()

18.3.2.5 Klassendiagramm – Museumstatistik

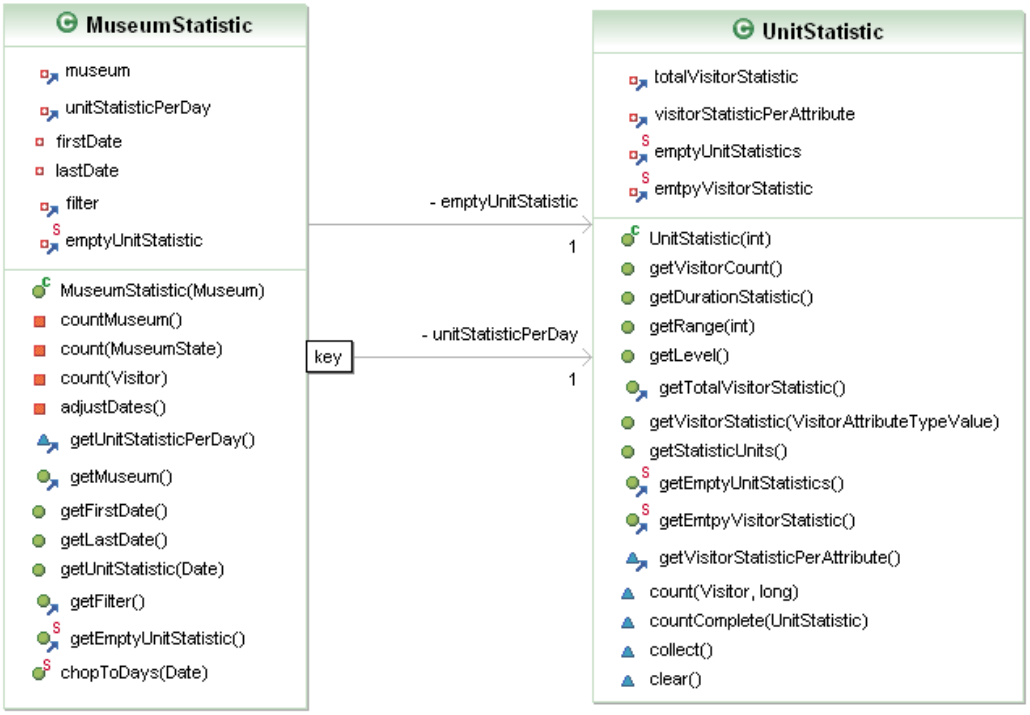


Abbildung 87: Einblick in die Klasse der Museumsstatistiken

18.3.2.6 MuseumStatistic

Die Klasse MuseumStatistic berechnet die Daten für die Visualisierung über ein gesamtes Museum. Im letzten Sprint führte diese Klasse eine *Map<Date, Set<Visitor>> totalVisitorsPerDay*, welche die Menge an Besucher pro Tag festhielt. Neu verwendet sie die *Map<Date, UnitStatistic> unitStatisticPerDay*, welche die Besucherzählung pro Attribut und total (unabhängig von Attributen) führt.

Im Datensammelprozess wird die Methode *count(visitor, 0)* der UnitStatistic Klasse verwendet. Sie führt, wie im vorherigen Kapitel beschrieben eine Zählung pro Attribut und dem Total aus. Da die Aufenthaltsdauer in der Museumsvisualisierung keine Rolle spielt, wird der Besucher mit einer Aufenthaltsdauer von 0 Millisekunden gezählt.

Falls bei *getUnitStatistic(day)* die UnitStatistic eines Tages *day* abgefragt wird, welche nicht in der Map *unitStatisticPerDay* enthalten ist, wird standardmässig die leere UnitStatistic *emptyUnitStatistic* zurückgeliefert. Dies verhindert eine Nullabfrage auf der Callerseite.

18.3.3 Qualitätskontrolle

18.3.3.1 Unit-Tests

Aus dem Coverage Report vom Hudson ist ersichtlich, dass mit den Testcases nahezu 100% der Klassen im Statistic Package abedeckt werden konnten. Für weitere Details kann im Hudson[21] nachgeschaut werden.

Package Coverage Summary

Name	Files	Classes	Methods	Lines	Conditionals
ch.hsr.ifs.visivis.visual.pd.statistic	100% 10/10	100% 10/10	100% 122/122	100% 384/384	97% 173/178

Coverage Breakdown by File

Name	Classes	Methods	Lines	Conditionals
UnitStatistic.java	100% 1/1	100% 16/16	100% 48/48	100% 22/22
RoomStatistic.java	100% 1/1	100% 6/6	100% 14/14	100% 2/2
Filter.java	100% 1/1	100% 21/21	100% 54/54	100% 30/30
TrackingDeviceStatistic.java	100% 1/1	100% 3/3	100% 5/5	100% 0/0
VisitorStatistic.java	100% 1/1	100% 15/15	100% 44/44	94% 17/18
FloorStatistic.java	100% 1/1	100% 6/6	100% 14/14	100% 2/2
MuseumStateStatistic.java	100% 1/1	100% 16/16	100% 75/75	97% 29/30
MuseumStatistic.java	100% 1/1	100% 14/14	100% 44/44	91% 20/22
Range.java	100% 1/1	100% 17/17	100% 66/66	98% 43/44
DurationStatistic.java	100% 1/1	100% 8/8	100% 20/20	100% 8/8

Abbildung 88: Package Coverage Summary von Hudson

18.4 Visualisierung

In diesem Sprint wurden mehrere Visualisierungen umgesetzt. Dies ist einerseits die Darstellung von weiteren Kurven pro Attribut in der Museums Visualisierung (zum Beispiel der Verlauf der „männlicher“ Besucher in einem bestimmten Zeitraum) und anderseits die Darstellung von Verteilungen (Kuchen- oder Balkendiagramme) in der Etagen Visualisierung. Die letztere war laut Sprintmeeting nicht verlangt, wurde aber als Zusatzleistung dennoch implementiert.

[21] http://sinv-56027.edu.hsr.ch:443/job/VisiVis%20-%20JUnit/, letzter Zugriff 15.012.09

18.4.1 Analyse

Um die Verteilungen demographischer Eigenschaften und weiterer Besucher Merkmalen (wie z.B. „mit Audioguide“) in der Etagen Visualisierung darzustellen, wurden die Kuchen- und Balkendiagramme in Betracht gezogen. In vorgängigen Studienarbeiten[22] sind diese Diagrammtypen bereits verwendet worden. Aus diesem Grund bauen wir auf ihren Analysen und Erkenntnissen auf.

Folgende kurze Analyse soll die Stärken und Schwächen der Diagrammtypen zeigen:

Diagrammtyp	Beschreibung	
Kuchen	Stärke	Das Kuchendiagramm eignet sich besonders gut um relative Verhältnisse darzustellen. Beim Vergleich von mehreren Kuchendiagrammen ist sofort ersichtlich, welche Teile eine Minder- oder Mehrheit ausmachen.
	Schwäche	Wird kein absoluter Wert im Kuchendiagramm angezeigt, ist es unmöglich auf die absolute Gesamtmenge zu schliessen.
Balken	Stärke	Das Balkendiagramm eignet sich besonders gut, um absolute Werte darzustellen. Beim Vergleich von mehreren Balkendiagrammen ist von der Höhe der Balken her abzulesen, wo sich die grösste Gesamtmenge befindet.
	Schwäche	In einem Balkendiagramm ist es schwieriger die relativen Verhältnisse abzulesen.

Da sich die Stärken beider Diagramme gegenseitig ergänzen, sind beide Diagrammtypen geeignet für eine Implementierung.

18.4.1.1 Problem mit dem Zoomen

Ein anderes Problem entsteht, falls während dem Anwenden der Software mehrere Fenster miteinander verglichen werden. Standardmässig versucht die Applikation den Grundriss des Museums auf dem Fensterinhalt zu maximieren, damit der Benutzer eine gute Übersicht gewinnt. Ein Nachteil dabei ist, dass mit dem Fensterinhalt auch die die Diagramme entsprechend mit skalieren. Falls der Benutzer die Ansicht vergrössert, werden die Diagramme zwar besser lesbarer, verkleinert er jedoch die Ansicht, werden die Schriften und Diagramme schlechter erkennbar.

[22] *\07_HS_SA_Ferrari_Kaelin\Gruppe_1\03 Berichte\Bericht.doc, S.173 ff*

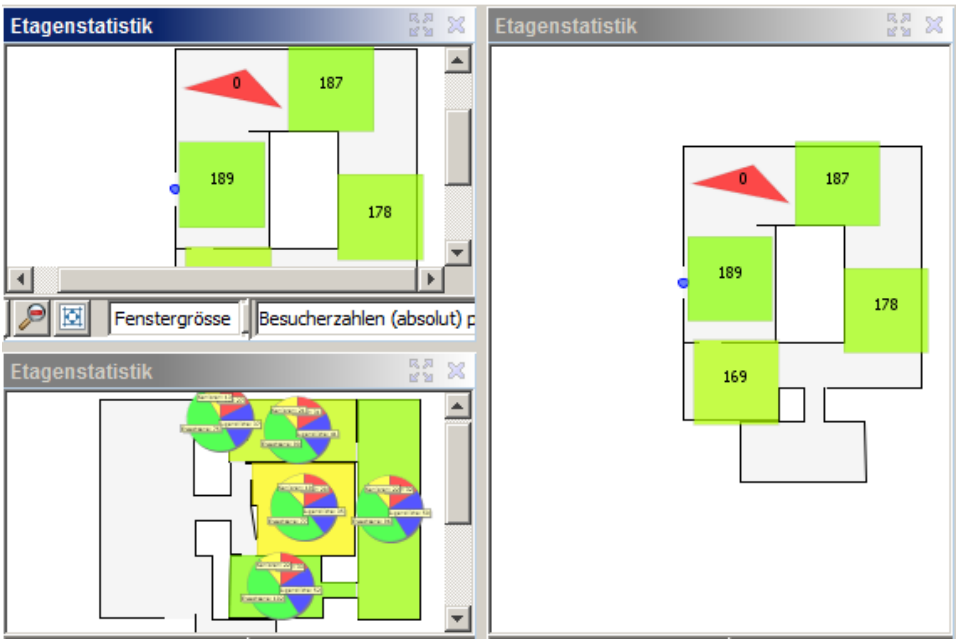


Abbildung 89: Zooming lässt einzelne Text und Werte unlesbar erscheinen

Folgende Analyse versucht diesem Zoomproblem entgegenzuwirken.

Lösungsansatz	Beschreibung	Auswertung
Kein Zoom	Einfachste Lösung wäre, dass in den Diagrammdarstellungen das Zoomen nicht zugelassen wird.	Ungenügend. Der Anwender wird dadurch zu stark einschränkt und Vergleiche mit mehreren Fenster werden beinahe unmöglich.
Grundrisszoom aber fixe Diagrammgrössen	Erlaubt das Zoomen des Grundrisses, jedoch die Skalierung der Diagramme bleibt fix.	Dies führt beim Hinaus-/Hineinzoomen zu inkonsistenter Darstellungen, da nur der Grundriss gezoomt wird.
Mit Zoom und Tooltips	Zoomen ist uneingeschränkt erlaubt. Durch Tooltipps können schlecht lesbare Werte trotzdem abgelesen werden.	Beste Lösung, da hier die Zoommöglichkeiten nicht eingeschränkt werden und der Anwender immer noch Informationen über einzelne Diagrammwerte erhalten kann.

18.4.2 Design

18.4.2.1 Hauptpanel - Museumvisualisierung

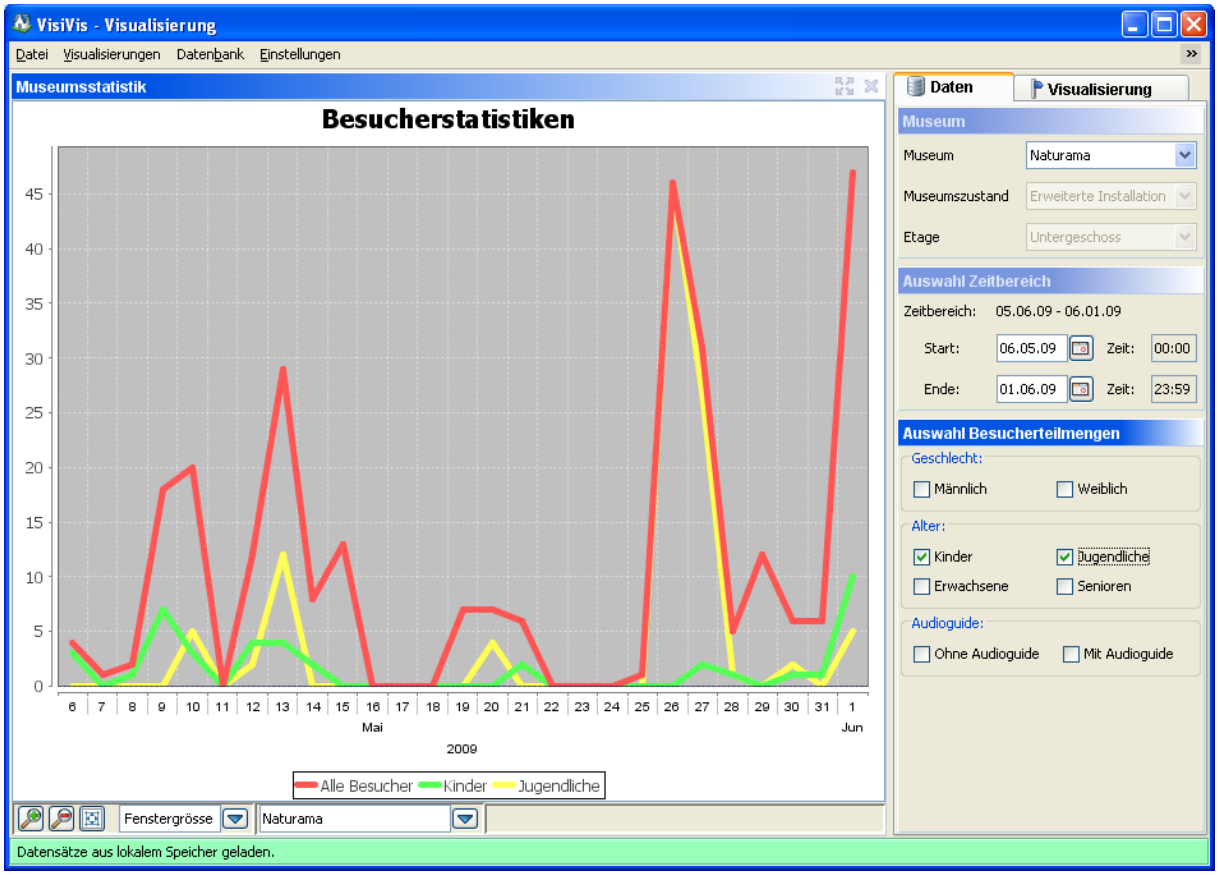


Abbildung 90: Besucherstatistiken

Auf dem Bildschirmfoto ist die Gesamtansicht der Museumvisualisierung zu sehen. Neu findet sich auf der rechten Seite eine Auswahl weiterer Kurven pro Attribut. Es können beliebig viele weitere Kurven dargestellt werden. Die zugewiesene Farbe pro Attributkurve bleibt über das Program hinweg konstant.

18.4.2.2 Hauptpanel - Etagenvisualisierung

Das Hauptpanel besteht aus verschiedenen Dockables, welche die Auswahl der verschiedenen Visualisierungen zulässt. Diese wurden im aktuellen Sprint um weitere Optionen erweitert.

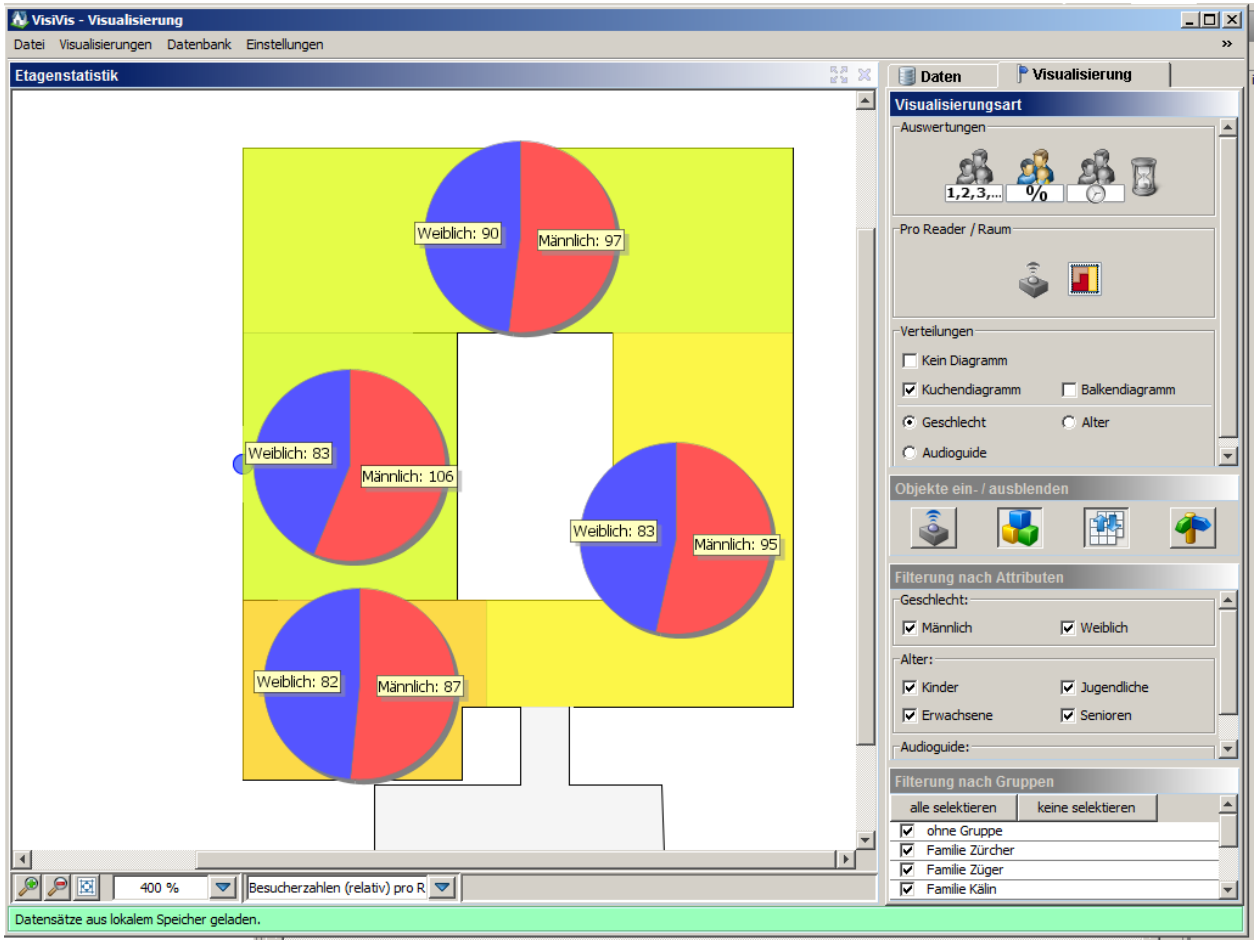


Abbildung 91: Bildschirmfoto zeigt die bis dato aktuellste Version vom 10.Dezember 2009

Auf der linken Bildschirmhälfte sind die Visualisierung sichtbar, rechts die Dockables.

18.4.2.3 Verteilung Auswahl

Neuerdings gibt es im „Visualisierungsart“-Dockable die Möglichkeit eine Verteilung auszuwählen.

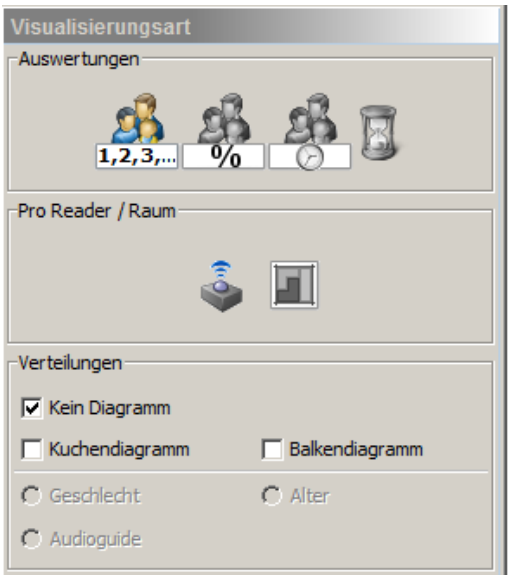


Abbildung 92: Dockable mit der neuen Verteilungoption

Wie im obigen Bild ersichtlich, wird standardmässig „Kein Diagramm“ selektiert. Dadurch sieht der Anwender den absoluten Wert ohne Diagramme. Grund dafür ist, dass in einem ersten Blick einzelne Werte einfacher zu interpretieren sind als mehrwertige Diagramme. Wünscht der Anwender diese Zahlen zusätzlich zu visualisieren, kann er mit Auswahl der Checkboxen („Kuchendiagramm“ oder „Balkendiagramm“) die gewünschte Darstellung auswählen. Die Auswahl des Diagramtyps ist dabei unabhängig von der gewählten Visualisation (Besucherzählung absolut / relativ, Aufenthaltsdauer und Gesamtbesuchszeit). Das bedeutet es werden alle Visualisierungstypen von den Diagrammen unterstützt.

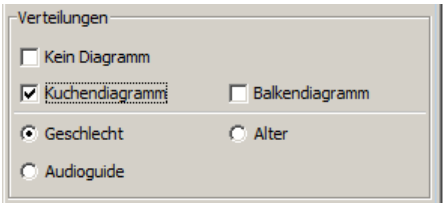
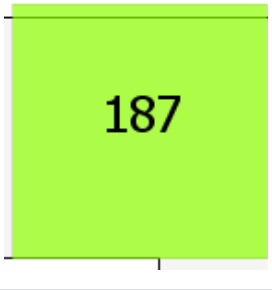
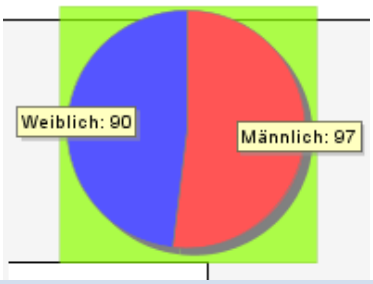
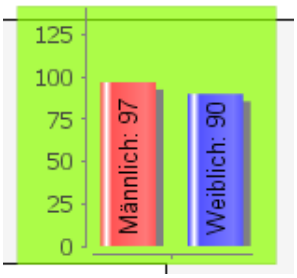


Abbildung 93: Verteilungen als Kuchendiagramme

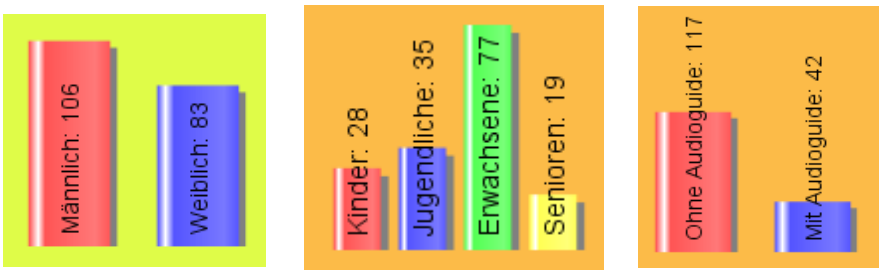
Diese Auswahl würde dazu führen, dass in den Readern (oder Räumen) Kuchendiagramme dargestellt werden.

Folgende Bilder zeigen die drei verschiedenartigen Diagrammtypen.

Benennung	„Kein Diagramm“	„Kuchendiagramm“	„Balkendiagramm“
Visualisierung			
Beschreibung der Werte / Grafik.	Zahl stellt die Aufsummierung von Weiblich und Männlich dar.	Die vorherige Zahl wird aufgeteilt in verschiedene Mengen: Weiblich und Männlich.	Aufteilung in zwei Mengen: Weiblich und Männlich.

Ist entweder das Kuchen- oder Balkendiagramm ausgewählt, werden die Attributgruppen (Radiobuttons) selektierbar. Standardmässig wird das erste Attribut (hier „Geschlecht“) selektiert. Die Attributgruppen sind erst zur Laufzeit bekannt, da sie dynamisch von der Datenbank oder dem lokalen Cache geladen werden. Um Platz zu sparen werden jeweils zwei Attributgruppen auf einer Zeile aufgeführt. Die Auswahl von Attributen hat direkte Auswirkung auf die dargestellten Werte des Kuchen- bzw. Balkendiagrammes. Nachfolgend wird das Verhalten beschrieben:

Je nach Auswahl werden die Entsprechenden Attributwerte dargestellt:



Mögliche Diagramm Darstellungen je nach Attributwahl

18.4.2.4 Kuchendiagramm

Von der Analyse her ist bekannt, dass Kuchendiagramme die relativen Verhältnisse besonders gut darstellen.

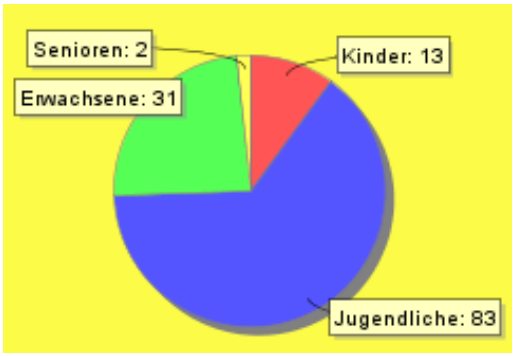


Abbildung 94: Darstellung eines Kuchendiagramms mit dem Attribut "Alter"

18.4.2.4.1 Behandlung der Nullwerte

Falls ein einzelnes Attribut mit einem Vorkommen von 0 auftritt, wird es im Diagramm trotzdem aufgeführt. Dadurch bleibt immer ersichtlich, welche Attribute aktuell dargestellt werden.

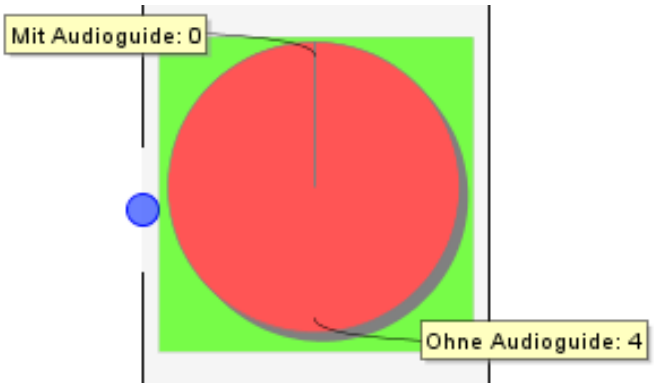


Abbildung 95: Kuchendiagramm mit Darstellung eines Null-Vorkommen

Falls alle Werte den Messwert 0 aufweisen, wird kein Diagramm erstellt. Es wäre nicht möglich ein Kuchendiagramm darzustellen, welches keinerlei Werte aufweist. Somit muss mindestens ein Wert im Diagramm positiv sein.

18.4.2.5 Balkendiagramm

Balkendiagramme dienen primär zur Veranschaulichung von absoluten Werten, beispielsweise der Besucheranzahl einzelner Altersgruppen.

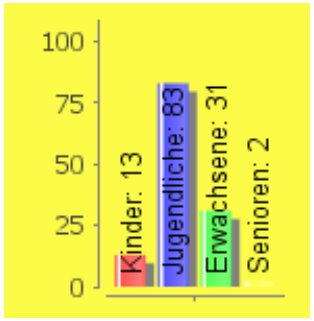


Abbildung 96: Beispielhafte Darstellung eines Balkendiagrammes

18.4.2.5.1 Behandlung der Nullwerte

Im Balkendiagramm werden einzelne Nullwerte ebenfalls toleriert und wie folgt dargestellt:

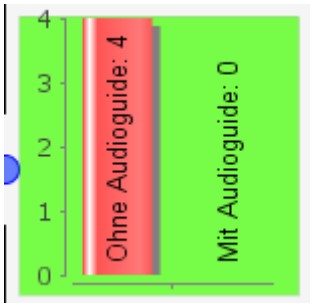


Abbildung 97: Balkendiagramm mit Darstellung eines Null-Wertes

Falls alle Werte 0 aufweisen, wird das Diagramm nicht angezeigt, da kein vernünftiges Diagramm angezeigt werden kann.

18.4.2.6 Verhalten bei Filterung

Der Anwender kann jederzeit den Filter anpassen, auch während die Darstellung von Balken- oder Kuchendiagramm aktiv ist. Die Diagramme werden dabei unmittelbar aktualisiert.

18.4.2.6.1 Wegfiltern von anzuzeigenden Attributen

Falls der Benutzer das Attribut „Kinder“ weggefiltert, wird dementsprechend bei dieser Kategorie der Wert 0 dargestellt. Beispielsweise führt eine Anzeige der Verteilung nach „Alter“ zu folgendem Verhalten:

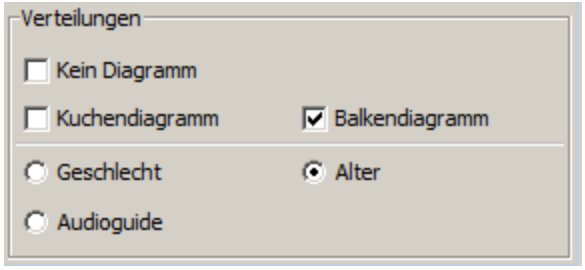


Abbildung 98: Ausgangslage: Balkendiagramm mit Verteilung des Alters

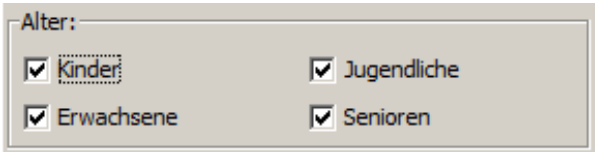


Abbildung 99: Das De-/SElektieren von Attributen haben direkte Auswirkungen auf alle Diagramme.

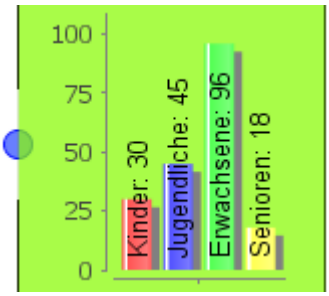


Abbildung 100: Ohne Filterung

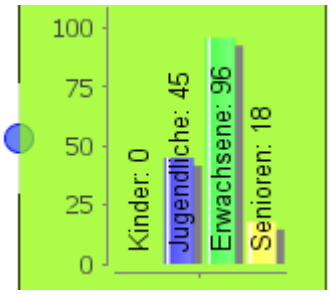


Abbildung 101: Mit Wegfiltern von "Kinder"

18.4.2.7 Das Zoomproblem

In der Analyse wurde festgehalten, dass der Tooltipp einen Lösungsansatz bietet, da er vom Zoom her unabhängig ist. Dementsprechend sind die Werte im Kuchendiagramm durch den Tooltipp lesbar, auch wenn der Benutzer stark heraus gezoomt.

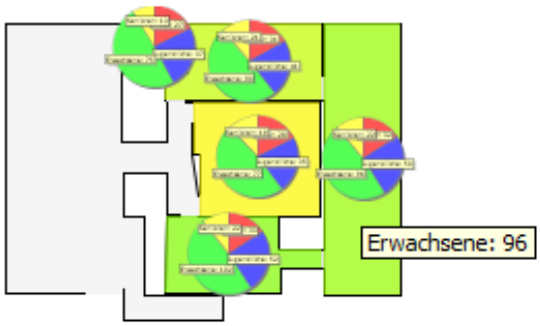


Abbildung 102: Trotz starken Zooming, können die Werte - Dank Tooltipp - gelesen werden.

18.4.2.8 Filterung nach Attributen

Das „Filterung nach Attributen“ Dockable wurde leicht umgestaltet. Die Attribute werden fortan in zwei Kolonnen dargestellt, was bedeutend platzsparender ist als die vorherige Lösung mit nur einer Kolonne.

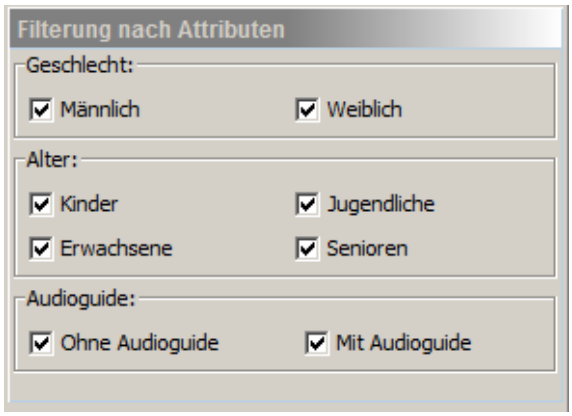


Abbildung 103: Filterung nach Attributen

18.4.2.9 Gruppenfilter

Das Dockable „Filterung nach Gruppen“ erhielt ein zusätzliches Feature, damit der Benutzer auf einfache Art und Weise alle Gruppen de- / selektieren kann.

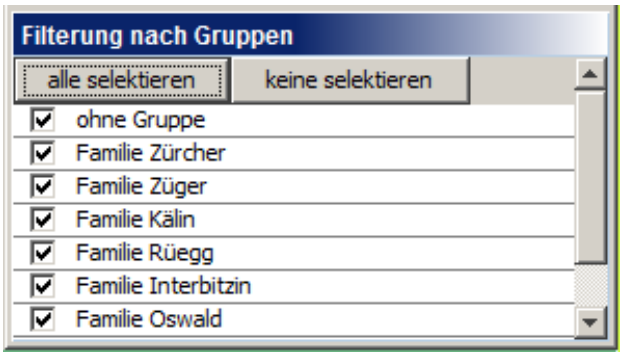


Abbildung 104: Neue Buttons: alle de-/selektieren...

19 Persönliche Berichte

19.1 Simon Inderbitzin

Im persönlichen Erfahrungsbericht wird einen Einblick in meine Gedanken und Erfahrungen während des Sprint 1 und 2 aufgeführt. Der zweite Teil zeigt eine Denkschrift über Scrum, das Team und das Projekt im Allgemeinen.

19.1.1 Beispiel aus der Praxis

19.1.1.1 Sprint 1

Ziel im Sprint1 war für mich, ein geeigneter Algorithmus für die Beschriftungsposition im Polygon zu finden. Die Implementation des Inkreis-Algorithmus zeigte sich schwieriger als erwartet. Alternativ wurde der Schwerpunkt-Algorithmus und das Aufsummieren der Koordinaten implementiert - welche für konvexe Polygone guten Lösungen darstellen. Konkave Polygone (wie z.B. Abbildung 19: Konvexes Polygon) stellen nach wie vor ein Problem dar. Es existieren einige gute Algorithmen, die aber meinen eingeteilten Zeitrahmen sprengen würden.

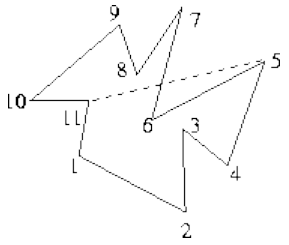


Abbildung 105: Konvexes Polygon

19.1.1.2 Sprint 2

Diese Woche erprobten wir das Pairprogramming und diskutierten über mögliche Implementierung der Task ID 1 (Auswahl des Zeitbereichs und Etage). Wir wählten genau diesen Task, denn dieser zeigte sind aufwändiger als erwartet. Das aktuelle Problemdomain erlaubt keinen direkten Zugriff auf die von uns benötigten Klassen. Die Lösung dieses Themas schien nicht ganz trivial und stellt somit eine geeignete Wahl für das Austesten des Pairprogramming dar. Zusammen konnten wir eine mögliche Realisierung erarbeiten und einen Prototyp ausprogrammieren. Es war sehr oft der Fall, dass einer von uns seine Gedanken mitgeteilt haben, der andere ihn dann auf Pro und Kontra hingewiesen hat. So gelang es uns, falsche Ansätze schnell zu verwerfen und neue, bessere gemeinsam zu entwickeln. Der einzig negative Punkt am Pairprogramming ist, dass dadurch immer 2-facher Zeitaufwand benötigt wird – so werden aus 3 Stunden Pairprogramming insgesamt 6 Mannstunden. Wir werden Pairprogramming auch in weiteren Sprints anwenden, jedoch wirklich nur da, wo es auch sinnvoll ist, denn das Gut “Zeit“ ist knapp bemessen.

19.1.2 Am Anfang war das „Scrum“

Unser Team besteht aus einer Zweiergruppe, Patrick Oswald und Simon Inderbitzin. Wir befanden uns in der glücklichen Lage, dass wir unser SA-Wunschthema zugesprochen erhielten. Dementsprechend zeigten wir von Anfang an grosse Motivation. Als Einführung in das Projekt wurden wir zum ersten Mal mit der Scrum-Technik vertraut. Ich persönlich hatte vorher noch nie wirklich von dieser Arbeitsstrukturierung und Ablaufstechnik vernommen. Zu Beginn zeigte ich mich skeptisch, denn diese agile Art der Programmierung fühlte sich nach viel Selbständigkeit an, da keine

wirklich fixen Meilensteine gelegt werden. Ich sah die Gefahr darin, dass die Entwicklung ziellos sein könnte und keine Fixpunkte aufwiesen. Während eines Meetings führte Thomas Kälin, seinerseits Scrum-Master, die Organisation und Vorgehensweisen von Scrum ein. Im Scrum gibt es nicht die wie sonst üblichen Meilensteine oder detaillierte Vorausplanung über mehrere Monate. Scrum kennt agile und iterative Arbeitspakete. Der Arbeitsaufwand dieser Pakete wird gemeinsam im Team für einen Sprint (oft 2-3 Wochen) geschätzt und schliesslich von den jeweiligen Softwareentwicklern abgesegnet. Der Productowner ist hierbei nicht anwesend, was ich durchaus positiv erlebte. Da meiner Meinung nach sonst ein Interessenskonflikt entstehen könnte, da jeder PO die Software am liebsten heute schon fertig hätte, der Entwickler jedoch möchte qualitativ gute Software erstellen und diese erst als erledigt markieren, wenn es zu 99% sicher ist, dass sie auch funktioniert.

Unser erster Sprint diente dazu, einen gewissen Einblick in diese Thematik zu erhalten. Folglich zeigte sich die Sitzung des Testsprints als äusserst zeitaufwändig. Bei späteren Sprintmeetings war dies glücklicherweise nicht mehr der Fall.

Im Laufe weiterer Sprints geriet das gesamte Projektteam immer mehr in Rückstand, da offensichtlich zu zeitkritisch geschätzt wurde. Dies lässt sich auch darin belegen, dass der Arbeitsaufwand unseres Teams deutlich über dem eigentlich verlangten Studienarbeitsaufwand befand. Als Gegenmassnahme ergriffen wir die Option, pro Sprint ein Entwicklungstag (8Stunden) als Pufferzone einzuteilen und allgemein einen optimistischeren und weniger zeitkritische Schätzungen abzugeben.

Mit der neuen Schätzungspolitik in unserem Scrum-Team, arbeiteten wir zeitlich nicht weniger, konnten aber durch die grosszügigen Schätzungen auch wirklich den ganzen Sprint abschliessen. Dies löst natürlich auch ein besseres Wohlbefinden innerhalb des Teams aus, weil die Entwickler etwas fertig haben und nicht ständig das Gefühl haben mit der Arbeit in Verzug zu sein. Persönlich schätzte ich diese Lockerung sehr, und fühlte mich Motiviert auch einige „nächste Task“ in Angriff zu nehmen, auch wenn diese nicht auf der Todoliste standen.

Während der ganzen Studienarbeit erlebten wir somit viel Neues und wurden durch unzählige positive Aspekte überrascht. Um hier eine davon zu nennen, kann ich mit guten Gewissen sagen, dass unser Scrum-Master (Thomas Kälin) eine wirklich sehr gute Projektleitung geführt hatte. Er war jederzeit für Fragen und Antworten verfügbar, half an kritischen Stellen und führte während den ganzen Sprintmeetings eine hervorragende Leitung.

Zum ersten Mal in einer produktiven Arbeit lernte ich den aktiven Umgang mit Hudson, Wiki und Ticketsystemen. Diese Arbeitsumgebung wendete ich bis dato nicht wirklich an, da in 2-3er Teams oft ein SVN ausreichend war. In einem Projekt von mehr 5 Personen, zeigten sich diese neuen Hilfsmittel als äusserst geeignet. Einzig der Hudson hatte die Einstellung, dass bei jedem fehlgeschlagenen Build eine Email an alle Entwickler rausging. Dies führte an Spitzentagen zu mehr als 15 Emails an einem Tag. Nichtsdestotrotz empfand ich alle Tools aus durchaus berechtigt und sogar als Notwendigkeit. In unserem Team intern benutzten wir im Sprint 1 bis 2 ein eigenes Wiki. Die Absicht war, dass wir all unsere Dokumentationen dort verfasst werden. Wir erhofften uns dadurch auch eine agile Möglichkeit der Dokumentationserstellung. Dieses Vorgehen zeigte sich aber schnell aus unbrauchbar und Patrick suchte nach guten Alternativen. Fündig wurden wir mit Dropbox, dieses Tool lässt mehrere Windowsordner direkt über das Internet teilen. Beispielsweise, wenn ein

Teammitglied ein Dokument oder eine Datei neu erstellt, werden diese innert Sekunden auf allen Partner erscheinen. Mit Dropbox machten wir so gute Erfahrung, dass wir diese auch für kommende Projekte einsetzen werden. Dasselbe gilt für das Zeiterfassungstool Paymo, das eine on- und offline Zeiterfassung erlaubt.

19.2 Patrick Oswald

19.2.1 Scrum

Beim Lesen der Original Aufgabenstellung ist mir das Wort Scrum sofort ins Auge gefallen. Bis anhin war mir aus dem Software Architecture 2 Projekt nur RUP bekannt. Der Eindruck aus dem Wikipediaeintrag über Scrum war mir allerdings sehr schnell sympathisch. Ziel von Scrum ist es, möglichst schnell und in regelmässigen Abständen testbare Prototypen zu erstellen. Der grösste negative Punkt bei RUP war meiner Ansicht nach die gewaltige Anzahl an Dokumenten und Richtlinien. Diesbezüglich scheint Scrum weniger Anforderungen zu stellen, was sich auch bewahrheitet hat.

Ich persönlich bin sehr überzeugt von Scrum und würde jederzeit wieder nach diesem Prozessmodell vorgehen.

19.2.2 XP-Praktiken

19.2.2.1 Planing Poker

Das Hinlegen einer Schätzung der Teammitglieder zur selben Zeit finde ich eine gute Idee. Würde jeder der Reihe nach seine Meinung beisteuern, würde man sich dieser unbewusst angleichen. Durch das gleichzeitige Hinlegen ist man jedoch gezwungen sich eine unabhängige Meinung zu bilden und diese auch zu vertreten.

Mit der Granularität der Karten bin ich jedoch nicht ganz einverstanden. Es gibt die Karten 1, 2, 3, 5, 8, 13, 21 und höhere. Die Idee dieser Einteilung ist, dass man die Zahl 4 schätzt keine Gedanken darüber macht, ob man die Karte 3 oder 5 richtig wäre. In der Tat entscheidend man sich nur ungern für das eine oder andere, war zumindest bei mir so war.

Für mich wäre es somit angenehmer, ich könnte die Zahl hinlegen, die ich auch geschätzt habe. Da wir ca. 10 Karten in der Hand halten, wär für mich folgende Granularität passender: 1, 2, 3, 4, 5, 6, 8, 10, „?“, „Pause“. Höhere Werte waren bisher nur selten gefragt, falls doch könnte mach sich überlegen eine 16 mit den Karten 10 und 6 abzudecken.

19.2.2.2 Pair-Programming

Pair-Programming im engsten Sinne hat sich bei uns nicht direkt durchgesetzt. Was wir stattdessen führten, könnte man am ehesten als Pair-Analyse und Pair-Review bezeichnen. Wir besprechen anfangs die Architektur und Schnittstellen und teilen die eigentliche Implementation in möglichst unabhängige Teile auf. Nachdem jeder seinen Teil umgesetzt hat, wurde der erstellte Code dem anderen genau erklärt. Gab es dabei Unstimmigkeiten, wurde diese erneut analysiert und umgesetzt. Dieses Vorgehen wurde solange fortgeführt, bis die Implementation von beiden Entwicklern vollständig akzeptiert wurde. Dieser Ablauf hat sich für uns sehr gut bewährt.

19.2.3 Projektart

Die Zusammenarbeit im Team hat überraschend gut funktioniert. Es hat allerdings ein bisschen gedauert bis unser Team und das der Wegverfolgung aufeinander eingegangen ist. Wenn man mit Leuten zusammenarbeitet die man nicht sehr gut kennt ist man wohl automatisch etwas skeptisch eingestellt und muss erst herausfinden, ob eine Zusammenarbeit funktionieren kann. Interessant war auch die Mischung aus Bachelor und Master Studenten. Ich war überrascht von der gut eingerichteten Umgebung, auf welcher wir aufbauen konnten. Thomas Kälin und Michael Rüegg waren immer für Fragen zur Stelle und halfen bei Problemen.

19.2.4 Erfahrungsbericht

In dieser Arbeit habe ich eine Menge neuer Tools kennengelernt, welche ich sicher wieder in weiteren Arbeiten verwenden werde. Zum einen ist dies Dropbox, welches das Synchronisieren von Dokumenten extrem erleichtert. Weiter war Paymo als Zeiterfassung sehr nützlich, da man bei einer eigenen Buchführung schnell einmal den Überblick verliert. Durch das Eclipseplugin Checkstyle wurde die Codequalität stark erhöht. Das Erfassen einer Dokumentation in einem Wiki hat sich leider nicht als sehr produktiv erwiesen. Man hat zwar eine gute Änderungshistorie und keine Konflikte beim Schreiben, allerdings ist das Erfassen selber sehr mühsam.

Die Teamarbeit zu zweit war sehr produktiv aber nicht immer ganz ohne Ecken und Kanten. Es gab einige Male unterschiedliche Ansichten, die man klären musste. Wir konnten allerdings immer eine Lösung finden, die schlussendlich beiden gedient hat. Das Finden einer gemeinsamen Lösung kostete allerdings in der Regel sehr viel Zeit, was zum Glück nicht allzu oft vorkam.

Im Softwarearchitecture Projekt nahmen wir selten die Chance den Betreuer nach Feedback zu fragen. Wir versuchten in dieser Arbeit, diese Chance vermehrt zu nützen. Das Feedback von Prof. Dr. Lothar Müller war stets hilfreich und produktiv. Wir werden auch in zukünftigen Arbeiten den Kontakt nicht scheuen und freuen uns auf die nächste Herausforderung.

20 Schlussbericht und Reflexion

Durch frühere Projektarbeiten erlernten wir den Umgang im Team zu interagieren. Trotzdem hatten wir noch nie das Vergnügen in einem Team solcher Grössenordnung mit wirken zu dürfen. Da alle Entwickler am ein und demselben Projekt arbeiteten, wurde eine starke Interkommunikation benötigt. Diese praxisnahe Ausgangslage zeigte sich als ein gutes Erlebnis und bereitete uns hervorragend auf die Arbeitswelt vor.

Ausserdem war es interessant zu sehen wie der Umgang in grossen Applikationen und deren Erweiterungen funktioniert und gehandhabt wird. Zusätzlich erhielten wir einen Einblick in die Vorgehensweisen anderer Teammitglieder. Gewisse Probleme analysierten und lösten diese unterschiedlich. Beim Studium dieser Fertigkeiten konnten wir oft neue Erkenntnisse mitnehmen und lernten so kontinuierlich dazu.

Während den vergangenen 12 Wochen realisierten wir die erwarteten Anforderungen der Studienarbeitsbeschreibung und fügten sogar einige zusätzliche Aspekte hinzu.

Wie mit unserem Betreuer abgesprochen, dürfen wir die nun erschaffene Basis weiterverwenden, indem wir diese interessante Arbeit auch während unseres Bachelorstudiums erweitern dürfen. Auf die kommende Herausforderung freuen wir uns ausserordentlich.

Zum Schluss möchten wir uns noch bei unserem Betreuer, Prof. Dr. Lothar Müller, für sein Engagement und seine stets konstruktive Zusammenarbeit bedanken. Ein besonderer Dank geht auch an Thomas Kälin, der sich als Scrum-Master - im Kampf gegen Bugs - heroisch bewiesen hat.

21 Anhang

21.1 Glossar

Die für das Projekt wichtigsten Begriffe und Abkürzungen sind in untenstehender Tabelle aufgeführt.

Begriff	Erklärung
API	Application Programming Interface
UI	User Interface
LM	Lothar Müller, zuständig für Betreuung, Product Owner
PO	Product Owner (siehe LM). Trägt die Verantwortung für das Produkt und steht in deren Besitz. Dieser legt das gemeinsame Ziel fest, das das Team zusammen mit ihm erreichen muss.
PD	Problem Domain
HSR	Hochschule für Technik Rapperswil
GUI	Graphical User Interface
Paymo	Zeiterfassungstool
RGB	Rot, Gelb, Blau
Thumbs	Ist die Beschreibung der Buttons in einem Slider
ECTS	European Credit Transfer System
MySQL	Open-Source Datenbankverwaltugnssystem welches von zahlreichen Hostern angeboten wird.
RFID	Radio Frequency Identification – Technologische Grundlage für das Tracking der Besucher.
RFID Reader	In der Umgangssprache das gleiche wie ein RFID Lesegerät. In der PD repräsentiert ein RFIDReader das Lesegerät. Im UI ist dies das RFIDLesegerät, welches ein RFIDReader Objekt umschliesst.
SVN	Subversion Versionierungssystem.
Visitor	In der PD wird so der Besucher genannt.
Build	Eine bestimmte Version oder Variante einer Software

21.2 Literaturverzeichnis

- Inkreis: http://de.wikipedia.org/wiki/Inkreis, letzter Zugriff am 15.12.09
- Umkreis: http://de.wikipedia.org/wiki/Umkreis, letzter Zugriff am 15.12.09
- Dreieck: http://en.wikipedia.org/wiki/Triangle, letzter Zugriff am 15.12.09
- Beispiel Applet von Dreieck: http://www.walter-fendt.de/m14d/inkreis.htm, letzter Zugriff am 15.12.09

- Elementargeometrie: http://www.uni-flensburg.de/mathe/zero/veranst/elemgeom/schwerpunkte/schwerpunkte.html, letzter Zugriff am 15.12.09
- Schwerpunkt: http://de.wikipedia.org/wiki/Schwerpunkt, letzter Zugriff am 15.12.09
- Straight skeleton: http://en.wikipedia.org/wiki/Straight_skeleton, letzter Zugriff am 15.12.09
- Anwendung von Straight skeleton: http://stackoverflow.com/questions/1109536/an-algorithm-for-inflating-deflating-offsetting-buffering-polygons, letzter Zugriff am 15.12.09
- OMONDO: http://www.uml2.org/, letzter Zugriff am 15.12.09
- Applets zum Thema Schwerpunkt, letzter Zugriff am 15.12.09
- Beispiele von 3-/N-Eck-Schwerpunkt, letzter Zugriff am 15.12.09
- JCommon: http://www.jfree.org/jcommon/api/, letzter Zugriff am 15.12.09
- JFreeChart: http://www.jfree.org/jfreechart/api/javadoc/, letzter Zugriff am 15.12.09
- http://www.analyzemath.com/, letzter Zugriff am 15.12.09
- http://de.wikipedia.org/wiki/Feuerbachkreis, letzter Zugriff am 15.12.09

21.3 Abbildungsverzeichnis

21.4 Dokumente des Projekts

21.4.1 Protokolle der Sitzungen

Alle Sitzungen wurden protokolliert und sind auf der CD ersichtlich unter dem Ordner Sitzungsprotokolle[23].

[23] CD-ROM: \4. Projekt Management\Sitzungsprotokolle\...