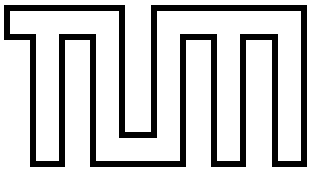


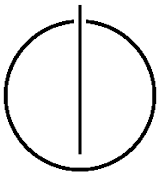
FAKULTÄT FÜR INFORMATIK

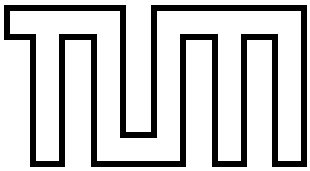
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Informatik

Entwicklung und Implementierung einer mobilen Befragungsanwendung mit 3D Produktmodellen

Adrian Schnell





FAKULTÄT FÜR INFORMATIK

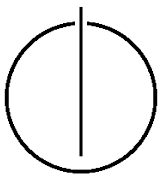
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Informatik

Development and implementation of a mobile survey application with 3D product models

Entwicklung und Implementierung einer mobilen Befragungsanwendung mit 3D Produktmodellen

Author:	Adrian Schnell
Supervisor:	Prof. Bernd Brügge, Ph.D.
Advisor:	Stephan Krusche
Date:	November 15, 2011



Abstract

Before an automobile goes into production, in addition to several years of development, market researches are needed. These are essential to assess the success of a new car model and serve the existing international tastes of the customers.

Currently, these market researches are paper-based conducted. This leads to increased organizational effort for the planning, procedure and evaluation of the market investigation. It is possible, that errors occure, resulting from the manuscriptal collection of the surveys. For example, the handwriting's unreadable. Also, there may not be enough space for freetext comments. It is desirable to reduce this effort and the sources of problems to a minimum.

This bachelor thesis wants to solve the described problems.. A prototype of a mobile application for tablet computers will be developed and implemented, which is characterized by simple and intuitive usage, faster creation of new questionnaires and mostly automated evaluation of the survey results.

Use Cases for the tablet approach will be identified, that can reduce these drawbacks. Following, use cases will be defined to reduce these drawbacks. After modelling and implementing an application it will be evaluated, if all use cases are met.

Kurzfassung

Bevor ein Automobil in Serienproduktion geht, ist neben mehreren Jahren Entwicklungszeit Marktforschung nötig. Diese ist von essentieller Bedeutung um die Erfolgschancen auf dem Markt zu prüfen und um den derzeit geltenden Autogeschmack der Käufer, möglichst international, zu bedienen.

Aktuell werden diese Marktforschungen papierbasiert durchgeführt. Dies führt zu einem erhöhten organisatorischen Aufwand für die Planung, Durchführung und Auswertung der Marktuntersuchung. Weitere Probleme ergeben sich durch Fehlerquellen, die sich durch eine handschriftliche Erfassung der Umfragen ergeben können. Beispielsweise könnte die Handschrift unlesbar sein. Auch könnte nicht ausreichend Freiraum für Freitexte vorhanden sein. Es wird daher angestrebt, diesen Aufwand und die Problemquellen auf ein Minimum zu reduzieren.

Diese Bachelorarbeit macht es sich zum Ziel, diese anspruchsvolle Aufgabe anzugehen. Es wird ein Prototyp einer mobilen Anwendung für Tablet-Computer entwickelt und implementiert, die sich durch einfache, intuitive Bedienung, schneller Einrichtung neuer Fragebögen sowie digitaler Bereitstellung der Umfrageergebnissen für eine größtenteils automatisierte Auswertung auszeichnet.

Um dies zu erreichen, wird zunächst die papierbasierte Umfrage analysiert, um die Nachteile zu identifizieren. Um diese zu eliminieren, werden im nächsten Schritt die Anforderungen definiert. Im Anschluss wird ein System modelliert, das diese Anforderungen erfüllt. Auf die darauf folgende Implementierung der Anwendung folgt eine Evaluierung, ob das System alle diese Forderungen erfüllt.

Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 14. November 2011 Adrian Schnell

Danksagung

Mein größter Dank gilt Stephan Krusche, meinem Betreuer für diese Arbeit, der mich von Anfang bis Ende tatkräftig mit Tipps und Ratschlägen begleitet hat und jederzeit ein offenes Ohr für meine Probleme gefunden hatte. Auch danke ich ihm für die zahlreichen Inspirationen, die er mir gegeben hat. Ebenfalls möchte ich Prof. Bernd Brügge herzlich danken, dass er mir die Möglichkeit gegeben hat, meine Bachelorarbeit an seinem Lehrstuhl schreiben zu dürfen.

Ebenfalls danken möchte ich meinem Kommilitonen und guten Freund Alexander Waldmann, der mir bei fachlichen Problemen immer zur Seite stand und mich dabei unterstützt hat, sofern es möglich war und sich auch nicht gescheut hat, das ein oder andere Feierabendbier mit mir zu genießen.

Auch meinen Eltern möchte ich herzlichst danken, die mir während meines gesamten Studiums hinter mir gestanden haben und den manchmal nötigen Rückhalt gegeben haben, den man gerade auch in Prüfungszeiten benötigt.

Zu guter Letzt danke ich selbstverständlich all den Korrekturlesern dieser Arbeit herzlichst.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Teil I.

Einführung

In dem folgenden Kapitel wird eine kurze Einführung und Problemstellung für diese Bachelorarbeit gegeben.

Dabei wird zunächst eine Beschreibung des Problems gegeben, woraus die Aufgabe und Motivation dieser Arbeit hervorgeht. Weiterhin wird ein Ausblick auf die weitere Gliederung dieser Arbeit aufgeführt sowie die Konventionen angegeben, die im Weiteren verwendet werden.

1. Problemstellung und Motivation

Die Entwicklung von Automobilen ist ein komplexer, aufwendiger und teurer Prozess. Abbildung 1.1 zeigt im groben den Produktlebenszyklus eines Autos. Nach einer etwa dreijährigen Entwicklungsphase folgen etwa sieben Jahre Produktionszeitraum, in der das Auto hergestellt wird. Parallel dazu läuft der 12 Jahre lange Betriebs- und Servicezeitraum. Dieser dauert länger als der Produktionszeitraum, da auch nach Beendigung der Herstellung dieses Automodells Betrieb, Service und Wartung sichergestellt sein müssen.

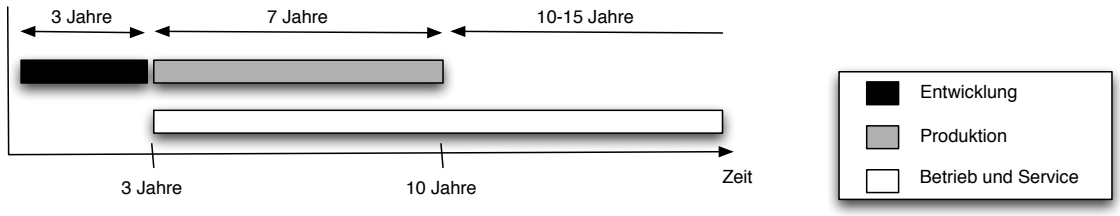


Abbildung 1.1.: Produktlebenszyklus eines Fahrzeuges, nach [SZ10, 20f]

Diese Arbeit beschäftigt sich ausschließlich mit der ersten Phase, der Entwicklung. Hier werden neben technischen Aspekten auch die Optik des zukünftigen Automodells betrachtet.

In dieser Konzeptplanung wird das allgemeine Design des zukünftigen Autos festgelegt, allgemeine Voruntersuchungen angestellt sowie das Budget festgesetzt. Ebenfalls in dieser Phase gibt es erste Marktuntersuchungen, um bereits in frühen Stadien feststellen zu können, ob dieses Auto zukünftigen Kunden gefällt. Diese Marktuntersuchungen werden auch **„car review"** oder allgemein **CAPI** [1] genannt und sind relevant, um frühzeitig die Erfolgschancen auf dem Markt zu prüfen. Der Ablauf eines Car Reviews ist in Abbildung 1.2 dargestellt.

Anschließend auf ein Car Review wird das Design verfeinert sowie ein erster Prototyp des neuen Autos gebaut. Darauf folgend wird ein neues Car Review durchgeführt. In diesem ersetzt der Prototyp das bisherige 3D-Modell.

Erst dann geht es in Produktion und den Verkauf des neues Automodells.

[1] übersetzt: Rechner-unterstützte, persönliche Befragung

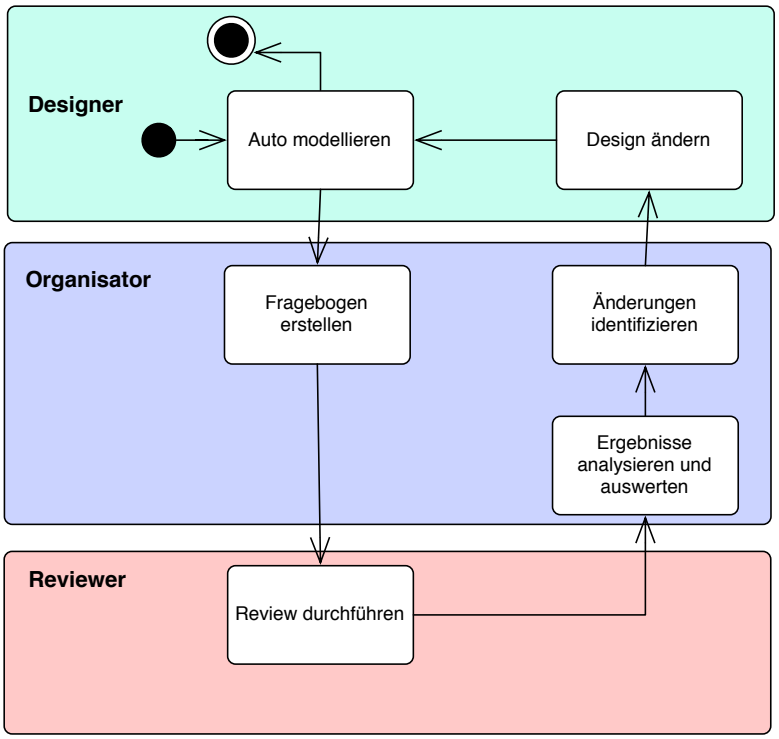


Abbildung 1.2.: Top-Level Aktivitätsdiagramm des Ablaufes eines Reviews

Aktuell werden Car Reviews papierbasierend durchgeführt. Der Aufwand der Planung und der Auswertung ist dabei sehr groß und benötigt viel Zeit. Bei der Durchführung können sich Probleme ergeben wie beispielsweise falsches Verständnis der Fragestellungen, eine falsche Dokumentation der Umfrageergebnisse oder eine schlecht lesbare Handschrift. Bei Freitextfragen können nicht beliebig lange Texte geschrieben werden, da nicht unbegrenzt viel Freiraum geboten werden kann. Probleme wie Datensicherheit werden hier nicht behandelt. Verloren gegangene Umfragebögen können von jedem gelesen werden. Sollte sich im Ablauf der Befragung kurzfristig etwas ändern, beispielsweise dass Autos hinzukommen oder wegfallen, müssen alle Umfragebögen geändert und an die Reviewer ausgehändigt werden. Den größten Zeitaufwand jedoch benötigt aktuell die Auswertung der Papierbögen.

Diese Arbeit beschäftigt sich mit der Entwicklung einer Software, die den Vorgang eines Car Reviews vereinfachen und beschleunigen soll. Ziel ist es, die oben genannten Probleme aufzugreifen und zu beseitigen. So sollen beispielsweise ergänzende Informationen zu dem Auto sowie seinen einzelnen Bauteilen abrufbar sein, sollten die Fragen der Gäste das Wissen der Reviewer überragen.

Der Ablauf des aktuellen und dem angestrebten Reviewverfahren ist in Abbildung 1.3 durch ein UML Aktivitätsdiagramm dargestellt. Auffällig ist, dass bei dem aktuellen papierbasierten Ablauf, im weiteren „As-Is“ bezeichnet, jeder Fragebogen manuell ausgewertet wird, während bei dem angestrebten Ablauf, im weiteren „To-Be“ genannt, die Auswertung größtenteils maschinell durchgeführt wird. Die Option, multimediale Kom-

mentare hinzuzufügen, ist nur bei dem To-Be Ablauf möglich. Voraussetzung dafür ist jedoch, dass der Tablet Computer über eine eingebaute Kamera und Mikrofon verfügt.

Im Vordergrund der Anwendung steht ein digitales 3D Modell eines Autos, auf welchem Notizzettel angepinnt werden können. Diese können Freitext, Multiple Choice Fragen, Sprachaufnahmen, Videos oder auch Fotos enthalten. Die so gesammelten Daten werden an einen Server übertragen, um sie dort automatisiert auszuwerten. Die Benutzer dieser Software sind die Reviewer, also diejenigen Personen, die die Gäste eines Car Reviews zu deren Meinung befragen.

Wie wir bereits gesehen haben, gibt es im Ablauf der Autoentwicklung verschiedene Stadien, an denen ein Car Review eingesetzt wird. Hier beschäftigen wir uns mit Reviews, bei denen noch kein Prototyp besteht und die Reviews rein auf das Modell bezogen sind. In Abbildung 1.4 entspricht das dem Zeitpunkt des 1. Reviews.

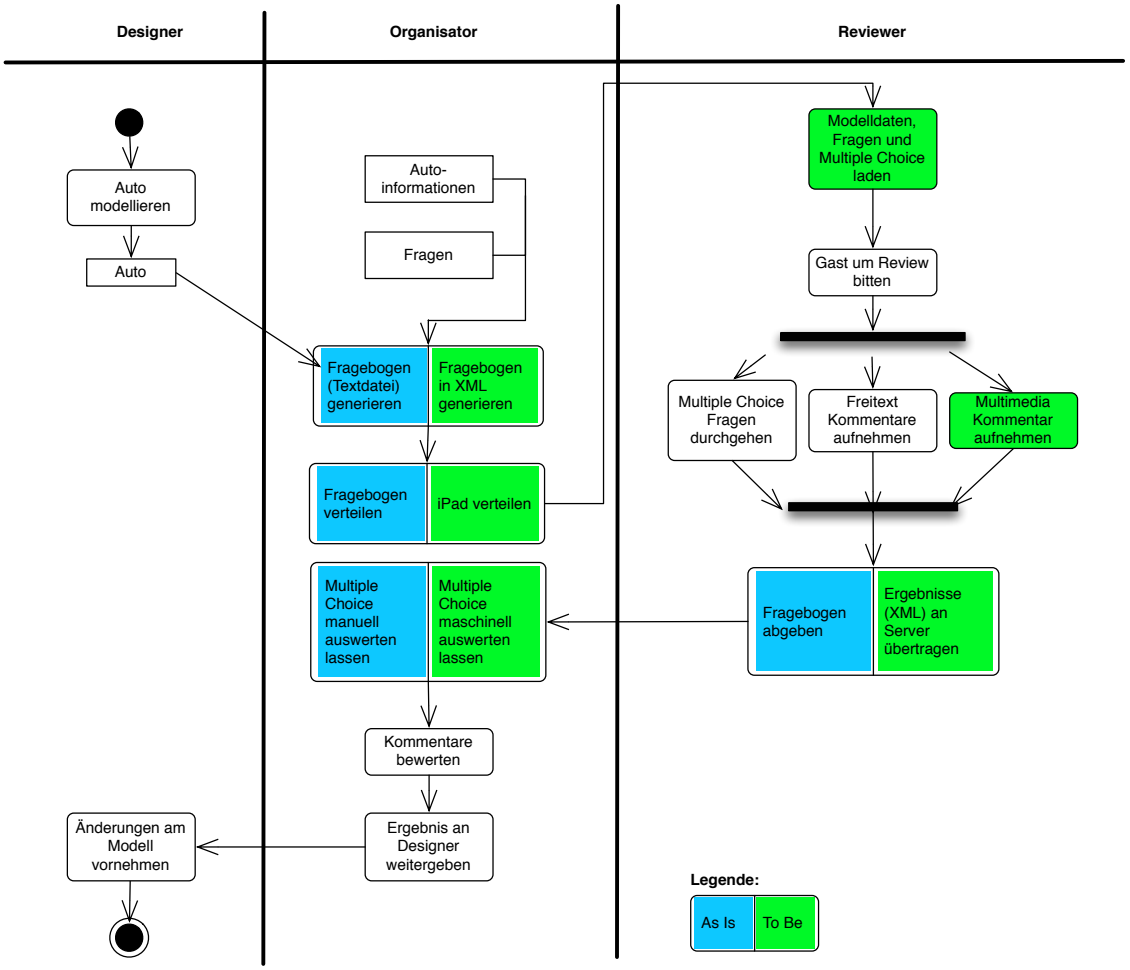


Abbildung 1.3.: Kombiniertes Aktivitätsdiagramm für den „As-Is“- und „To-Be“-Ablauf eines Reviews

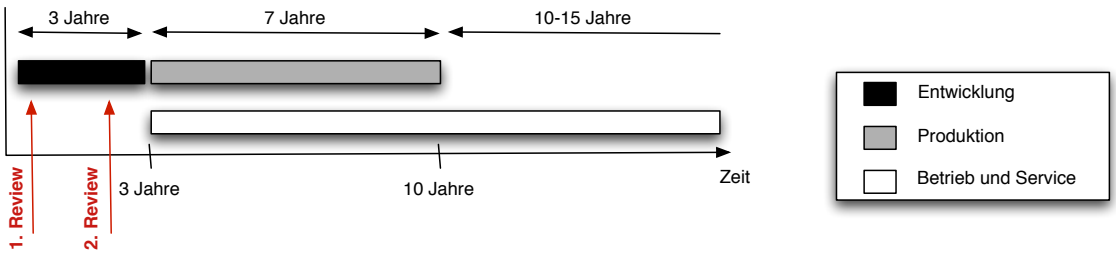


Abbildung 1.4.: Produktlebenszyklus eines Fahrzeuges inklusive Zeitpunkte eines Car Reviews, nach [SZ10, 20f]

2. Gliederung

Zunächst wird in Kapitel II eine Analyse der Anforderungen durchgeführt. Es werden hier neben den geforderten Zielen unter anderem Szenarien, funktionale und nicht funktionale Anforderungen sowie das Systemmodell behandelt.

Anschließend wird in Kapitel III eine umfassende Recherche zu Technologien betrieben, die unsere hier verwendete Plattform, einem iPad mit dem iOS Betriebssystem von Apple, bereitstellt. Dabei werden neben einer kurzen Einführung in die Technologie jeweils Vor- sowie Nachteile untersucht und anschließend passend für die hier gestellte Fragestellung bewertet. Im Detail werden hier Grafikframeworks vorgestellt und gegenübergestellt. Des Weiteren werden mehrere Möglichkeiten zur Lokalisierung eines iPads innerhalb geschlossener Räume gezeigt.

In Kapitel IV wird das System und Objekt Design vorgestellt. Das entwickelte System sowie das Objekt Modell werden hier im Detail gezeigt. Das vorletzte Kapitel V fasst die Ergebnisse dieser Arbeit nochmals zusammen und bietet einen Ausblick für die mögliche Zukunft dieses Projektes. Es werden Erweiterungen angedacht, die in den kommenden Iterationen realisiert werden können.

3. Dokument Konventionen

Innerhalb dieser Arbeit werden folgende Konventionen eingehalten:

- Typographische Konvention

  - Ein neu eingeführter Begriff ist **Fett** geschrieben, wenn das erste Mal darauf Bezug genommen wird, Diese Begriffe sind im Glossar im Anhang zu finden.

- Die Namen von System- und Modellelementen werden in `Courier` dargestellt. Selbiges gilt für Programmcode.

- Notation und Diagramme

  - Es wird **UML** in der Version 2. für alle Modelle und Diagramme innerhalb dieser Arbeit verwendet (Definition siehe [Gro07]).

Teil II.

Anforderungsspezifikation

Dieses Kapitel widmet sich den Anforderungen an die zu entwickelnde Car Review Anwendung, die aus der Problembeschreibung und Szenarien abgeleitet werden. Relevant für das Festlegen der funktionalen und nicht funktionalen Anforderungen ist die Problembeschreibung aus Kapitel 1 sowie einige Szenarien aus Kapitel 5.1.

1. Aktuelles System

Hier wird auf die aktuelle Situation des bestehenden Systems näher eingegangen. Nach Brügge und Dutoit [BD04] wird dies auch als **Ist-Szenario** beschrieben. Anwender werden beobachtet, um dabei Szenarien zu extrahieren um das System und die Aufgabenstellung anschaulich und verständlich zu machen. Auf bestehende Probleme wird dabei ebenfalls eingegangen.

1.1. Papiergebundene Umfrage

Die zwei Reviewer Martina und Tobias befinden sich auf einem Car Review und befragen die Gäste zu ihrer Meinung zu den gezeigten Autos. Als Hilfsmittel verfügen sie über Stift und Papierfragebögen. Das Problem hierbei ist, dass durch die papierbasierten Umfragebögen die Reviewer Martina und Tobias schnell die Übersicht verlieren, welche Personen sie zu welchen Autos bereits befragt haben. Martina ist nicht die ganze Zeit aufmerksam. Sie übersieht zwei Fragen und beantwortet eine falsch. Ihre Dokumentation ist daher fehlerhaft. Da sie sich nicht mit jedem neuen Auto bis in jedes Detail auskennt und sie auch keine Möglichkeit hat, mobil weitere Informationen abzurufen, kann sie nicht jede Frage der Besucher beantworten.

1.2. Datensicherheit

Mario hat bereits einige papierbasierte Umfragen durchgeführt. Diese hat er hintereinander an sein Klemmbrett gespannt. Nachdem er eine kleine Pause gemacht hat, vergisst er es wieder mitzunehmen. Als er danach sucht, findet er es nicht mehr.

Findet eine andere Person die Umfragebögen, hat sie uneingeschränkt Einsicht in die bisherigen Umfrageergebnisse. Dies wollen die Automobilhersteller und damit die Organisatoren des Reviews auf keinen Fall, da die Vertraulichkeit der Daten eine hohe Priorität hat. Primär geht es hierbei um Konkurrenten und dass diese nicht die Ergebnisse abfangen können.

1.3. Auswertung

Stephan ist der Hauptverantwortliche für den Car Review. Seine Aufgabe ist es, nach der Befragung die Auswertung durchzuführen. Hierfür sammelt er alle Befragungsbögen ein und sortiert sie nach den verschiedenen Autos, um seine weitere Auswertung durchzuführen.

Dabei merkt Stephan schnell, dass es sehr aufwendig ist, zuerst die Umfragebögen nach Autos zu sortieren und anschließend die Meinungen auszuwerten. Dabei übersieht er manche Papierbögen und macht einige Fehler, da er nicht jede Handschrift lesen kann. Für die gesamte Auswertung benötigt er viel Zeit.

1.4. Flexibilität

Wenige Stunden vor dem Review wird Stephan, dem Organisator des Car Reviews, mitgeteilt, dass kurzfristig ein weiteres Auto ausgestellt und bewertet werden soll.

Stephan muss nun kurzfristig die Fragebögen mit einer Software abändern oder neu erstellen und für die Reviewer ausdrucken und verteilen. Die aufzubringende Organisationsleistung sowie Koordination und der zeitliche Druck hierbei bereiten Stephan Schwierigkeiten.

1.5. Mehrfache Befragung

Stephan hat mit seinem Team inzwischen eine Fragerunde inklusive Auswertung durchgeführt. Damit ist der erste von zwei Tagen des Car Reviews bereits vorbei. Um auf gewonnene Ergebnisse der ersten Runde näher einzugehen, soll am zweiten Tag eine weitere Fragerunde durchgeführt werden.

Für die Auswertung der Ergebnisse benötigt Stephan viel Zeit. Um diese zu evaluieren und anschließend die von den Besuchern gewünschten Änderungen am Design des Autos vorzunehmen, vergeht erneut viel Zeit. Eine (zumindest) teilweise Automatisierung würde den Vorgang beschleunigen, um gegebenenfalls sogar mehrere Fragerunden an einem Tag durchzuführen und somit das beste Ergebnis aus den Reviews zu erhalten.

2. Ziele

Die im Rahmen dieser Arbeit zu entwickelnde Software soll bei dem Reviewprozess von neuen Autos helfen und diesen Prozess vereinfachen sowie beschleunigen. Es werden Aufgaben übernommen, die von der Eingabe der Umfragewerte bis zur digitalen Übertragung an einen Server reichen, um eine digitale Auswertung zu ermöglichen. Den Reviewern soll ein Hilfswerkzeug zur Hand gegeben werden, mit welchem sie durch den gesamten Prozess einer solchen Befragung geführt werden. Der Reviewer wird Schritt für Schritt durch die Befragung geleitet. Das Problem, dass ein Arbeitsschritt vergessen wird, ist nicht mehr vorhanden. Die Software besteht aus verschiedenen Komponenten und jeweils mit unterschiedlichen Aufgabengebieten:

1. Benutzerverwaltung
2. Modellhandhabung
3. Kommentarsystem
4. Serverschnittstelle

Im folgenden werden diese Komponenten eingeführt und näher erläutert.

2.1. Benutzerverwaltung

Da es sich bei den in Reviews erhobenen Daten um sensible Daten handelt, die nicht jeder einsehen darf, müssen Benutzerrechte eingeführt werden. Dabei sollen selbst erstellte Reviews jederzeit anzeigbar und innerhalb der aktiven Sitzung bearbeitbar sein, fremde Reviews sollen nicht sichtbar sein. Ebenfalls soll es nur den Organisatoren des Car Review möglich sein, Daten wie Umfrageergebnisse oder auch Modelldaten über eine Serververbindung auszutauschen.

2.2. Modellhandhabung

Da der Befragte an verschiedenen Stellen eines Autos dazu aufgefordert wird, Fragen zu beantworten oder seine Meinung zu diesem Bauteil abzugeben, soll ein Modell auf dem Bildschirm helfen, sich besser zu orientieren, wo man sich gerade befindet und was man bereits erledigt hat. Außerdem soll es dem Arbeitsablauf einen spielerischen und optisch ansprechenden Charakter verleihen.

2.3. Kommentarsystem

Die Kernkomponente stellt das Kommentarsystem dar. Darüber soll es möglich sein, vordefinierte Fragen zu beantworten sowie die eigene Meinung mitzuteilen. Dies kann als Freitexteingabe geschehen oder auch als eine Audio- bzw. Videoaufnahme. Das Zusammenspiel von Modell- und Kommentarsystem ist besonders wichtig, denn es soll am Modell visualisiert werden, für welche Autoteile bereits ein Kommentar eingegeben wurde.

2.4. Serverschnittstelle

Über die Serverschnittstelle sollen die Ergebnisse des Reviews von der mobilen Software an einen Server zur Auswertung übertragen werden. Vom Server können über die Schnittstelle Modelle neuer Autos sowie die Fragen für die Umfrage abgerufen werden, um diese in dem lokalen Speicher der Car Review Software zu speichern. Reviewer haben dabei die Berechtigung, ihre Ergebnisse an den Server zu übertragen. Das Herunterladen der Modelle und der Fragen ist den Organisatoren vorbehalten.

3. Funktionale Anforderungen

Im folgenden Kapitel werden die **funktionalen Anforderungen** identifiziert.

Unter funktionalen Anforderungen verstehen Brügge und Dutoit [BD04] eine Beschreibung der „Interaktion des Systems mit seiner Umgebung unabhängig von seiner Implementierung.“ Dabei beschreibt die Umgebung die relevanten Akteure, mit denen das System interagiert.

- **Benutzerverwaltung**
  Die Organisatoren eines Reviews sind in der Lage, bestehende Benutzer zu verwalten, also zu bearbeiten oder zu löschen, sowie neue Benutzer hinzuzufügen. Des Weiteren wird den Reviewern die Möglichkeit gegeben, ihr eigenes Kennwort für den Login abzuändern. Das initiale Passwort wird von einem Organisator vergeben.

- **Kommentieren und Multiple Choice**
  Die Software bietet dem Benutzer die Möglichkeit Kommentare in Form von Freitext, Audio- oder Videoaufnahmen zu verschiedenen Komponenten eines Autos zu verfassen. Des Weiteren soll die Möglichkeit gegeben werden, Multiple Choice Fragen abzurufen und zu beantworten. Alle gegebenen Antworten werden von dem System automatisch gespeichert und sind nachträglich nur von dem Ersteller innerhalb der aktiven Sitzung änderbar. Das Löschen von Kommentaren ist grundsätzlich nicht möglich.

- **Import von Modelldaten**
  Die Software lädt eine Liste der bereitgestellten Modelldaten. Der Benutzer hat die

Möglichkeit, sich die gewünschten Modelle einzeln zu laden. Bereitgestellt werden diese Daten von einem zentralen Server.

- **Export der Umfragedaten**
  Die Software fasst alle gesammelten Daten in einer XML Datei zusammen und überträgt diese anschließend an den zentralen Server. Um den Datenschutz sicherzustellen wird diese Datei über einen SSL verschlüsselten Kanal übertragen.

- **Login und Logout**
  Der Benutzer muss sich vor dem Verwenden der Anwendung mit seinen persönlichen Zugangsdaten verifizieren. Die Software weist den betreffenden Nutzern ihre entsprechenden Rechte zu und stellt damit die Sicherheit sowie den Datenschutz sicher.

- **Multimediaroutine**
  Neben der Darstellung von dreidimensionalen Objekten wird die Funktionalität geboten, mit dem Objekt zu interagieren. Damit ist gemeint, sich Details durch Vergrößerung genauer anschauen zu können beziehungsweise das Objekt durch Drehen von verschiedenen Seiten zu betrachten.

- **Positionsbestimmung**
  Existiert bereits ein Prototyp des zu bewertenden Automodells, bietet die Anwendung die Möglichkeit, selbstständig und ohne weitere Notwendigkeit von Interaktionen des Benutzers aus, die aktuelle Sicht auf den Prototypen in dem digitalen dreidimensionalen Modell auf dem Monitor widerzuspiegeln und dieses passend dazu zu drehen.

4. Nichtfunktionale Anforderungen

Im folgenden Kapitel werden die **nichtfunktionalen Anforderungen** definiert, die für diese Bachelorarbeit erhoben werden.

Bei nichtfunktionalen Anforderungen handelt es sich um „[...] Aspekte des Systems, die nicht unmittelbar in Bezug zu seiner Funktionalität stehen. Nichtfunktionale Anforderungen betreffen verschiedenste Aspekte des Systems, die von der Bedienbarkeit bis hin zur Leistungsfähigkeit reichen.“ [BD04]

Die hier beschriebenen Anforderungen sind wichtiger Bestandteil für die später in Kapitel III durchgeführte Recherche der technischen Realisierungsmöglichkeiten. Sie folgen den **FURPS+**-Kategorien [1].

[1] englisch: Functionality, Usability, Reliability, Performance and Supportability

4.1. Benutzerfreundlichkeit

Das System, das im Zusammenhang mit dieser Arbeit entwickelt werden soll, dient im späteren als Unterstützung des Reviewprozesses eines neuen Automobils. Dabei ist davon auszugehen, dass nicht ausschließlich Fachpersonal als sogenannter Reviewer fungieren. Es ist daher denkbar, dass auch kurzfristig angelerntes Personal diese Aufgabe übernimmt.

Aus diesem Grund ist es besonders wichtig, dass eventuell geringes Vorwissen zu berücksichtigt wird. Um die Benutzung der Software möglichst einfach zu halten, sollte auf den Einsatz von Entscheidungszweigen verzichtet werden und statt dessen der Vorgang eines Reviews vom System linear vorgegeben werden.

Die Benutzeroberfläche orientiert sich dabei an den Designstandards, die von Apple in den Human Interface Guidelines [App11a] vorgegeben werden. Durch diese Richtlinien wird sichergestellt, dass Buttons und Symbole in jeder Software für **Apple iOS** ein einheitliches **Look and Feel** haben und sich Benutzer schnell zurechtfinden.

Die Oberfläche sollte so intuitiv bedienbar sein, dass auch für die erstmalige Nutzung auf eine Dokumentation oder ein Handbuch verzichtet werden kann. Alle verfügbaren Optionen müssen mit maximal drei Klicks erreichbar sein. Über den Button „Hauptmenü" sind jederzeit alle Optionen einsehbar.

4.2. Zuverlässigkeit

Unter Zuverlässigkeit verstehen wir „[...] Richtlinie[n] für Funktionszuverlässigkeit, Instandhaltbarkeit, Sicherheit [und] Verfügbarkeit [...]" [Ver07], aber auch den „Unterschied zwischen vorgeschriebenem und beobachtetem Verhalten" [BD04] des Systems.

Bezogen auf unsere Arbeit heißt dies im Besonderen, dass die Software während eines Reviews keine Fehler erzeugen darf, die zu einem Absturz führen. Es darf nicht vorkommen, dass bereits erhobene Daten verloren gehen beziehungsweise ein Neustart des Systems notwendig ist, da dies die Befragung unnötig stören würde und das Review dadurch gegebenenfalls sogar beendet wäre. Ebenfalls sollte der Export der Daten an einen Server zuverlässig sein, um diese schnell evaluieren zu können. Fehlübertragungen dürfen zu keinen Beeinträchtigungen im Sinne von Abstürzen oder anderen Fehlverhalten der Software führen. Dem Benutzer wird ein Hinweis angezeigt, um über Erfolg oder Misserfolg der Übertragung zu informieren. Im Falle einer Fehlübertragung ist der Export erneut auszuwählen.

Der Wechsel zwischen 3D-Modellen, das umfasst das Entladen des aktuellen 3D-Modells und dem Laden des gewünschten 3D-Modells sowie der dazugehörigen Umfragedaten, darf nicht mehr als fünf Sekunden betragen.

Da gegebenenfalls auch fachlich weniger geschultes Personal diese Software später nutzen soll, muss sichergestellt sein, dass durch ungültige Eingaben des Nutzers keine Fehler auftreten und weiterhin ein zuverlässiger Betrieb möglich ist.

Um dies zu garantieren, muss die Benutzeroberfläche einfach gestaltet sein, um dem Benutzer wenig Auswahlmöglichkeiten bereitzustellen und das Problem der fehlerhaften Nutzereingaben damit zu minimieren oder gar zu eliminieren. Eingegebene Texte werden auf ihre Gültigkeit überprüft. So dürfen sie beispielsweise keine Sonderzeichen enthalten,

um Code Injection [2] vorzubeugen. Sollten solche Zeichen in der Eingabe gefunden werden, werden diese vom System vor der Speicherung entfernt. Der Ablauf eines Reviews ist linear und wird vom System vorgegeben.

Neben dem Anwenderverhalten muss aber auch das Verhalten gegenüber Umwelteinflüssen minimiert werden. Diese spielen jedoch hier eine geringere Rolle, da die Software nach der einmaligen Einrichtung zu Beginn des Reviews bis zum schlussendlichen Export der Daten rein lokal läuft. Allerdings sollte sie auch bei In- und Export von Daten mit dem Server stabil laufen. Sollten Übertragungsfehler auftreten, darf die Software nicht abstürzen oder anderweitige Fehlverhalten aufzeigen. Beispielsweise sollte diese nach einem fehlerhaften Import von Modelldaten den Fehler beim anschließenden Laden bemerken und einen Warnhinweis ausgeben mit der Aufforderung, den Import erneut zu starten.

4.3. Sicherheit

Weiterhin muss sichergestellt sein, dass nur berechtigtes Personal Einsicht in die Umfragewerte erhält. Aus diesem Grund ist festzustellen, ob sich der Benutzer mit ihm persönlich zugewiesenen Zugangsdaten, einem Benutzernamen und einem Passwort, identifizieren kann. Wurde die Anwendung 10 Minuten lang nicht bedient, muss ein automatischer Logout durchgeführt werden.

4.4. Unterstützung und Wartung

Derzeit soll die Anwendung ausschließlich in iOS für das iPad entwickelt werden. Es ist jedoch denkbar, es in der Zukunft für weitere Systeme wie **Android** von Google zu portieren.

Die Wartung, im Sinne vom Einspielen neuer Datensätze wie dreidimensionale Auto Modelle sowie vordefinierte Fragen und Hinweise zu den Autos, soll von den Organisatoren eines Reviews eigenständig und möglichst einfach durchführbar sein. Um zu erreichen, dass nicht nur Modelldaten eines Herstellers unterstützt werden, soll das Power VR POD Format [3] unterstützt werden. Dies ist ein offener Standard für 3D-Modelle. Mit geringem Aufwand kann jeder Format in POD umgewandelt werden. Da mehrere Modelle parallel auf einem Gerät verfügbar sein sollen, ist es wichtig, dass diese Daten streng voneinander getrennt verwaltet und verarbeitet werden.

Eine bereits bekannte Erweiterung für das System ist ein Modul für die Lokalisierung, welches benötigt wird, wenn bereits ein Prototyp von einem Auto besteht (siehe 2. Review in Abbildung 1.4). Dabei soll die Anwendung selbstständig feststellen können, an welcher Stelle des Prototypen sich der Anwender derzeit befindet, um das Modell auf dem Display dementsprechend auszurichten.

[2] bezeichnet das Einschleusen von eigenem Code, der von der Zielanwendung ausgeführt wird

[3] Plain Old Documentation ist eine Auszeichnungssprache, um Dokumentation in Perl zu Programmen hinzuzufügen. Alle gängigen Modellformate lassen sich in das POD Format konvertieren.

4.5. Beschränkungen

Implementierungsanforderungen
Die Anwendung „ReviewApp“ ist an die Plattform iOS von Apple gebunden. Dementsprechend muss sie in Objective-C entwickelt werden (siehe Apple [App10a], [App11b] sowie Dudney und Adamson [DA10]). Eine Entwicklung auf Basis einer anderen Plattform ist ausgeschlossen. Die zugrundeliegende Grafik-Komponente muss diese Anforderung ebenfalls erfüllen.

Schnittstellenanforderungen
Unter dieser Eigenschaft verstehen Brügge und Dutoit [BD04] die „Eigenschaft die durch externe Systeme vorgegeben werden“. Es wird eine Menge von öffentlichen Methoden, Merkmalen und Verpflichtungen beschrieben, die durch die Schnittstelle zwingend bereitgestellt werden müssen.

Die Anwendung bietet eine Schnittstelle zu einem entfernten Server, um für die Anwendung relevante Daten zu exportieren oder zu importieren.

5. System Modell

5.1. Visionäre Szenarien

Szenarien oder auch „preiswerte Prototypen“ sind nach Brügge und Dutoit [BD04] Instanzen von Anwendungsfällen (behandelt in Kapitel 5.2), wobei der Fokus auf spezifischen und oft auftretenden Fällen liegt. Weiterhin liegt das Ziel darin, den Fall möglichst verständlich zu formulieren.

Die aktuelle Situation des bestehenden Systems, basierend auf Umfragen mit Papierbögen, wurde bereits im Kapitel 1 erläutert. Im Folgenden werden die visionären Szenarien, die das zu entwickelnde System beschreiben, im Detail erläutert.

- **Szenario: Download**
  Der Organisator des Car Reviews, Stephan, bekommt kurz vor Beginn des Reviews die Information, dass ein weiteres Auto ausgestellt und in dem Review berücksichtigt werden soll. Das entsprechende digitale Modell ist bereits auf dem Server verfügbar. Er möchte es daher mit möglichst geringem Aufwand in die lokale Anwendung laden. Der Unterschied zum aktuellen System wird in Abschnitt 1 als Ist Szenario „Flexibilität“ beschrieben.

- **Szenario: Datenexport**
  Nachdem die Veranstaltung beendet ist, soll die Auswertung der Ergebnisse beginnen. Stephan benötigt dazu alle Umfragedaten auf seinem PC. Um die Übertragung zu starten, reicht es aus, in der mobilen Software den Export auszuwählen.

- **Szenario: Kommentieren**
  Hans soll das erste Mal ein Review durchführen, er ist noch nicht ganz mit dem Ablauf vertraut. Doch das System führt ihn sicher durch den gesamten Ablauf und gibt Warnhinweise aus, sollte er etwas übersehen haben. Außerdem möchte er nicht alle Bemerkungen aufschreiben, er hat allerdings auch die Möglichkeit, Video- und Audioaufnahmen abzuspeichern. Der Unterschied zum aktuellen System wird hier in Abschnitt 1 als Ist Szenario „papiergebundene Umfrage“ beschrieben.

- **Szenario: Multiple Choice**
  Da sich Franziska nicht mit Autos auskennt, fällt es ihr schwer, die Aussagen der Besucher als sinnvolle Kommentare zu formulieren. Daher bietet ihr das Multiple Choice System die Möglichkeit, Fragen und die dazu passende Antwortmöglichkeiten vorzugeben. Es können je nach Frage eine oder mehrere Antworten angegeben werden. Auch hier beziehen wir uns wie beim vorherigen Szenario auf „papiergebundene Umfragen“ aus dem Abschnitt 1.

- **Szenario: Informationen zeigen**
  Michael werden spezifische Frage zu den Autos gestellt. Obwohl er sich selbst nicht ausreichend auskennt, kann er mit Hilfe der Software diese Fragen beantworten. Beispielsweise bei Fragen zu den Felgen hat er die Option, auf einen Informationsbutton zu klicken, der mit einem „i“ gekennzeichnet ist. Er erhält die Information, dass die Felgen aus Aluminium sind.

5.2. Anwendungsfälle

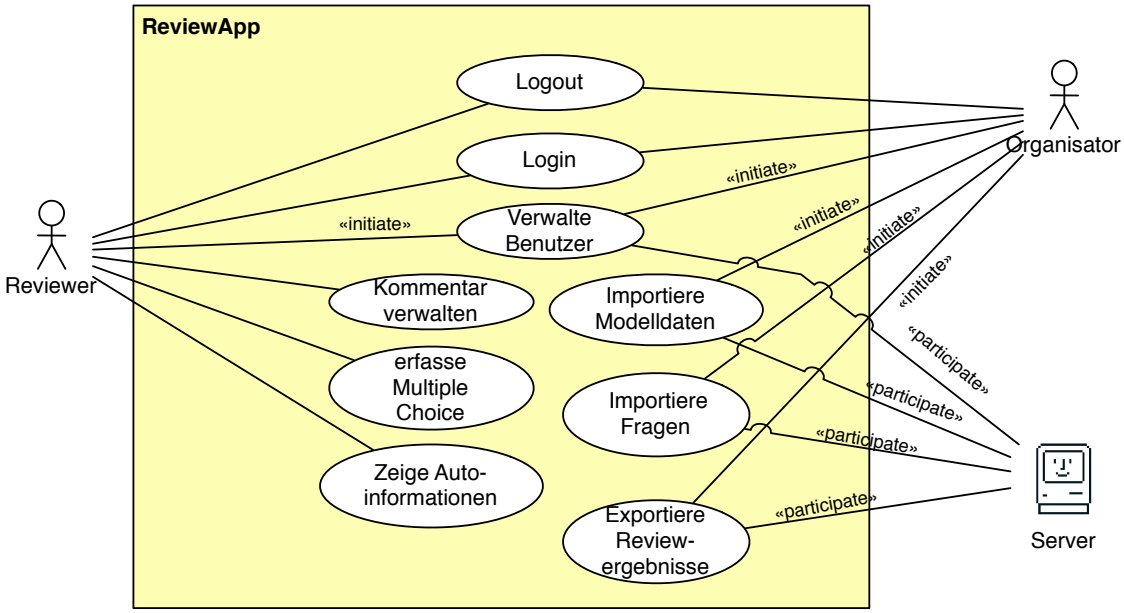


Abbildung 5.1.: Übersicht der verschiedenen Anwendungsfälle, die für die Review App relevant sind

Im vorherigen Abschnitt wurden Szenarien formuliert, aus denen sich abstrakte Anwendungsfälle modellieren lassen. Hier wird die Interaktion der Anwender mit dem System beschrieben. Im Zuge der Anforderungsermittlung wurden Reviewer, **Organisatoren** und Server als die Akteure identifiziert.

In Abbildung 5.1 sehen wir eine Übersicht aller High Level Anwendungsfälle, auf die im Weiteren tiefer eingegangen wird.

5.2.1. Verwalte Kommentare

Zu Beginn des Anwendungsfalles „Kommentieren“, im Detail in Abbildung 5.2 zu sehen, hat sich der Reviewer bereits mit seinen Benutzerdaten verifiziert und ein Modell ausgewählt, welches ihm auf dem Display angezeigt wird.

Um einen Kommentar zu der gewünschten Stelle an dem Modell einzugeben, tippt er mit dem Finger an dieses Autoteil, an welchem sich ein kleines „i“ befindet. Das System antwortet ihm mit einer Eingabeaufforderung, bei welcher man auswählen kann um welche Art eines Kommentars es sich hierbei handeln soll.

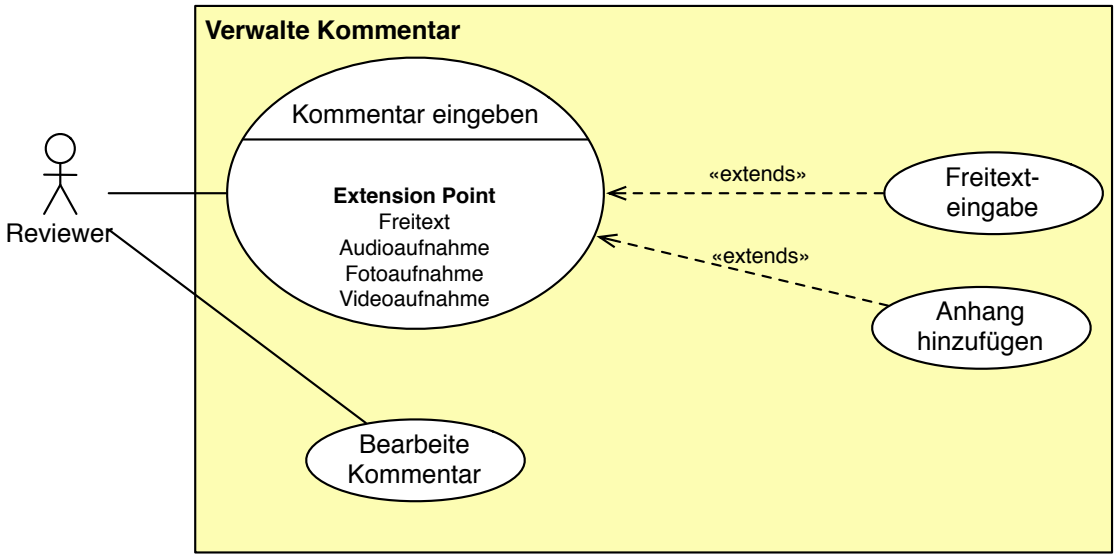


Abbildung 5.2.: UML Diagramm des Anwendungsfalls „Kommentieren“

Der Anwender hat hierfür vier verschiedene Optionen:

1. Freitext
2. Audioaufnahme
3. Foto
4. Videoaufnahme

Bei Option 1 kann ein beliebiger Freitext eingegeben werden, der die Meinung des Betrachters zu diesem Bauteil widerspiegeln soll.

Option 2 bietet die Möglichkeit, eine Tonaufnahme zu machen, bei welcher der Betrachter seine Meinung diktieren kann.

Die Optionen drei und vier sind nur verfügbar, wenn es sich um ein iPad der neusten Generation handelt, welches über eine Kamera verfügt. Derzeit ist dies das iPad in der Version 2. Dabei wird ein Video beziehungsweise ein Foto aufgenommen, das die betreffende Stelle im Detail zeigt. Bei Option 4 wird dabei ein bewegtes Bild mit einem Audiokommentar kombiniert.

Nach Beendigung des Kommentars bestätigt der Anwender seine Eingabe mit einem Druck auf den Knopf „Fertig“. Das System schließt die Eingabeaufforderung und speichert diesen Kommentar automatisch. Dabei wird bei korrektem Ablauf an dem Modell ein Hinweis angezeigt, dass hier bereits ein Kommentar vorliegt. Alle Textinformationen werden in einer XML Datei abgelegt. Ton-, Video- und Bildaufnahmen werden als separate Dateien abgelegt und in der XML Datei verlinkt.

5.2.2. Erfasse Multiple Choice

Im Anwendungsfall „Multiple Choice“ hat der Reviewer die Möglichkeit, aus einer Liste, in der alle verfügbaren Fragen gelistet sind, zu wählen. Voraussetzung zum Öffnen der Liste ist, dass der Reviewer sich eingeloggt hat und sich im Hauptmenü befindet. Dort ist der Punkt „Multiple Choice“ auszuwählen. Alternativ dazu können Multiple Choice Fragen zu einem gewünschten Bauteil des Autos beantwortet werden, indem die jeweilige Stelle des Autos angeklickt wird.

Nach der Auswahl einer Frage lädt das System alle dazugehörigen Antworten aus einer XML Datei und zeigt diese an. Wählt der Anwender nun eine oder mehrere der passenden Antworten aus, muss dies mit „Fertig“ bestätigt werden. Das System schließt daraufhin die Eingabeaufforderung der Frage und zeigt die Übersicht an. War die Eingabe und die Speicherung korrekt, wird die Frage nun mit einem grünen Haken als beantwortet markiert. Andernfalls wird sie rot hervorgehoben, um auf einen Fehler oder fehlende Beantwortung hinzuweisen.

5.2.3. Zeige Autoinformationen

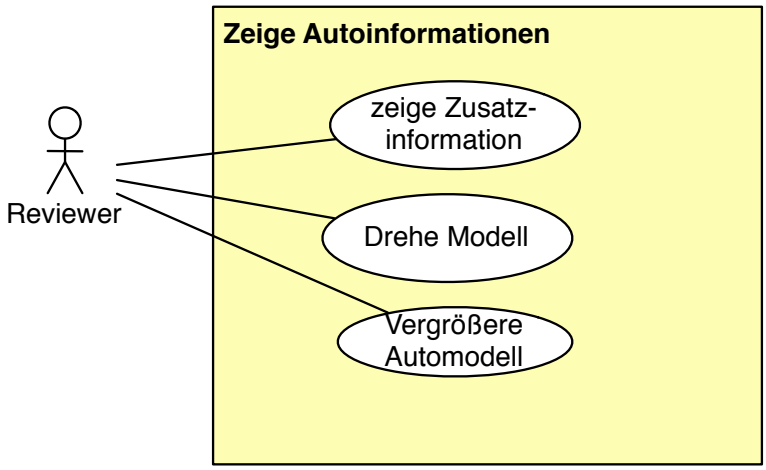


Abbildung 5.3.: UML Diagramm des Anwendungsfalls „Zeige Autoinformationen“

Abbildung 5.3 zeigt die Möglichkeiten des Reviewers, um weitere Informationen zu dem Auto beziehungsweise dem Modell zu erlangen. Dazu muss er sich erfolgreich verifiziert haben und das Modell geöffnet haben.

Durch die Auswahl eines Autoteiles, welches mit einem „i“ Symbol markiert ist, soll das System ein Fenster öffnen, welches weitere Informationen dazu anzeigt. Alternativ können alle verfügbaren Informationen über das Hauptmenü erreicht werden.

Dazu hat der Anwender auch die Möglichkeit, durch Wischgesten das Modell zu drehen oder zu zoomen.

Das System schließt dieses Informationsfenster erst, wenn der Anwender dies wünscht und zum Schließen das in der Apple Human Interface Guideline beschriebene „Fertig“

bestätigt.

5.2.4. Verwalte Benutzer

Der Anwendungsfall „Verwalte Benutzer“, zu sehen in Abbildung 5.4, beschreibt die Möglichkeiten für Organisatoren, neue Benutzer anzulegen oder Bestehende zu bearbeiten oder auch zu löschen. Reviewer haben dabei nur die Berechtigung, ihr eigenes Kennwort zu ändern.

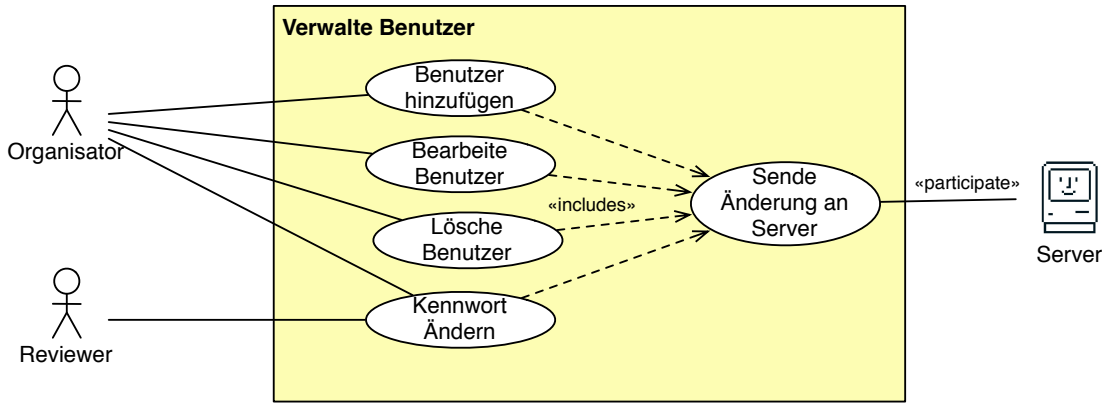


Abbildung 5.4.: UML Diagramm des Anwendungsfalls „Verwalte Benutzer“

Als Voraussetzung muss gelten, dass der jeweilige Anwender sich bereits verifiziert hat und sich im Verwaltungsmenü befindet, welches automatisch vom System nach dem Login angezeigt wird. Dort wird unter anderem der Menüpunkt „Verwaltung Benutzer“ angezeigt, welcher auszuwählen ist.

Dort werden, je nach jeweiliger Benutzerberechtigung, die gewünschten Optionen „Füge Benutzer hinzu“, „Bearbeite Benutzer“ und „Lösche Benutzer“ für Organisatoren oder für Reviewer „Bearbeite eigenes Benutzerkonto“ angezeigt.

Beim Hinzufügen von neuen Benutzern oder der Änderung des Passwortes werden vom System nach Bestätigung der gewünschten Option Eingabefelder angezeigt, welche vom Anwender auszufüllen sind. Nach Beendigung der Eingabe muss der Anwender dem System dies mit der Bestätigung des Buttons „Fertig“ signalisieren. Er wird anschließend automatisch zurück in die Benutzerverwaltung gebracht.

Beim Bearbeiten oder Löschen von Benutzern zeigt das System zunächst eine Liste aller verfügbaren Benutzern. Nach Auswahl des gewünschten Benutzers wird beim Bearbeiten erneut eine Eingabemaske angezeigt, bei der Benutzername, Berechtigung sowie Kennwort geändert werden können. Beim Löschen fragt das System einmalig, ob man diesen Benutzer unwiderruflich löschen möchte. Wird dies bestätigt, löscht das System den Benutzer aus der lokalen XML Datei und führt den Anwender zurück in die Benutzerverwaltung. Des Weiteren werden alle Änderungen an den Server mitgeteilt.

5.2.5. Importiere Modelldaten

Um ein neues Modell in das System zu laden, muss sich der Reviewer erfolgreich eingeloggt haben. Anschließend ist im Hauptmenü der Punkt „Importiere Modelldaten“ auszuwählen. Das System fordert daraufhin den Server auf, eine Liste aller verfügbaren Modelle zu senden. Im Erfolgsfall zeigt das System diese Liste im Anschluss dem Anwender an. Aus dieser Liste kann je Importvorgang jeweils ein Modell gewählt werden. Ohne weitere Bestätigung des Anwenders fordert das System den Server daraufhin auf, das Modell zu übertragen. War die Übertragung erfolgreich, wird dies dem Benutzer durch ein PopUp angezeigt, welches nach kurzer Zeit automatisch wieder verschwindet. Das System speichert das Modell in seinem lokalen Speicher und trägt es in eine XML Datei ein.

Der Anwender wird zurück zu der Liste der Modelle geführt, woraus er erneut ein Modell laden oder sich zurück in das Hauptmenü begeben kann.

Bei einer fehlerhaften Übertragung der Liste der verfügbaren 3D-Modellen oder des 3D Modells selbst, wird dies dem Benutzer mitgeteilt. Der Importvorgang muss daraufhin vom Benutzer erneut gestartet werden.

5.2.6. Exportiere Reviewergebnisse

Um nach dem Car Review die gesammelten Reviewergebnisse zur weiteren Auswertung an einen Server zu übertragen, wählt der Reviewer nach erfolgreichen Login aus dem Verwaltungsmenü den Punkt „Exportiere Reviewergebnisse“ aus. Es wird eine lokal angelegte XML Datei an den Server über eine verschlüsselte SSL Verbindung übertragen. War die Übertragung erfolgreich, wird eine Bestätigung angezeigt. Selbiges gilt für den Fehlerfall, bei welchem der gesamte Vorgang zu wiederholen ist.

6. Analyse Objektmodell

In diesem Kapitel wird das konzeptionelle Modell des Systems vorgestellt. Die hier beschriebenen Klassen und Objekte wurden während der Anforderungsermittlung aus den vorherigen Kapiteln dieses Abschnittes ermittelt.

Im Folgenden wird daher zunächst auf das Klassendiagramm eingegangen und im Anschluss auf das spezifischere Objektdiagramm.

6.1. Klassendiagramm

Nach Rupp, Queins und Zengler [RQZ07] sind Klassen die Abstraktion der identifizierten Objekten. „Dieser Objekttyp stellt das Innenleben der Objekte, d.h. ihre Attribute und Operationen sowie deren Beziehungen nach außen, d.h. Generalisierungs-, Assoziations- und Abhängigkeitsbeziehungen, dar.“ [RQZ07]

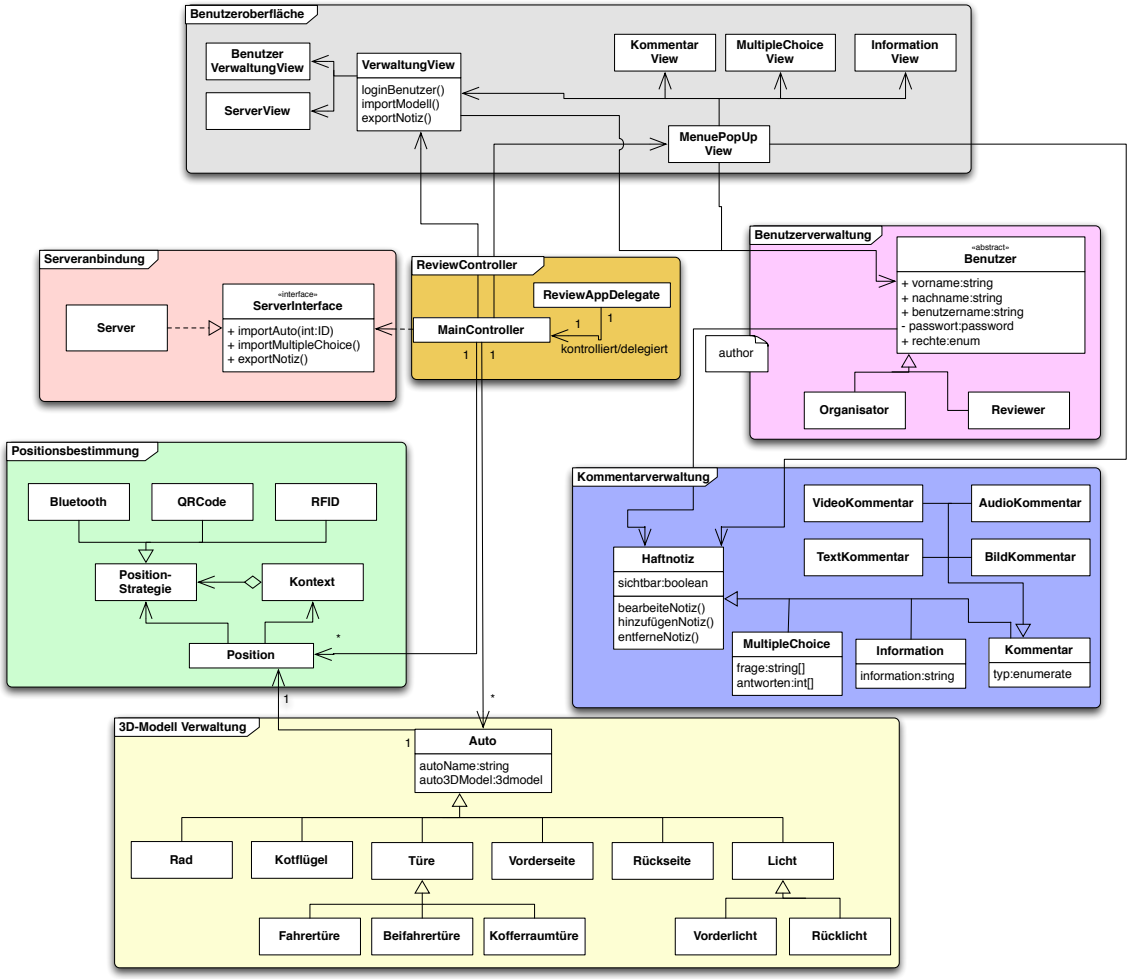


Abbildung 6.1.: Identifizierte Klassen

In Anlehnung auf die in Kapitel 2 definierten Komponenten, werden die in Abbildung 6.1 gezeigten Klassen identifiziert. Im Folgenden werden diese in die sieben Kategorien eingeteilt, die gefordert waren.

- **Kommentarverwaltung**
  Das zentrale Element der ReviewApp ist das Kommentarsystem. In Abbildung 6.1 ist dieses Modul durch „Haftnotiz“ dargestellt. Zu seinen Aufgaben gehört das Anlegen neuer Notizen, welche aus einem Kommentar wie Videos, Bildern, Audioaufnahmen oder Freitexten bestehen sowie Multiple Choice Fragen und Antworten enthalten können. Dabei ist die Klasse Haftnotiz eine Generalisierung für (Bauteil-) Information, Kommentar sowie MultipleChoice.

- **Benutzerverwaltung**
  Die Benutzerverwaltung identifiziert die Klasse „Benutzer“. Diese ist für die persistente Verwaltung der Benutzer verantwortlich.

- **3D-Modell Verwaltung**
  Die Klasse „Auto“ beschreibt das 3D-Modell. Die Klasse Auto ist eine Generalisierung und beschreibt eine Vielzahl von weiteren Klassen, den einzelnen Bauteilen. und fasst sie als eine Einheit zusammen. Sie ist auch dafür zuständig, dass das Modell korrekt angezeigt wird.

- **Serveranbindung**
  Die Serveranbindung wird durch das Interface „ServerInterface“ vertreten. Hierüber findet der Datenaustausch zwischen dem „MainController“ und dem Server statt.

- **ReviewController**
  Der ReviewController ist die zentrale Steuerung der Anwendung. Mit der Klasse „MainController“ stellt er die Verbindung zu einem Server her, steuert die Darstellung der Benutzeroberfläche und verwaltet das 3D-Modell. Er bietet auch die Möglichkeit, mit dem 3D-Modell zu interagieren. Die Klasse „ReviewAppDelegate“ ist die steuernde Klasse, die den MainController startet und beendet.

- **Positionsbestimmung**
  Die identifizierte Klasse „Position“ bietet unterschiedliche Strategien, um die aktuelle Position des Gerätes, auf dem die Anwendung läuft, relativ zu einem realen Prototypen zu ermitteln.

- **Benutzeroberfläche**
  Die zwei Klassen „VerwaltungView“ sowie „MenuePopUpView“ stellen die zwei Menüs dar, die in dieser Anwendung vorgesehen sind. Das PopUp Menü ist jederzeit aufrufbar. Darüber sind alle weiteren Benutzeroberflächen aufrufbar.

6.2. Objektdiagramm

Ein Objektdiagramm gibt die Möglichkeit, Instanzen von Klassen, Komponenten, Knoten, Multiplizitäten, Assoziationen und Attributen in einem „Schnappschuss des Systems“ aus Kapitel 6.1 zu einem gewissen Zeitpunkt zu modellieren [RQZ07].

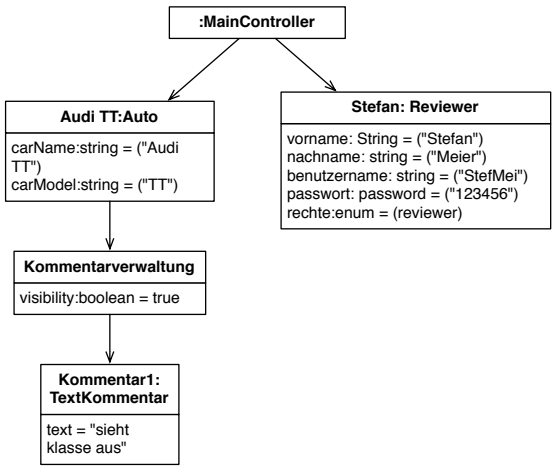


Abbildung 6.2.: Konzeptionelles Objektdiagramm für das Kommentieren einer Autotür

In Abbildung 6.2 sehen wir ein Objektdiagramm, in welchem der Reviewer Stefan Meier einen Kommentar für die Fahrertür eines roten Audi TT anlegt. Hier wird erkennbar, wie aus dem Klassendiagramm aus Abbildung 6.1 ersichtlich ist, dass der MainController die oberste Instanz ist, welche alle weiteren Objekte verwaltet.

In Abbildung 6.3 ist der Export der Umfragewerte an den Server zu sehen. Dieser wird von dem Organisator Hans Mustermann angestoßen.

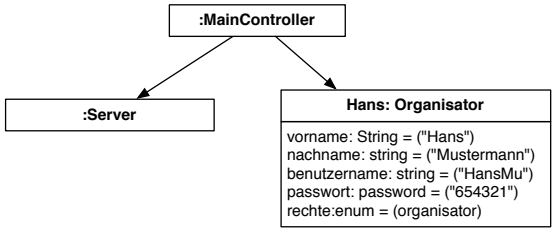


Abbildung 6.3.: Konzeptionelles Objektdiagramm für den Export der Umfragedaten

6.3. Sequenzdiagramm

Ein Sequenzdiagramm verknüpft Anwendungsfälle mit Objekten [BD04]. Es zeigt dabei das Verhalten eines Anwendungsfalles und die daran beteiligten Objekte und beschreibt den Austausch von Nachrichten. So ist es Entwicklern möglich, fehlende Objekte in der Anforderungsspezifikation zu finden.

In Abbildung 6.4 wird mittels Sequenzdiagramm der Anwendungsfall „verwalte Kommentar“ dargestellt, genauer gesagt wird hierbei die Fahrertüre betrachtet und ein Freitext Kommentar eingegeben. Dazu klickt der Reviewer zunächst auf die Fahrertüre des 3D-Modells. Das System antwortet mit einem PopUp. Hier werden die unterschiedlichen Optionen für die Fahrertüre angezeigt. Nach der Auswahl „Kommentar“ des Benutzers, wird er zur Eingabe des Freitextkommentars aufgefordert. Nach der Beendigung der Eingabe wird der Kommentar in der Kommentarverwaltung persistent gespeichert. Der Reviewer erhält die Meldung, dass der Speichervorgang erfolgreich war.

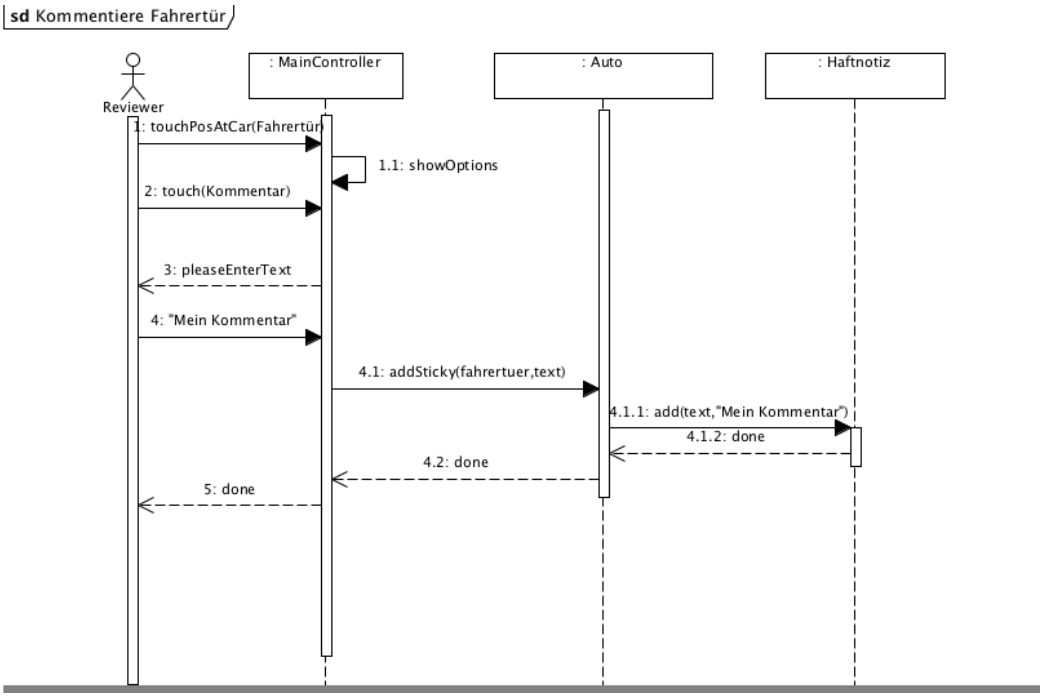


Abbildung 6.4.: Sequenzdiagramm für das Kommentieren einer Autotür

Teil III.

Technologie Recherche

In diesem Kapitel werden Technologien vorgestellt, die für diese Arbeit relevant sein können. Dabei werden jeweils die Vor- und Nachteile aufgeführt und diese im Anschluss gegenseitig abgewogen, um eine sinnvolle Entscheidung treffen zu können.

Die wichtigste aller verwendeten Technologien ist die mobile Plattform iOS von Apple. Dies liegt daran, dass diese Anwendung im späteren auf dieser Plattform lauffähig sein soll. Grundsätzlich ist jedoch die Portierung auf andere mobile Plattformen nicht ausgeschlossen (siehe Kapitel V). Es wird eine (3D-)Visualisierung benötigt die durch ein Grafik-Framework bereitgestellt wird. Darauf stützt sich die Implementierung dieser Anwendung.

Durch iOS und ein Grafik-Framework erfüllt sich die identifizierte Anforderung einer dreidimensionalen Komponente (siehe Abschnitt II.3 Nichtfunktionale Anforderungen „Multimediaroutine“).

Für die Persistierung der erhobenen Daten wird XML in Betracht gezogen.

Im Anschluss daran wird auf Möglichkeiten eingegangen, um die Lokalisierung des Gerätes anhand eines Autos realisieren zu können.

1. 3D Frameworks

Für diese Arbeit wurden zwei 3D Frameworks, ISGL3D [1] und Cocos3D [2], näher betrachtet. Beide Frameworks bieten den Vorteil, dass sie in Objective-C geschrieben sind und somit Programmcode in C sowie komplexe Berechnungen für OpenGL komplett vom Entwickler fern halten. Entwickler können so schneller und einfacher arbeiten als direkt mit OpenGL. Auch ist die Darstellung von Modellen mit ähnlich geringem Aufwand realisierbar. Einen weiteren Vorteil bietet die Verbindung zum iOS runtime system. So implementieren diese Frameworks beispielsweise eine spezielle Form von „touchesMoved“.

Jedoch ist die Unterstützung beider Frameworks für Modelldaten etwas eingeschränkt. Es wird jeweils nur das Format „Power VR POD“ unterstützt. Dabei handelt es sich um Perl Dateien.

Die Konvertierung in dieses Format ist aus den meisten bekannten anderen Formaten möglich. Beispiele hierfür sind Motion Capture, Scalable Vector Graphics, 3D Studio, Wavefront sowie Collada.

ISGL3D ist im Vergleich zu Cocos3D relativ unbekannt. Die Entwicklergemeinde ist daher deutlich kleiner als bei Cocos3D. Dies zeigt sich nicht nur an der besseren Dokumentation, sondern auch an den regelmäßigen Weiterentwicklungen.

Beide Frameworks stehen unter der **MIT** Lizenz [Ope] [3] und sind daher frei verwendbar. Die Wahl für ein Grafikframework für diese Anwendung fällt auf Cocos3D, da die Stabilität und Zuverlässigkeit im direkten Vergleich zu ISGL3D derzeit größer ist. Des Weite-

[1] Link: `http://isgl3d.com/`
[2] Link: `http://brenwill.com/cocos3d/`
[3] Link: `http://www.opensource.org/licenses/mit-license.php`

ren ist auch die detailliertere Dokumentation ein Grund für diese Entscheidung. Entwickler, die neu in der Programmierung von Grafikanwendungen sind, können sich durch die größere Entwicklergemeinde einfacher Ratschläge suchen. Oft finden sich auch Lösungen auf Problemstellungen anderer Entwickler, die dem eigenen Problem ähnlich sind. Dies hilft vor allem in der den ersten Einarbeitungsphasen.

1.1. Cocos3D

Cocos3D ist eine Erweiterung zu dem 2D Framework Cocos2D [4], das auf **OpenGL** [5] und seine Spezifikation OpenGL ES [6] aufbaut. Bei OpenGL ES handelt es sich um eine vereinfachte Version der OpenGL Spezifikation, die sich besonders für den Einsatz in eingebette Systeme eignet [Gro]. Daher müssen beide Frameworks in das Zielprojekt eingebunden werden, um lauffähig zu sein. Das Hauptaugenmerkt dieses Frameworks liegt in der Verwaltung und Kontrolle grafischer Elemente.

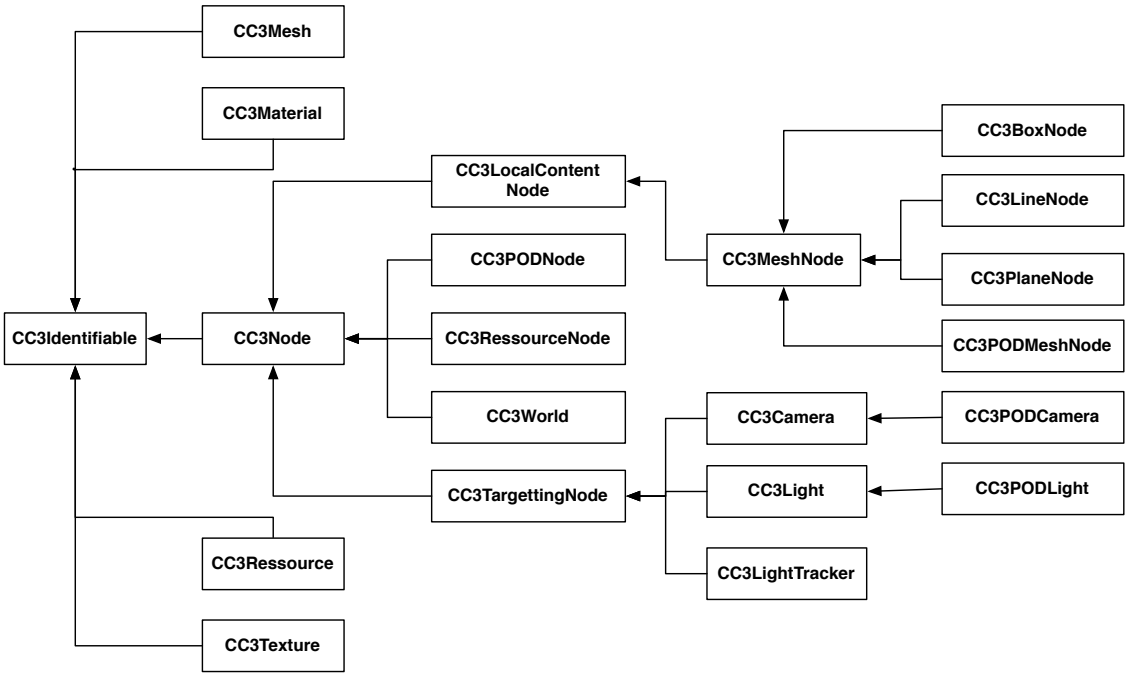


Abbildung 1.1.: *CC3Identifiable* repräsentiert und verwaltet alle Klassen, die mit Namen und Tags individualisiert werden können, nach [Ltd11]

[4] Link `http://www.cocos2d-iphone.org/`

[5] Link `http://www.opengl.org/`, eine plattform- und programmiersprachenunabhängige Schnittstelle für 2D und 3D Grafikelemente

[6] Link: `http://www.khronos.org/opengles/`

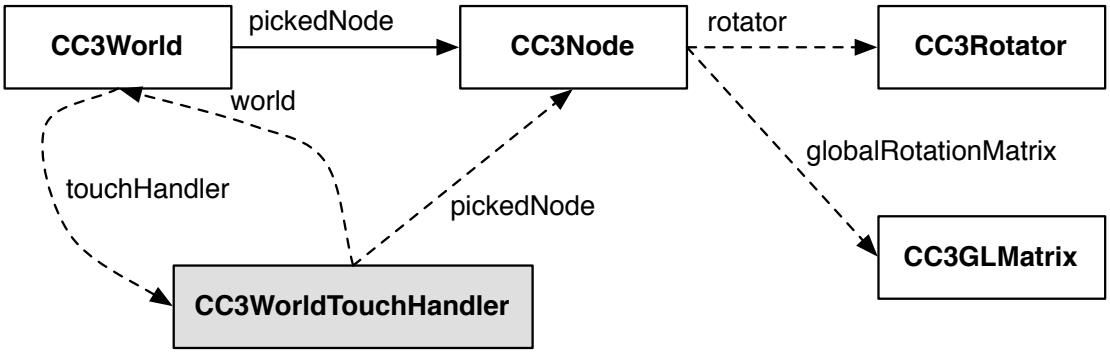


Abbildung 1.2.: der `CC3WorldTouchHandler` ermöglicht Manipulationen der `CC3Node` Klassen in ihrer jeweiligen `CC3World`, nach [Ltd11]

Der Umfang aller Klassen aus Cocos3D zu zeigen würde den Umfang dieser Arbeit sprengen. Daher wurden beispielhaft repräsentative Klassen für die wesentlichen Aufgaben des Frameworks ausgewählt. Die in den Abbildungen 1.1 und 1.2 gezeigten Klassendiagramme veranschaulichen das Konzept der Verwaltung und Kontrolle grafischer Elemente.

In Abbildung 1.1 ist die Superklasse `CC3Identifiable` mit einigen wichtigen Subklassen gezeigt. Es werden eine Menge von grafischen Objekten gebündelt, damit ist `CC3Identifiable` die Hauptklasse aller anderen Klassen, die sich über Namen oder Tags identifizieren lassen. Dies ist eine unerlässliche Funktionalität eines Grafikframeworks.

In Abbildung 1.2 ist der `CC3WorldTouchHandler` zu sehen, mit dem Interaktionen in der jeweilige Welt ermöglicht werden. Auftretende TouchEvents werden von ihm an die `CC3World` weitergereicht.

2. Persistierung

Da persistierte Daten mit einem Server ausgetauscht werden, ist die Vorauswahl auf **XML** [1] als Speichermöglichkeit gefallen. Der XML Standard [2] wird vom W3C [3] spezifiziert und beschreibt die Darstellung hierarchisch strukturierter Daten in Form von Textdaten. Mit XML können beliebige Daten dargestellt werden. [Con]

[1] englisch: Extensible Markup Language
[2] Link `http://www.w3.org/standards/xml/`
[3] englisch: World Wide Web Consortium

	A	B	C	D	E	F
1	**Datensatz**	**Vorname**	**Nachname**	**Strasse**	**PLZ**	**Ort**
2	**1**	Hans	Mustermann	Musterweg 1	12345	Musterstadt
3	**2**	Franz	Meier	Hauptstrasse 5	54321	Neustadt

Abbildung 2.1.: tabellarische Darstellung von Adressdatensätzen

In Abbildung 2.1 sind zwei Adressdatensätze dargestellt. Diese triviale Darstellung lässt sich semantisch in XML übersetzen. Im folgenden sind die selben Daten in XML abgebildet:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ROOT-ELEMENT -->
<adressen>
        <!-- Der 1. Datensatz -->
        <datensatz>
                <!-- Elemente eines Datensatzes sind VORNAME, NACHNAME usw. -->
                <vorname>Hans</vorname>
                <nachname>Mustermann</nachname>
                <strasse>Musterweg 1</strasse>
                <plz>12345</plz>
                <ort>Musterstadt</ort>
        </datensatz>
        <!-- Der 2. Datensatz -->
        <datensatz>
                <vorname>Franz</vorname>
                <nachname>Meier</nachname>
                <strasse>Hauptstrasse 5</strasse>
                <plz>54321</plz>
                <ort>Neustadt</ort>
        </datensatz>
</adressen>
```

Um XML in einer Anwendung lesend sowie schreibend verwenden zu können, wird ein entsprechender XML Parser benötigt, da der NSXML Parser von Apple XML Daten nur unkomfortabel schreibend genutzt werden kann. Es sind mehrere XML Parser als Framework frei verfügbar, wie beispielsweise TouchXML.

Eine Alternative zu nativem XML sind Property Lists, wie im Folgenden genauer beschrieben.

2.1. Property List

Bei einer Property List handelt es sich um ein XML Format, das von Apple auf all seinen Plattformen verwendet wird [App10b]. Durch das hierbei verwendete „Key-Value“ [4] Verfahren und die in das Betriebssystem integrierte Property List Schnittstelle ist ein einfacher und schneller Zugriff auf die Daten sichergestellt. Werden einfache Datenstrukturen wie Integer, String und Boolean verwendet, ist die Nutzung von Property Lists effizienter als XML. Im Normalfall werden Property Lists für die Konfiguration von Software auf Mac OSX oder iOS verwendet. Für die Verwendung von Property Lists keine weitere Einrichtung notwendig.

Der Einarbeitungsaufwand zur Nutzung von Propertylists ist für den Entwickler ist sehr gering. So kann beispielsweise mit dem Befehl `[NSDictionary writeToFile:filepath atomically:YES]` ein Dictionary als PropertyList gesichert werden. Das Auslesen einer Property List ist ähnlich einfach. Mit `[NSDictionary dictionaryWithContentsOfFile: plistPath]` wird der Inhalt in NSDictionary geladen.

[4] jedem Wert ist ein eindeutiger Schlüssel zugewiesen

Die von der Anwendung erstellten Property Lists sind ohne weiteren Aufwand von jedem Computer, wie jede XML Datei, lesbar und somit auswertbar. Es besteht jedoch auch die Möglichkeit, einen Plist-Reader zu verwenden. Dieser kann die Daten aus der Property List tabellarisch wie in Abbildung 2.1 übersichtlich und komfortabel darstellen.

Ein Nachteil vom Property Lists ist die bereits erwähnte Key-Value Form. Dadurch ist es nur sehr schwer möglich, Objekte nachzubilden und zu persistieren.

Die Anwendung, die für diese Bachelorarbeit entwickelt wird, wird daher für die Persistierung der erhobenen Daten Property Lists verwenden.

3. Lokalisierung

Lokalisierung soll in dieser Anwendung dazu genutzt werden, um feststellen zu können, wo sich der Reviewer mit der Anwendung im Verhältnis zu einem realen Prototypen eines Autos befindet. Ist der aktuelle Standort beispielsweise die Fahrertüre, so soll die Anwendung dies feststellen können und auf dem Monitor eine dementsprechende Reaktion zeigen. Dies kann visuell durch Drehung des 3D-Modells zur Fahrertür dargestellt werden.

Mögliche Alternativen um dies feststellen zu können sind im folgenden aufgeführt.

3.1. CoreLocation

CoreLocation ist fester Bestandteil des iOS. Damit ist es einfach möglich, die aktuelle Position des Gerätes herauszufinden. Berücksichtigt werden dabei Werte aus WLAN- und Mobilfunknetztriangulation sowie von **GPS**. Allerdings hat der Benutzer beziehungsweise der Entwickler wenig Einfluss darauf, wie diese Werte abgerufen werden. Weiterhin ist es nicht möglich, eigene WLAN Stationen anzugeben, da diese von einem Server von Apple bezogen werden.

Durch die unzureichende Genauigkeit von GPS- sowie GSM-Empfang innerhalb von Gebäuden, ist dies keine Option für dieses Projekt.

3.2. RFID

Ein System, das **RFID** nutzt, benötigt neben einem Lesegerät, wie es auch in Abbildung 3.1 zu sehen ist, sogenannte RFID Tags. Dabei handelt es sich um Transponder, auf denen Daten gespeichert werden. Ausgelesen werden die Transponder über ein elektromagnetisches Wechselfeld, das von dem Lesegerät erzeugt wird [Wan06].

‚Im [Gegensatz] zum Barcode zeichnet sich RFID dadurch aus, dass die Maschinenlesbarkeit ohne Sichtkontakt gegeben ist, dass viele RFID-Transponder im Lesefeld quasi gleichzeitig erfasst werden können und dass prinzipiell eine größere Datenmenge auf RFID-Transpondern gespeichert werden kann.‘ [DD10, S. 02]

Es gibt verschiedene Arten von Transpondern. Passive Transponder haben keine eigene Stromversorgung und dadurch auch keine so große Reichweite. Dabei wird der Chip von

Abbildung 3.1.: Abbildung eines iPhones mit angeschlossenem RFID Lesegerät, Quelle: [Arn09]

dem Lesegerät bestrahlt und wirft eine Reflektion zurück, in dem die auf ihm gespeicherten Daten kodiert sind. Um dabei ein zuverlässiges Ergebnis zu erhalten, muss die für die Bestrahlung des Transponders aufgebrachte Energie 1000 mal höher sein als die für die Antwort benötigte.

Im Gegensatz dazu stehen die aktiven Transponder, die über eine eigene Stromversorgung verfügen [Wan06], was ihre Reichweite erhöht, da sie nicht abhängig von der Energieversorung des Lesegerätes sind.

Dadurch unterscheiden sich auch die erreichbaren Entfernungen zwischen Lesegerät und Transponder stark. Wir bewegen uns hier in der Größenordnung von Direktkontakt bis zu mehreren Metern Entfernung.

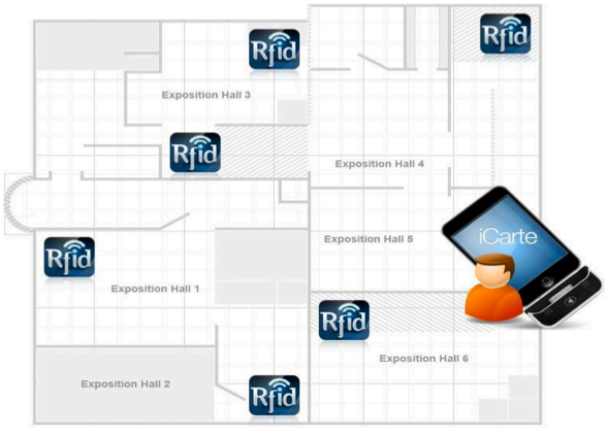


Abbildung 3.2.: Lokalisierung über RFID Tags, Quelle: [Dyn11]

Ein großes Problem, das sich beim Einsatz von RFID in unmittelbarer Nähe von Metallen ergibt, sind unkontrollierbare Gegenfrequenzen, die durch energetische Schwingungen des Trägermaterials entstehen [Sch09].

Um eine Lokalisierung möglich zu machen, müssen also Transponder verteilt werden. Ist das Lesegerät in der Lage, einen Transponder auszulesen, kann es aus einer Tabelle nachgeschlagen werden, wo es sich derzeit befindet. Dies ist auf Abbildung 3.2 zu sehen.

3.3. Bluetooth

Bluetooth ist ein Funkprotokoll, das im lizenzfreien Funkbereich von 2,4 bis 2,485 GHz eine kabellose Verbindung zwischen Geräten ermöglicht. Üblicherweise liegt die Reichweite von Bluetooth bei mobilen Geräten etwa bei zehn Metern [Blu].

Um eine Lokalisierung via Bluetooth zu ermöglichen, müssen an allen relevanten Orten an dem Auto, ähnlich wie bei RFID aus Kapitel 3.2, Sensoren verteilt werden. Da jedes Gerät eine eindeutige **ID** besitzt, ist es darüber möglich den Standort zu bestimmen. Voraussetzung dazu ist, dass eine lokale Datenbank vorliegt, in welcher alle IDs einem Ort zugewiesen wurden.

3.4. QR-Code

Ein **QR Code** ist ein zweidimensionaler Strichcode. Entwickelt wurde er von der japanischen Firma Denso Wave im Jahr 1994 [Wava].

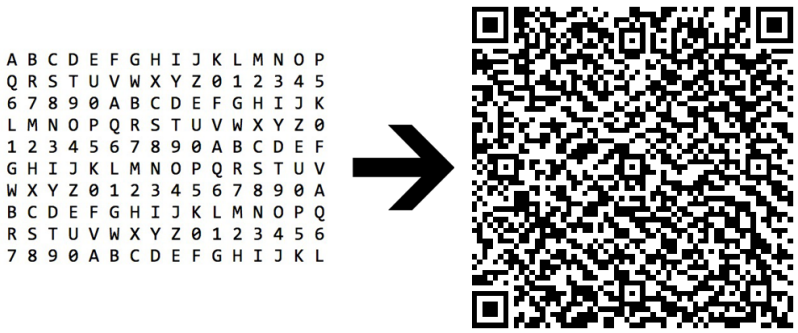


Abbildung 3.3.: Beispiel QR Code mit 327 alphanumerischen Zeichen

QR Codes haben wesentliche Vorteile, auch im Gegensatz zu einfachen Strichcodes. Zum einen ist die Informationsdichte wesentlich höher: ein Strichcode kann etwa 20 Zeichen speichern, was bei dem üblichen Anwendungsbereich im Supermarkt zur Identifizierung der einzelnen Produkte auch ausreichend ist. Ein QR Code dagegen kann bis zu 7089 Zeichen speichern, wenn es sich dabei ausschließlich um Zahlen handelt. Will man die Inhalte alphanumerisch speichern, sind es immerhin noch 4296 Zeichen [Wavb]. Abbildung 3.3 zeigt dabei ein Beispiel, in dem 327 alphanumerische Zeichen gespeichert sind.

Ein weiterer Vorteil an QR Codes ist, dass sie aus jeder Richtung, also 360 Grad, gelesen werden können. Dies wird durch Richtungsmuster ermöglicht, die sich in drei von vier Ecken befinden [Wavb].

QR Codes bieten außerdem die Möglichkeit der Fehlererkennung sowie Korrektur. Dadurch ist es möglich, bis zu 30% der Codewörter [1] wiederherzustellen. Es können daher

[1] ein Codewort entspricht einem Byte

auch teilweise verschmutzte oder beschädigte Codes gelesen werden [Wavb].

Es gibt zahlreiche Programmbibliotheken [2], die das Decodieren der Codes übernehmen. Einige davon sind **Open Source**, wie beispielsweise „ZXing“ [Ano] oder auch die „Open Source QR Code Library“ [Yan] und können daher für diese Arbeit verwendet werden, da sie keinen finanziellen Aufwand für die Entwicklung erzeugen.

3.5. Vorgegebene Wegführung/Navigation

Bei dieser Variante der Wegführung können wir auf alle oben genannten technischen Möglichkeiten verzichten und führen die gesamte Navigation innerhalb der Anwendung durch Benutzereingaben durch.

Die Idee dabei ist, dem Nutzer zu jeder Zeit die Möglichkeit zu geben, selbst zu entscheiden wohin er als nächstes gehen will und dies der Anwendung durch eine Eingabe mitteilt. Dies wäre beispielsweise durch Buttons mit *links* und *rechts* möglich. Der Vorteil dieser Lösung ist, dass die technische Realisierung einfach ist. Die Grundfunktionalität wird bereits durch das genutzte Grafikframework bereitgestellt und muss vom Entwickler nur freigeschalten werden. Für den Benutzer erfordert diese Methode allerdings einen erhöhten Aufwand im Vergleich zu den anderen Lösungen, darunter leidet vor allem der Komfort der Anwendungsnutzung.

3.6. Bewertung

Für die zu entwickelnde Anwendung ist die Auswahl auf alle bereits vorgestellten Möglichkeiten zur Lokalisierung gefallen. Ausgeschlossen davon sind lediglich CoreLocation sowie Bluetooth. Die Gründe dafür sind, dass CoreLocation innerhalb von Gebäuden zu ungenau ist beziehungsweise zu wenig Freiraum für den Entwickler bietet, um beispielsweise eigene WLAN Stationen zur Positionsbestimmung einzubinden. Die Verwendung von Bluetooth wird ausgeschlossen, da der Aufwand und die aufzubringenden Kosten sehr groß sind. Es müssten zum einen zahlreiche Bluetoothstationen aufgestellt werden, um eine möglichst genaue Positionsbestimmung sicherstellen zu können. Zum anderen ist es notwendig, alle Stationen bei jedem Review mit ihrer jeweiligen Position in eine Datenbank einzutragen und diese auf jedes Gerät zu kopieren, das für das Review verwendet wird.

Die Tags für RFID sowie QR Codes dagegen können festgelegt werden und in einer Konfigurationsdatei innerhalb der Anwendung vorliegen. Denkbar ist hierfür beispielsweise, die unterschiedlichen Autoteile mittels eindeutiger Identifikationsnummer durchzunummerieren oder den Namen des Bauteils direkt in die Tags mit einzubinden.

Durch Nutzung des Strategy Entwurfsmusters kann zur Laufzeit die bestmögliche Lokalisierungsmethode ermittelt und genutzt werden. Organisatoren haben ebenfalls die Möglichkeit, innerhalb der Anwendung festzulegen, welche Methode genutzt werden soll.

[2] eine Sammlung von Funktionen für ähnliche beziehungsweise zusammengehörige Aufgaben

Teil IV.

System und Object Design

1. Systementwurf

Dieses Kapitel beschäftigt sich mit dem Systementwurf. Dieser geht aus dem Analysemodell hervor. Nach Brügge und Dutoit [BD04] verstehen wir darunter die Transformation des Analysemodells in das Systementwurfsmodell. Hier werden die Entwurfsziele spezifiziert und die einzelnen, möglichst lose gekoppelte, Subsysteme identifiziert.

Im ersten Abschnitt werden die Entwurfsziele definiert. In Abschnitt 1.2 findet die eigentliche Systemzerlegung statt. Die einzelnen Subsysteme werden hier identifiziert. Die jeweilige Beschreibung der Subsysteme findet in Abschnitt 1.3 statt.

1.1. Entwurfsziele

Entwurfsziele „identifizieren die Beschaffenheit des [...] zu optimierenden Systems“ [BD04]. Diese leiten wir aus den in den Kapiteln II.3 sowie II.4 identifizierten Anforderungen ab. Genauer ausgedrückt werden die nichtfunktionalen Anforderungen, die sich gegenseitig einschränken, priorisiert.
Die absteigende Priorität unserer Ziele ist im Folgenden aufgeführt. Sie ergibt sich zunächst aus dem Komfort in der Nutzung der Anwendung und er Fähigkeit, sich an jedes Review anzupassen. In wie fern die Anwendung nach Abschluss der Bachelorarbeit weiterentwickelt wird, ist derzeit nicht gewiss und spielt auch nur eine untergeordnete Rolle.

Nutzbarkeit

Ein elementarer Faktor beim Entwurf dieser Software ist seine Nutzbarkeit sowie Auswertbarkeit. Dies ist einer der Hauptvorteile gegenüber dem bisherigen papierbasierten System. Daher wird es vermieden, das System unnötig überladen oder kompliziert zu konzipieren, um die bereitgestellten Funktionalitäten effizient und effektiv verwendbar zu machen. Ansonsten besteht die Gefahr, dass Nutzer das System meiden.

Anhand der nichtfunktionalen Anforderung Benutzbarkeit aus Abschnitt II.4.1 muss sichergestellt sein, dass alle Funktionen innerhalb maximal drei Klicks erreichbar sind. Alle Funktionen sind leicht zu finden und intuitiv zu bedienen. Tests für die Nutzbarkeit sind nur schwer evaluierbar und wurden deshalb für diese Anwendung nicht durchgeführt.

Anpassungsfähigkeit

Das System muss für neue Datensätze leicht anpassbar sein und diese möglichst eigenständig einbinden und verwalten. Diese Daten können importierte, dreidimensionale Modelle eines Autos sein oder die dazugehörende Property List, in der bereits Multiple Choice Fragen und die dazu gehörigen Antworten vordefiniert sind. Des Weiteren werden auch die aktuellen Benutzer (Reviewer sowie Organisatoren), in einer Property List gespeichert, in die Anwendung geladen.

Das System soll in der Lage sein, diese Dateien zu importieren und anschließend eigenständig zu erkennen, um welche Art es sich handelt und sie dementsprechend einzubinden und zu verarbeiten.

Dies soll durch eine zentrale Verwaltungskomponente realisiert werden, die unabhängig von allen anderen Komponenten arbeitet.

Erweiterbarkeit

Dieses System entsteht im Kontext einer Bachelorarbeit und soll nach Abschluss dieser Arbeit gegebenenfalls weiterentwickelt werden um weitere Funktionalitäten bereitstellen zu können. Mögliche Optionen hierfür werden im Weiteren in Kapitel V aufgeführt.

Um Erweiterungen mit möglichst geringem Aufwand zu ermöglichen, wird dies im Entwurf der einzelnen Subsystemen und deren Kopplung aneinander berücksichtigt. Speziell wird die Datenhaltung, die Darstellung der Daten sowie die Steuernden Elemente streng voneinander abgegrenzt. Erweiterungen können durch Kopplung an den `MainController` in die Anwendung eingebunden werden.

1.2. Systemzerlegung

In diesem Abschnitt behandeln wir die Zerlegung des Systems in seine Subsysteme. Hierbei handelt es sich um vereinfachte Modelle der einzelnen Pakete des Systems. Um das grundlegende Verständnis der relevanten Klassen hervorzuheben, werden explizit nicht alle Klassen der aufgeführten Subsystemen aufgeführt.

1.2.1. Eingesetzte Softwarearchitektur

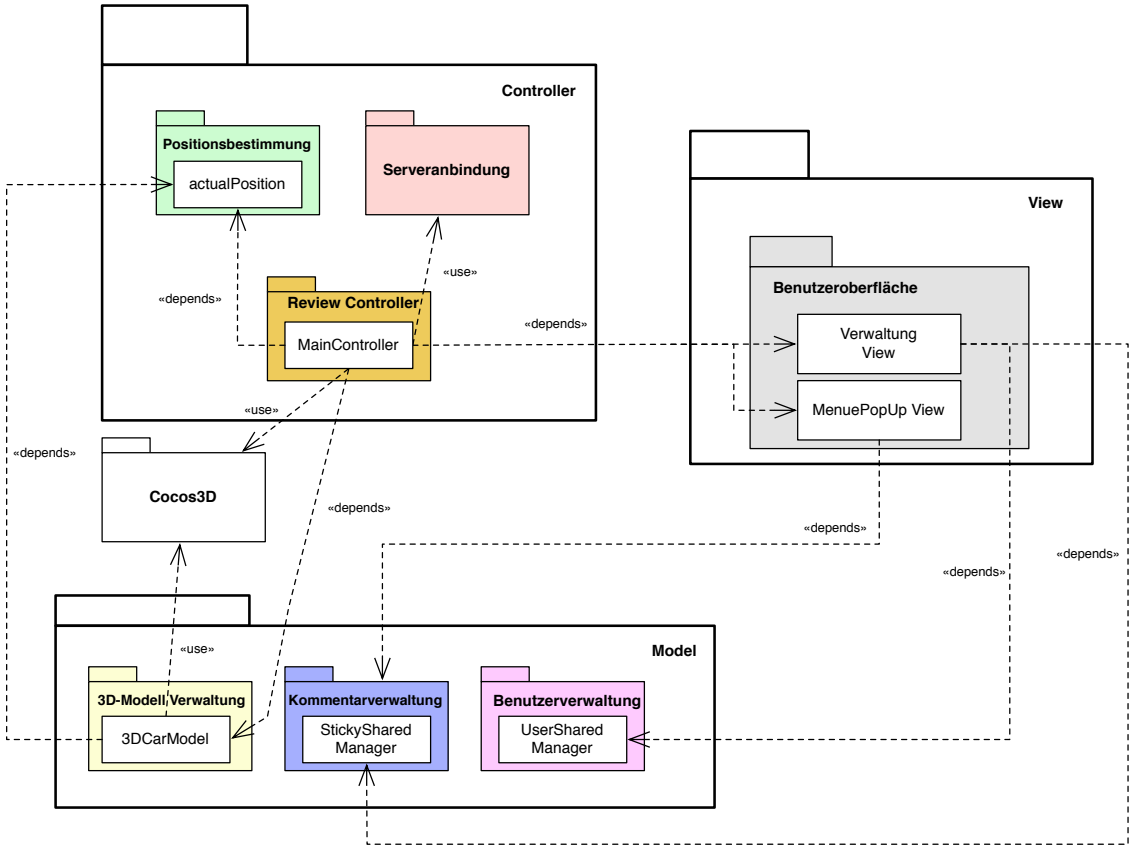


Abbildung 1.1.: MVC Softwarearchitektur der Anwendung

Im Analysismodell aus Abschnitt II.6.1 wurden sieben Komponenten identifiziert. Diese bilden die Subsysteme aus Abbildung 1.1 dieser Software. Es handelt sich hier um eine Model-View-Controller [1] Architektur. Diese Architektur ermöglichst die Kapselung aller Subsysteme in drei Klassen, die in Abbildung 1.1 zu sehen sind. Die farbliche Kennzeichnung dient der Wiedererkennung anhand des Klassendiagramms aus Abschnitt II.6.1. Dieser flexible Programmentwurf erlaubt spätere Änderungen, Erweiterungen der Anwendung sowie die Wiederverwendung der einzelnen Komponenten, da sie durch das MVC-Paradigma entkoppelt werden [GHJV04]. Beispielsweise kann so das Datenmodell sowie die Steuerung für eine Portierung auf andere Plattformen herangezogen werden.

Das Modell enthält die für die Anwendung darzustellenden Daten, in unserem Fall die Benutzer- und Reviewdaten. Es ist unabhängig von der Präsentation (View) und Steuerung (Controller). Die Präsentation gibt den aktuellen Zustand des Modells wieder und ermöglicht es dem Benutzer, die Daten zu ändern. Sie kennt das zu ihr gehörige Modell sowie ihre Steuerung. Die vom Benutzer eingegebenen Daten werden jedoch nicht von

[1] im Weiteren abgekürzt **MVC**

der Präsentation, sondern von dem dazu gehörigen Modell verarbeitet. Die Steuerung verwaltet in unserem Fall zwei Präsentationen: Die Verwaltung sowie das Menü PopUp. Wie und wann welche Präsentation angezeigt wird, ermittelt die Steuerung anhand der Benutzereingaben. So ist es für die Steuerung möglich, gewisse Einschränkungen für den Benutzer vorzunehmen. In unserer Anwendung findet dies, je nach Benutzerberechtigung, in der Verwaltung statt. Reviewer haben hier nur eingeschränkte Optionen zur Auswahl, während Organisatoren den vollen Zugriff auf alle Funktionen haben, beispielsweise den Import von 3D-Modellen sowie die Bearbeitung der Benutzer.

1.3. Subsysteme

Im Folgenden werden die einzelnen Subsysteme im Detail beschrieben. Die farbliche Markierung wurde, wie im vorherigen Abschnitt, zur Wiedererkennung der groben Unterteilung des Klassendiagrammes aus Abschnitt II.6.1 verwendet.

1.3.1. Modell

3D-Modellverwaltung

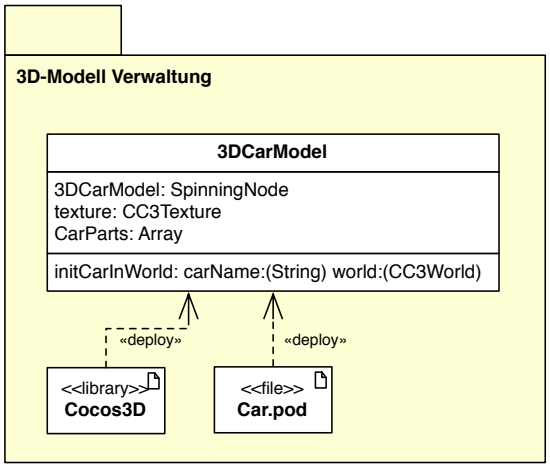


Abbildung 1.2.: Detailansicht des Subsystems 3D-Modellverwaltung

Die Klasse `3DCarModel` aus dem Subsystem 3D-Modellverwaltung ist für die Repräsentation der dreidimensionalen Automodelle zuständig. Die Referenz auf die Power VR POD wird hier gesetzt. Die unterschiedlichen Bauteile eines Autos sind bereits in der POD vorliegend und werden in einem Array gesichert, wie in Abbildung 1.2 zu sehen ist.

Initialisiert wird ein Automodell, indem man den Namen zu der POD Datei sowie die eine Instanz der Welt aus Cocos3D übergibt, in der das Auto dargestellt werden soll.

Kommentarverwaltung

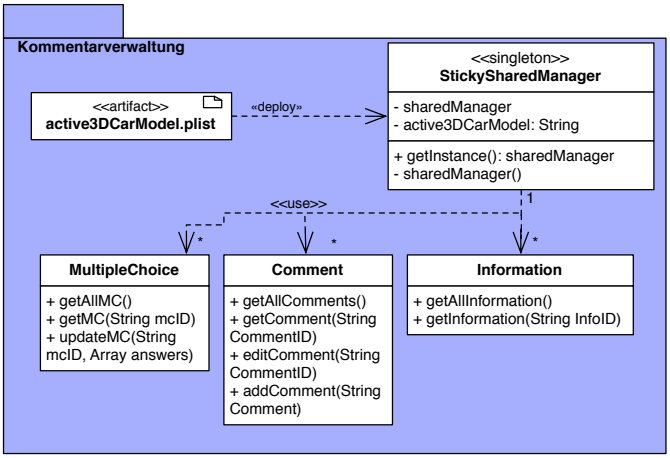


Abbildung 1.3.: Detailansicht des Subsystems Kommentarverwaltung

Die Kommentarverwaltung aus Abbildung 1.3 besitzt die Klasse `StickySharedManager`, wobei es sich hier um ein Singletonobjekt handelt. Die Verwendung eines Singleton ist hier sinnvoll, dem Benutzer gezeigten Präsentationen, auf diese Daten zugreifen. Es wird hierbei sichergestellt, dass keine redundante Datenhaltung stattfindet, da alle Klassen auf die selbe Instanz zugreifen. [GHJV04]

Dieses Singleton organisiert alle Daten, die für ein Review relevant sind. Im Detail sind das Multiple Choice Fragen inklusive der jeweiligen Antworten, Freitextkommentare sowie weiterführende Informationen zu dem jeweiligen Autobauteil.

Für Informationen werden Methoden gegeben, um diese auszugeben. Multiple Choice Fragen können nur aktualisiert werden, was im Detail bedeutet, dass gegebene Antworten gespeichert werden. Für Kommentare werden Methoden geliefert, um diese auszugeben, anzulegen sowie zu bearbeiten. Hierbei ist wichtig, dass nur Kommentare aus einer laufenden Sitzung bearbeitet werden können. Dies festzustellen obliegt der Verantwortung des StickySharedManagers.

Benutzerverwaltung

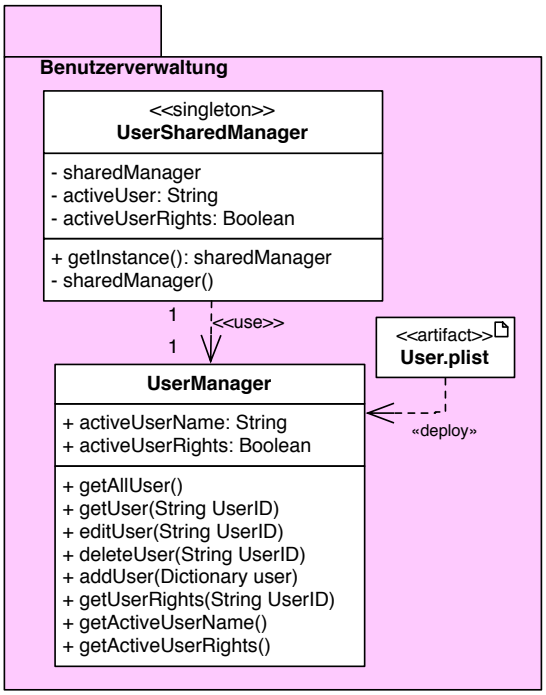


Abbildung 1.4.: Detailansicht des Subsystems Benutzerverwaltung

Bei dem Subsystem Benutzerverwaltung handelt es sich, wie bei der Kommentarverwaltung, um ein Singletonobjekt. Dies ist hier sinnvoll, da von einigen Klassen aus dem Subsystem `Benutzeroberfläche` geprüft werden muss, über welche Benutzerrechte der aktuelle Nutzer verfügt. Um sicherzustellen, dass nicht beliebig viele Instanzen des `UserSharedManagers` angelegt werden, die parallel lesend und schreibend auf die Daten zugreifen, muss bei der Verwendung von Singletons nicht geprüft werden, ob gerade gelesen oder geschrieben wird.

Die Klasse UserSharedManager organisiert dafür den Zugriff auf die Klasse `UserManager`. Sie hat Zugriff auf die User.plist, in der alle Benutzer inklusive ihrer Berechtigungen gespeichert sind. Der UserManager bietet Methoden zum Anzeigen, Bearbeiten, Hinzufügen und Löschen von Benutzern. Des Weiteren sichert sie den Benutzernamen sowie die Berechtigung des Anwenders der aktuellen Sitzung. Abbildung 1.4 stellt diese Zusammenhänge dar.

1.3.2. Präsentation

Benutzeroberfläche

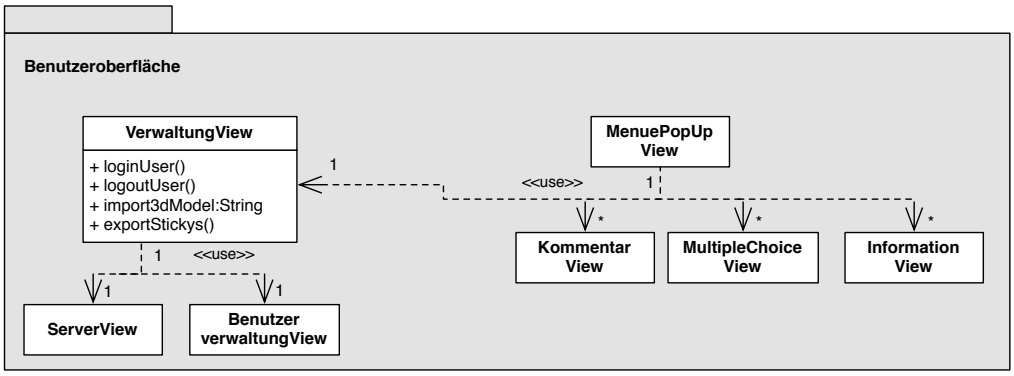


Abbildung 1.5.: Detailansicht des Subsystems Benutzeroberfläche

Das Subsystem `Benutzeroberfläche` aus der Klasse Präsentation fasst alle unterschiedlichen Bildschirme zusammen. Über einen Button auf dem Monitor lässt sich jederzeit eine Instanz der Klasse `MenuePopUp` aufrufen. Hier stehen vier Optionen, Kommentar, Multiple Choice, Information sowie Verwaltung, zur Wahl, die jeweils eine neue Maske öffnen. Vom `VerwaltungsView` aus lässt sich die Benutzerverwaltung sowie die Serverkonfiguration öffnen.

Diese Subsystem ist in Abbildung 1.5 zu sehen.

1.3.3. Steuerung

ReviewController

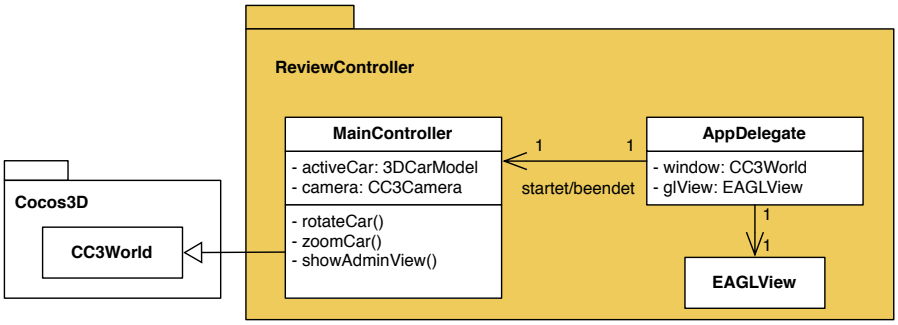


Abbildung 1.6.: Detailansicht des Subsystems ReviewController

Mit dem Start der Anwendung wird eine Instanz des `AppDelegate` erzeugt. Das AppDelegate erzeugt eine Instanz eines EaglViews, auf den der `MainController`, der die Superklasse `CC3World` hat, gesetzt wird.

Der MainController implementiert Methoden seiner Superklasse, wodurch er zu der zentraler Instanz für alle grafischen Elemente der Anwendung wird. So wird neben einer Kamera, dem Blickfeld für den Benutzer, und einer Lampe auch das 3D-Automodell platziert. Interaktionen mit diesen Elementen wie Rotation und Zoom werden hier aufgerufen und durchgeführt, siehe Abbildung 1.6.

Positionsbestimmung

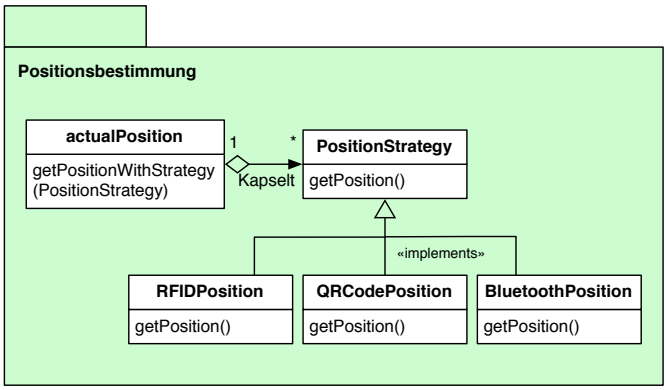


Abbildung 1.7.: Detailansicht des Subsystems Positionsbestimmung

In Abbildung 1.7 ist das Subsystem „Positionsbestimmung“ dargestellt. Die Klasse `actualPosition` orientiert sich an dem `Strategie` Entwurfsmuster. Ziel ist es, eine Familie von Algorithmen zu definieren, die ein Problem auf unterschiedliche Art und Weise lösen. Mit dem Strategiemuster ist es möglich, „den Algorithmus unabhängig von ihn nutzenden Klienten zu variieren“ [GHJV04].

In unserer Anwendung folgt `actualPosition` diesem Prinzip. So kann je nach Umgebung zur Laufzeit der Anwendung festgelegt werden, mit welcher Strategie die aktuelle Position bestimmt werden soll. Für eine exaktere Positionsbestimmung sind auch mehrere Strategien parallel anwendbar.

Serveranbindung

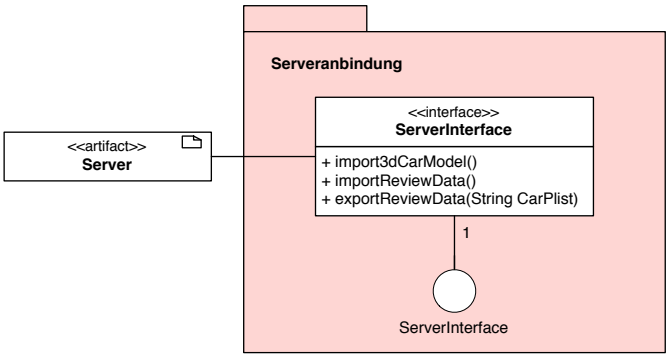


Abbildung 1.8.: Detailansicht des Subsystems Serveranbindung

Das `ServerInterface` aus Abbildung 1.8 stellt ein Interface zu einem entfernten Server dar. Es bietet drei Methoden, für eine Interaktion mit diesem Server.

Zum einen lassen sich mit „exportReviewData“ die während des Review erhobenen Daten an den Server exportieren. Zum anderen sind über `import3dCarModel` und `importReviewData` die Grafikdaten für ein 3D-Modell mit den dazu passende XML-Fragebögen importierbar.

2. Identifizierung und Speicherung von persistenten Daten

Nach Brügge und Dutoit [BD04] bestehen persistente Daten [1] über die einmalige Ausführung eines Systems hinweg. In diesem Abschnitt wird geklärt, welche Daten wie gespeichert werden. Dies wird erreicht, indem „alle Klassen daraufhin untersucht [werden], ob der von ihnen verwaltete Zustand die Beendigung des Systems überleben muss“ [BD04].

Im vorherigen Abschnitt wurden die Subsysteme dieser Anwendung identifiziert. Die Objekte, die persistent gespeichert werden, werden mit den Subklassen `KommentarVerwaltung` sowie `BenutzerVerwaltung` beschrieben.

Laut Brügge und Dutoit [BD04] gibt es aktuell drei Möglichkeiten für die Speicherverwaltung:

- **Dateien**
  Die Verwendung von Dateien für die Speicherverwaltung bietet sich an, wenn entweder umfangreiche Daten (z.B. Bilder), temporäre Daten (z.B. Kerndateien) oder geringe Informationsdichten gesichert werden sollen.

- **Relationale Datenbanken**
  Relationale Datenbanken sollten verwendet werden, wenn komplexe Abfragen gestellt werden oder es sich um große Datensätze handelt. Auch bieten sie sich an, wenn mit zeitgleichen Zugriffen gerechnet werden muss oder heterogene Plattformen im Einsatz sind.

- **Objektorientierte Datenbanken**
  In objektorienterten Datenbanken können Klassen und Assoziationen direkt gespeichert werden. Sie bieten sich daher für mittelgroße Datensätze an, wenn mit irregulären Assoziationen zwischen Objekten gerechnet wird und diese Assoziationen häufig genutzt werden. Ähnlich wie bei relationalen Datenbanken sind objektorientierte Datenbanken einzusetzen, wenn mit zeitgleichen Zugriffen gerechnet werden muss oder heterogene Plattformen im Einsatz sind.

In dieser Anwendung werden, nach dem aktuellen Systementwurf, nur einfache Datentypen wie Strings und Integer gespeichert. Durch diese geringe Informationsdichte ist die Auswahl auf die Persistierung mit Dateien gefallen. In einem Testdurchlauf der Anwendung sind beispielsweise nur 7kB Daten angefallen, nachdem für jedes Bauteil Informationen angelegt wurden, jeweils ein Kommentar geschrieben wurde sowie Multiple Choice Fragen beantwortet worden sind. In Kapitel III wurden Möglichkeiten evaluiert, welche Möglichkeiten für eine Persistierung mit Dateien in Betracht gezogen werden. Es werden nach dieser Evaluation Property Lists als Speichermedium verwendet.

[1] auch dauerhafte Daten genannt

Key	Type	Value
▼comments	Diction...	(7 items)
▼1	Diction...	(5 items)
comment	String	kein Kommentar vorhanden
author	String	Schnell Adrian
car	String	Lexus
part	String	Reifen
date	String	20.10.2011 15:34

Abbildung 2.1.: Beispieldatensatz für einen Kommentar

In Abbildung 2.1 ist der Datensatz eines Kommentars für den Reifen eines Lexus zu sehen. Hier werden die verschiedenen Attribute als Strings abgespeichert und in einem Dictionary zusammengefasst.

Key	Type	Value
▼MultipleChoice	Diction...	(8 items)
▼1	Diction...	(5 items)
▼answered	Array	(0 items)
question	String	Was sollte an dem Fahrgestell verändert werden?
author	String	
▼answers	Array	(6 items)
Item 0	String	Form des Kotflügels
Item 1	String	Fahrer-/Beifahrertüren
Item 2	String	Heck
Item 3	String	allgemein die Form
Item 4	String	sportlicheres Design
Item 5	String	Tieferlegen
part	String	Fahrgestell

Abbildung 2.2.: Beispieldatensatz für eine Multiple Choice Frage inklusive Antwortmöglichkeiten

Multiple Choice Daten werden, wie in Abbildung 2.2 zu sehen, ähnlich wie Kommentare gespeichert. Allerdings werden Antwortmöglichkeiten sowie die dazugehörigen Antworten in Arrays gespeichert.

Kommentare, Multiple Choice sowie Informationen werden zusammen in einer Property List verwaltet. Der Vorteil davon für den Entwickler ist zu sehen, wenn ein Export der Reviewdaten durchgeführt wird - es muss nur eine einzige Datei übertragen werden. Die Wahrscheinlichkeit von auftretenden Übertragungsfehlern wird so auf eine Datei reduziert.

Key	Type	Value
▼Adrian Schnell	Diction...	(3 items)
benutzer	String	adrian
passwort	String	adrian
organisatorRights	Boolean	YES
▼Max Mustermann	Diction...	(3 items)
benutzer	String	max
passwort	String	max
organisatorRights	Boolean	NO

Abbildung 2.3.: Beispieldatensatz für zwei Benutzer der Anwendung

In einer separaten Property List werden die Benutzer verwaltet, die sich authentifizieren dürfen. Dadurch ist eine Entkopplung von Review- und Benutzerdaten hergestellt. Wie in Abbildung 2.3 abgebildet, werden hier Strings für den Benutzernamen sowie das Passwort zusammen mit einem Boolean für die Berechtigung in einem Dictionary gebündelt. Für die vorliegende Iteration des Prototypen wurde auf die Verwendung verschlüsselter Passwörter aus Zeitgründen verzichtet, auch wenn dies ein Sicherheitsrisiko darstellt. Ein Boolean kann hier verwendet werden, da das System derzeit nur zwei unterschiedliche Benutzergruppen vorsieht. Sollte sich diese Anforderung im Laufe der Zeit ändern, könnte an dieser Stelle auf jeden beliebigen Datentyp zurück gegriffen werden. Durch die verwendete Softwarearchitektur müsste nur eine Methode in der Benutzerverwaltung geändert werden, um den weiteren Betrieb der Software weiterhin sicherzustellen.

Durch die Besonderheit von iOS, dass neu importierte Daten von dem Hauptordner der Software in einen von der Software beschreibbaren Bereich kopiert werden. Dadurch voll die Anwendung durch unvorsichtige Vorgehensweise der Entwickler vor Korruption anderer Daten geschützt werden. Die Anwendung ist jederzeit in den Ursprungszustand wiederherstellbar.

3. Einrichten von Zugriffskontrolle

In einem Mehrbenutzersystem haben die Akteure unterschiedliche Berechtigungen auf Funktionalitäten sowie Daten [BD04]. In der Analyse aus Kapitel II wurde dies bereits anhand von Anwendungsfällen modelliert.

VerwaltungsView (hat Zugriff)	**Akteur: Organisator**	**Akteur: Reviewer**
Auto wechseln	ja	ja
Exportiere Reviewergebnisse	ja	ja
Importiere Automodell/-daten	ja	ja
eigenes Benutzerkonto bearbeiten	ja	ja
Benutzer verwalten	ja	
Aktualisiere Benutzerdaten (Server)	ja	
Impressum	ja	ja
Ausloggen	ja	ja

Tabelle 3.1.: Zugriffsmatrix für die Hauptverwaltung

Bei Anwendungsbeginn wird dem Benutzer die Verwaltungsmaske angezeigt. Hier wird er dazu aufgefordert, sich mit seinen Benutzerdaten zu authentifizieren. Nach einer gültigen Eingabe von Benutzername und dazugehörigem Passwort wird die Berechtigung überprüft. Davon ist abhängig, welche Optionen ihm im Menü angezeigt werden. Der Unterschied der Berechtigung wird in Tabelle 3.1 verdeutlicht. Reviewer haben die Aufgabe, Befragungen durchzuführen. Die Verwaltung der Benutzerdaten gehört nicht zu ihren Aufgabengebieten und ist daher untersagt.

Teil V.

Zusammenfassung

Bei der Neuentwicklung von Autos findet vor der Produktion ein etwa dreijähriger Entwicklungsprozess statt. In dieser Zeit wird neben der technischen Spezifikation des Autos auch das Design festgelegt. Für den Hersteller des Autos ist es wichtig, vor Beginn der Produktion den Verkaufserfolg zu evaluieren und inwiefern es den Zeitgeschmack der Zielgruppe trifft. Aus diesem Grund werden Marktforschungen, sogenannte Car Reviews, durchgeführt. Derzeit werden diese mit Papierfragebögen erfasst, was zu einigen Problemen führt. Besonders bei der Planung und der Auswertung entsteht ein großer Arbeits- und Zeitaufwand. Zum einen müssen Fragebögen für jedes Auto erstellt und gedruckt werden. Zum Anderen müssen die Fragebögen nach Abschluss des Reviews ausgewertet werden, was derzeit manuell geschieht. Weitere Probleme ergeben sich bei der Auswertung durch eventuell unlesbare Handschriften oder fehlerhafte Dokumentation.

Diese Bachelorarbeit hat sich mit diesen Car Reviews befasst. Ziel war es, derzeitige Probleme aufzugreifen und diese zu minimieren beziehungsweise zu beseitigen. Des Weiteren wurde angestrebt, weitere Arbeitserleichterungen einzuführen. Die Fragebögen für das Review werden nun einmalig von einem Organisator, beispielsweise durch Nutzung einer kleinen Anwendung, in XML erstellt und zentral auf einem Server abgelegt. Die Reviewer, die die Befragungen durchführen, wählen vor Beginn aus einer Liste die Autos aus, für die sie zuständig sind. Die Anwendung importiert vom Server ein Grafikmodell des Autos sowie die dazugehörigen Fragebögen. Bereits hier findet eine Verringerung des Arbeitsaufwands der Organisatoren statt.

Während der Durchführung wird dem Reviewer ein 3D-Modell des Autos dargestellt. Mit einem Klick auf ein Teil des Autos öffnet sich eine Eingabemaske, in der ausgewählt werden kann, ob man eine Multiple Choice Frage beantworten möchte, einen Kommentar abgeben oder weitere Informationen einsehen will. Letzteres bietet einen Mehrwert gegenüber der bisherigen Reviewdurchführung. Kommentare werden automatisch dem Reviewer sowie dem Auto zugeordnet. Eine Übersicht über alle Multiple-Choice-Fragen, Kommentare sowie weiterführende Informationen ist über das Hauptmenü erreichbar.

Nach Abschluss des Reviews übertragt der Reviewer seine Umfragedaten, gesichert in einer XML Datei, mit einem Klick an den Server. Die Auswertung kann hier größtenteils automatisch durchgeführt werden. Freitextkommentare müssen allerdings, wie bisher auch, manuell ausgewertet werden. Durch die Automatisierung können Ergebnisse schneller evaluiert werden. Dies bietet die Möglichkeit, gegebenenfalls mehrere Reviews an einem Tag durchzuführen. Bisher ist nur ein Review je Tag möglich, daher bietet das neue System einen enormen Mehrwert für die Automobilhersteller.

In Weiteren Verlauf dieses Kapitels wird der implementierte Prototyp sowie seine Funktionsweise anhand der grafischen Oberfläche vorgestellt. In der derzeit vorliegenden Fassung erfüllt der implementierte Prototyp bereits den größten Teil seiner in Kapitel II identifizierten Anforderungen. Es wird überprüft, welche Anforderungen von diesem Prototyp realisiert sind.

1. Erfüllte Ziele

In diesem Abschnitt wird beschrieben, inwiefern die in Kapitel II identifizierten Anforderungen durch den Prototypen der Anwendung erfüllt sind.

1.1. Erfüllte funktionale Anforderungen

Die prototypische Implementierung erfüllt folgende funktionale Anforderungen:

1.1.1. Benutzerverwaltung

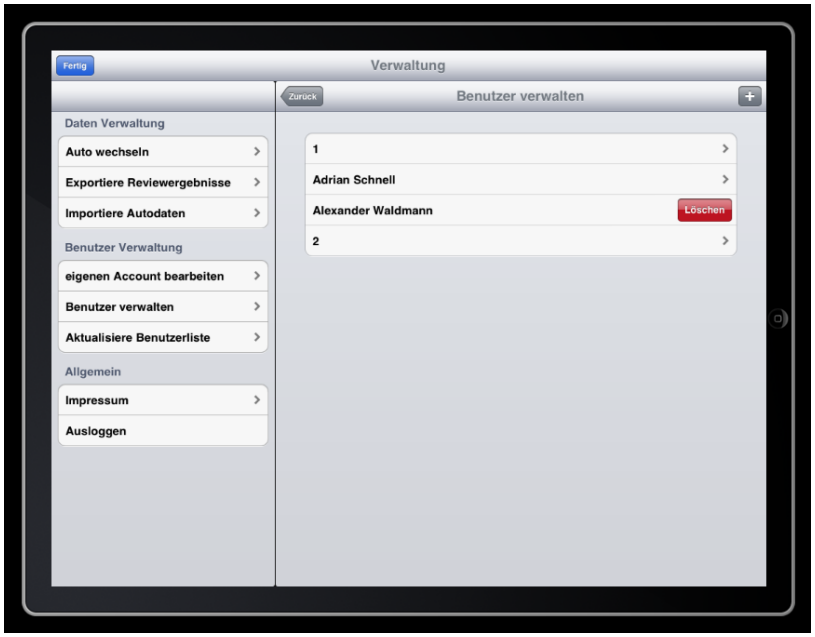


Abbildung 1.1.: Benutzerverwaltung der Anwendung

Die Verwaltungsmaske für Benutzer aus Abbildung 1.1 ist, wie in den funktionalen Anforderungen aus Abschnitt II.3, ausschließlich für Organisatoren sichtbar.

Die Funktionen „Benutzer hinzufügen“ sowie „Benutzer löschen“ sind hier direkt ausführbar. Um einen Benutzer zu entfernen, ist die Wischgeste [1] erforderlich, wie aus vielen anderen iOS Anwendungen bekannt. Um einen Benutzer zu bearbeiten, wird die entsprechende Zeile aus der Tabelle ausgewählt. Es wird die Eingabemaske geladen, wie sie in Abbildung A.3 aus dem Anhang zu sehen ist.

[1] bezeichnet die horizontale oder vertikale Bewegung eines Fingers über das Touchscreen

1.1.2. Kommentieren und Multiple Choice

Abbildung 1.2.: Übersicht aller Multiple Choice Fragen inklusive Markierung, ob diese bereits beantwortet wurde

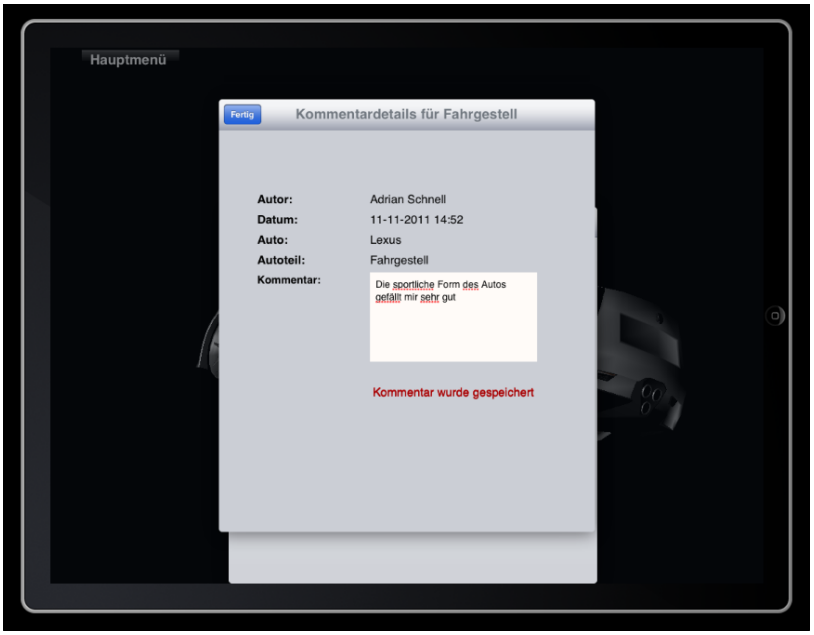


Abbildung 1.3.: Eingabe eines Freitextkommentars. Bis auf den Kommentar werden alle Felder vom System automatisch gesetzt

Die Erfüllung der Anforderung, Kommentare sowie Multiple Choice Fragen beantworten zu können, ist in den Abbildungen 1.2, A.5 sowie 1.3 zu sehen.

Der Reviewer hat die Möglichkeit zum einen einen Kommentar oder eine Multiple Choice Frage aus einer Liste auszuwählen, in der alle Bauteile zusammen aufgeführt sind. Alternativ dazu können, mit einer Berührung des entsprechenden Bauteils, Kommentare und Multiple Choice speziell hierfür ausgewählt werden.

In der Übersicht der Multiple Choice Fragen wird mit einem Icon hervorgehoben, was bereits beantwortet wurde. Die Anzahl der Antworten ist flexibel und wird in der Property List festgelegt. Für Kommentare sind bei dem ersten Start der Anwendung für jedes Bauteil bereits Standardeinträge angelegt, der Benutzer kann diese entweder bearbeiten oder einen neuen Kommentar anlegen. Aus zeitlichen Gründen wurde das Anlegen von Video- sowie Audiokommentaren nicht mehr fertiggestellt.

1.1.3. Login und Logout

Abbildung 1.4.: Loginmaske nach dem Start der Anwendung

Bei Start der Anwendung fordert das System den Benutzer auf, sich mit seinen Zugangsdaten zu authentifizieren. Ohne Eingabe der Benutzerdaten ist die Nutzung der Anwendung ausgeschlossen. Diese Aufforderung ist in Abbildung 1.4 zu sehen. Wie in den funktionalen Anforderungen gefordert, werden beim Login die Zugangsrechte gesetzt.

Die Unterschiede in der Darstellung der Verwaltung ist in den Abbildungen A.1 und A.2 aus dem Anhang zu erkennen.

1.1.4. Multimediaroutine

Abbildung 1.5.: Darstellung eines 3D-Automodells

Die für Grafikfunktionen zugrunde liegende Bibliothek Cocos3D ist stark an diese Anwendung gekoppelt. Der Benutzer hat die Möglichkeit, mit dem 3D-Modell zu interagieren im Sinne von Rotation. Dies geschieht durch einen Fingerwisch zu. Wird auf das Modell getippt, öffnet sich die bereits unter Abschnitt 1.1.2 beschriebene Eingabeaufforderung.

1.2. Erfüllte nicht funktionale Anforderungen

Im folgenden werden nicht funktionale Anforderungen aufgeführt, die durch die prototypische Implementierung erfüllt werden.

1.2.1. Benutzerfreundlichkeit

Bei der Gestaltung der Benutzeroberfläche wurde streng auf die von Apple gegebenen „Human Interface Guidelines“ [App11a] geachtet, um ein gewisses „Look and Feel“ zu bieten. Die von diesen Richtlinien geregelten Gestaltungsmöglichkeiten bieten den Vorteil, dass sich neue Benutzer schnell in der Anwendung zurecht finden. So wird beispielsweise geregelt, dass der Button „Hinzufügen“ immer gleich aussieht.

Wie gefordert, ist jede von der Software gebotene Funktionalität innerhalb maximal drei Klicks erreichbar. Eine weitere Verschachtlung der Funktionen in den Menüs hätte das Ergebnis, dass die Anwendung unübersichtlich wird und Optionen vom Benutzer übersehen werden.

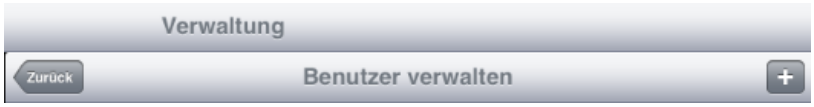


Abbildung 1.6.: Erweiterte Übersicht innerhalb des Menüs durch Verwendung von zwei Navigationsleisten

Um die Übersicht zu erhöhen, werden innerhalb der Hauptverwaltung zwei Navigationsleisten verwendet. Dem Benutzer wird so jederzeit eindeutig mitgeteilt, wo er sich innerhalb der Verwaltung befindet, siehe Abbildung 1.6.

1.2.2. Zuverlässigkeit

Die Zuverlässigkeit, die auch oft als Verlässlichkeit bezeichnet wird, umfasst nach Brügge und Dutoit [BD04, S. 151] ebenfalls Robustheit und die Sicherheit.

Diese Anwendung erfüllt alle Kriterien hierfür, die in den Anforderungen festgelegt wurden. Da keine Unit-Tests [2] angelegt wurden, sind einige dieser Anforderungen schwer zu evaluieren. Daher kann hier nur das Verhalten in beispielhafter Durchführung eines Reviews mit der Anwendung beurteilt werden.

Es konnte bisher kein Absturz und damit eventuell auftretender Datenverlust festgestellt werden. Sollte in längerer Verwendung der Software ein gravierender Fehler auftreten, der zu einem Absturz führen kann, ist die Gefahr eines Datenverlustes ausgeschlossen. Grund dafür ist, dass alle Eingaben sofort persistent gesichert werden.

Im vorherigen Abschnitt „Benutzerfreundlichkeit“ wurde die Benutzeroberfläche bereits beschrieben. Im Sinne von Robustheit hilft sie, fehlerhafte Eingaben des Benutzers zu vermeiden. Bei Beantwortung der Multiple Choice Fragen werden Auswahlmöglichkeiten vorgegeben. In der Eingabe von Kommentaren sind Freitexteingaben möglich. Hier eingegebene Zeichenketten werden darauf überprüft, ob Sonderzeichen enthalten sind. Falls solche gefunden werden, entfernt die zuständige Persistierungskomponente diese vor der Speicherung um Code Injections vorzubeugen.

Der Import von 3D-Modelldaten sowie der Export der erhobenen Review-Daten kann wegen Mangels eines Servers nicht überprüft werden. Es kann jedoch überprüft werden, wie das System reagiert, wenn innerhalb der Testumgebung „iOS-Simulator“ neue Daten eingebunden werden. Dabei konnten keine Fehler festgestellt werden. Der Wechsel zwischen zu bewertenden Autos innerhalb der Anwendung ist realisiert und benötigt im Durchschnitt drei Sekunden und damit weniger als die geforderten fünf Sekunden.

Der Aspekt der Sicherheit wurde bereits in Abschnitt 1.1.3 beschrieben. Es ist daher sichergestellt, dass nur berechtigte Benutzer Zugang zu den Daten der Anwendung haben.

1.2.3. Unterstützung und Wartung

Änderungen innerhalb des Systems lassen sich leicht vornehmen. Im Analysemodell und dem Systementwurf wurde auf das Entwurfsziel der Entkopplung von Komponenten geachtet [GHJV04]. Auf eine ausführliche Dokumentation, im Sinne von Kommentaren, wur-

[2] ein Testverfahren der einzelnen Komponenten, um frühzeitig Fehler feststellen zu können

de hohen Wert gelegt. Dies ist bereits an der Anzahl der Kommentarzeilen (siehe Anhang B) zu sehen.

Trotz fehlender Serveranbindung unterstützt die Anwendung den Import von 3D-Modellen sowie Reviewdaten, die die Multiple Choice Fragen und Anworten enthalten. Die Verbindung zwischen ReviewDaten sowie 3D-Modell wird selbstständig erkannt und hergestellt.

1.2.4. Beschränkungen

Alle definierten Beschränkungen wurden eingehalten. Die Anwendung ist in Objective-C entwickelt worden und auf der Plattform iOS lauffähig. Für Multimediaroutinen wurde das Grafikframework Cocos3D angebunden.

Eine Schnittstelle für die Serveranbindung wurde angelegt. Die Implementierung konnte jedoch nicht abgeschlossen werden, keine dies zeitlich nicht mehr gereicht hat.

2. Nicht erreichte Ziele

Bei dieser Anwendung handelt es sich um einen Prototypen. Um alle angestrebten Ziele zu erreichen, müssen weitere Entwicklungsiterationen durchgeführt werden. In der aktuell vorliegenden Fassung ist es nicht möglich, einen Import von 3D-Autodaten sowie einen Export der Reviewergebnissen durchzuführen. Diese Funktionen wurden aus Zeitmangel nicht implementiert. Das Grundgerüst für die Schnittstelle zu einem Server liegt bereits vor. Die konkreten Methoden müssen noch implementiert werden.

Bereits zu einem frühen Entwicklungsstadium wurde beschlossen, dass die Lokalisierung der Anwendung an einem realen Autoprototypen nicht implementiert wird. Eine Recherche, welche Möglichkeiten in Erwägung gezogen werden, wurde allerdings durchgeführt. Im Systemdesign ist die Lokalisierung berücksichtigt worden und kann durch eine definierte Schnittstelle am `MainController` erweitert werden. Dieser ist unter anderem für die Darstellung und Interaktion mit dem 3D-Automodell verantwortlich.

Des Weiteren ist es nicht möglich, multimediale Kommentare in Form von Video- sowie Audioaufnahmen durchzuführen. Die Anwendung unterstützt nur die grundlegenden Funktionen wie Freitextkommentare sowie die Beantwortung von Multiple Choice Fragen.

Diese drei nicht erreichten Ziele wurden in der Analyse und Planung der Anwendung berücksichtigt, wurden jedoch mit einer geringeren Priorität angesetzt. Für den Prototypen war es wichtig, die Grundfunktionalität sicherzustellen und demonstrieren zu können. In folgenden Iterationen der Weiterentwicklung ist es daher notwendig, die nicht erreichten Ziele zu realisieren..

3. Zukünftige Arbeit

In den vorherigen Abschnitten 1 sowie 2 wurden die erreichten sowie nicht erreichten Ziele aufgeführt. In folgenden Weiterentwicklungsschritten sollten die nicht erreichten Ziele die oberste Priorität haben.

Der Import und Export von Daten mit einem Server ist die derzeit wichtigste Funktionalität, die für eine sinnvolle Anwendung der Software fehlt. Daher sollte diese Funktionalität als erstes hinzugefügt werden. Neben der Einrichtung eines Servers muss dazu die bereits angelegte Klasse `ServerConnector` implementiert werden. Diese ist soweit in die so lose gekoppelt, dass keine anderen Subsysteme von Änderungen betroffen sind.

Die Lokalisierung des Benutzers und damit der Anwendung anhand der Position an einem realen Prototypen eines Autos wurde aufgrund der Komplexität und des Zeitaufwands vernachlässigt, da sie den Zeitaufwand der Bachelorarbeit übersteigen würden. Falls bereits der Prototyp eines Autos existiert, ist die Lokalisierung zu implementieren, um diesen zu reviewen. Durch bereits durchgeführte Analyse und Recherche wird diese Arbeit erleichtert.

Die Möglichkeit multimediale Kommentare wie Video- und Audioaufnahmen hinzuzufügen, bietet eine Funktionalität, die bei der bisherigen papierbasierten Durchführung eines Reviews nicht möglich ist. Um die technischen Möglichkeiten, den Spaßfaktor bei einem Review zu erhöhen sowie die erleichterte Feedback Umsetzung, sollte sie ebenfalls mit hoher Priorität eingeführt werden.

In weiteren Schritten der Weiterentwicklung wäre es denkbar, die Informationsdichte der Anwendung zu erhöhen. Es könnte dabei zum Start einer Befragung zunächst ein Video gezeigt werden, in dem der Prototyp des Autos den zu befragten Personen vorgestellt wird. Im gleichen Zuge könnte die Funktion hinzugefügt werden, Detailvideos für jedes Bauteil, zusätzlich zu den bereits vorhandenen Textinformationen, abzuspielen.

Die Funktion „Siri“ [1] könnte den Komfort der Kommentareingabe wesentlich erhöhen. Damit können diktierte Kommentare direkt in Text umgewandelt werden. Um diese Funktion mit Nutzung des Assistenten Siri zu implementieren, ist es jedoch erforderlich, dass Apple dies auch für iPads freigibt. Derzeit ist die Funktion nur auf dem iPhone 4S verfügbar.

Im Laufe der Entwicklung des Systems ist klar geworden, dass die Möglichkeit, ein Review vom Autointerieur durchzuführen, der Qualität und Aussagekraft der Befragung zugute kommen würde. Wenn die 3D-Modelle ausführlich genau gestaltet werden, müssten hierfür lediglich Erweiterungen an der Kommentarverwaltung vorgenommen werden. Außerdem muss dem Benutzer die Möglichkeit gegeben werden, zwischen Aussen- und Innenansicht zu wechseln.

Sei XCode in Version 4.2 beziehungsweise iOS 5 besteht die Möglichkeit der Nutzung von Storyboarding sowie ARC [2] [App]. Storyboarding bezeichnet eine Oberfläche, die den

[1] sprachgesteuerter Assistent auf neuen iOS Geräten. Kann Spracheingaben in Text konvertieren

[2] englisch: **A**utomatic **R**eference **C**ounting

Ablauf einer Anwendung grafisch beschreibt und bei navigationsbasierten Anwendungen die ViewController für den Entwickler steuert. Durch ARC übernimmt der Compiler die gesamte Speicherverwaltung für den Entwickler. Befehle wie `retain` oder `release` sind daher nicht mehr nötig. Memory Leaks und damit verbundenen Abstürzen sind dadurch nicht weiter möglich. Die Anwendung sollte im nächsten Schritt zunächst für diese Neuerungen angepasst werden, um die weitere Entwicklung zu vereinfachen und die Anwendung für kommende iOS Versionen zu rüsten.

Literaturverzeichnis

[Ano] ANONYM: *ZXing (SZebra Crossing")*. http://code.google.com/p/zxing/

[App] APPLE: *iOS 5 for Developers*. http://developer.apple.com/technologies/ios5/

[App10a] APPLE: *Cocoa Fundamentals Guide*. http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/CocoaFundamentals/Introduction/Introduction.html. Version: 2010

[App10b] APPLE: *Introduction to Property Lists*. http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/PropertyLists/Introduction/Introduction.html. Version: 2010

[App11a] APPLE: iOS Human Interface Guidelines. In: *Strategies* (2011). http://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf

[App11b] APPLE: *The Objective-C Programming Language*. http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html. Version: 2011

[Arn09] ARNALL, Timo: *iPhone RFID: object-based media*. http://www.nearfield.org/2009/04/iphone-rfid-nfc. Version: 2009

[BD04] BRÜGGE, Bernd ; DUTOIT, Allen H.: *Objektorientierte Softwaretechnik*. 2. Edition. München : Pearson Education Deutschland, 2004. – ISBN 3–8273–7082–5

[Blu] BLUETOOTH SIG: *Building with the Technology: Overview*. www.bluetooth.org/Building/overview.htm

[Con] CONSORTIUM, World Wide W.: *Extensible Markup Language (XML)*. http://www.w3.org/XML/

[DA10] DUDNEY, Bill ; ADAMSON, Chris: *Entwickeln mit dem iPhone SDK*. 1. Version. Köln : O'Reilly Verlag, 2010. – ISBN 978–3–89721–951–9

[DD10] DR. PROF. TAMM, Gerit ; DR. TRIBOWSKI, Christoph: *RFID*. Heidelberg : Springer-Verlag Berlin, 2010. – ISBN 978–3–642–11459–5

[Dyn11] DYNAMICS, Wireless: *iCarte*. http://www.icarte.ca/. Version: 2011

[GHJV04] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Entwurfsmuster*. München : Addison-Wesley, 2004. – ISBN 978–3–8273–2199–2

[Gro] GROUP, Khronos: *OpenGL ES - The Standard for Embedded Accelerated 3D Graphics*. http://www.khronos.org/opengles/

[Gro07] GROUP, Object M.: *OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2*. Object Management Group, 2007 (November). http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF/

[Hof08] HOFFMANN, Dirk W.: *Software-Qualität*. Heidelberg : Springer-Verlag Berlin, 2008. – ISBN 978–3–540–76323–9

[Ltd11] LTD., The Brenwill W.: *cocos3d API reference*. http://brenwill.com/docs/cocos3d/0.6.0-sp/api/. Version: 2011

[Mar08] MARTIN, Robert C.: *Clean Code*. Pearson Education, 2008. – ISBN 9780136083252

[Ope] OPENSOURCE.ORG: *Open Source Initiative OSI - The MIT License (MIT):Licensing*. http://www.opensource.org/licenses/mit-license.php

[RQZ07] RUPP, Chris ; QUEINS, Stefan ; ZENGLER, Barbara: *UML 2 Glasklar*. 3. Auflage. Carl Hanser Verlag, 2007

[Sch09] SCHMIDT, Julia: *Bleibt RFID auf Metall eine Herausforderung?* http://www.rfid-im-blick.de/200906231503/bleibt-rfid-auf-Metall-eine-herausforderung.html. Version: 2009

[SZ10] SCHÄUFFELE, Jörg ; ZURAWKA, Thomas: *Automotive Software Engineering*. 4. Auflage. Wiesbaden : Vieweg+Teubner, 2010. – ISBN 978–3–8348–0364–1

[Ver07] VEREIN DEUTSCHER INGENIEURE: *Zuverlässigkeitsmanagement VDI 4003*. http://www.vdi.de/uploads/tx_vdirili/pdf/9771310.pdf. Version: 2007

[Wan06] WANT, Roy: An Introduction to RFID Technology. In: *IEEE Pervasive Computing* (2006), Januar. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1593568. – ISSN 1536–1268

[Wava] WAVE, Denso: *About QRcode.com*. http://www.denso-wave.com/qrcode/index-e.html

[Wavb] WAVE, Denso: *QR Code Features*. http://www.denso-wave.com/qrcode/qrfeature-e.html

[Yan] YANBE, Yusuke: *Open Source QR Code Library*. http://qrcode.sourceforge.jp/

Glossary

Android
Android ist ein quelloffenes und freies Betriebssystem von dem Suchmaschinenunternehmen Google Inc.. 4.3

Apple iOS
das von Apple entwickelte Betriebssystem für mobile Geräte wie beispielsweise das iPhone, iPod Touch oder iPad. Es basiert auf dem Mac OSX. 4.1

CAPI
englisch: **C**omputer-**A**ssisted **P**ersonal **I**nterviewing, übersetzt: Rechner-unterstützte persönliche Befragung. 1.0

FURPS+
Eine Gruppe von Softwareanforderungen. Der Name besteht aus den Anfangsbuchstaben der englischen Bezeichnungen *Functionality, Usability, Reliability, Performance* und *Supportability*. Das + zeigt zusätzliche Unterkategorien an. Diese Definition stammt aus [BD04, S. 150]. 4.0

GPS
Global Positioning System. 3.1

ID
Ein zugewiesener Identifikator, um Objekte voneinander unterscheiden zu können. 3.3

Look and Feel
übersetzt: Aussehen und Handhabung. Look and Feel bezeichnet standardisierte Design-Aspekte in Software mit grafischer Benutzeroberfläche. 4.1

MIT
eine vom „Massachusetts Institute of Technology“ stammende Softwarelizenz. Sie erlaubt den freien Einsatz der unter ihr stehenden Software. Außerdem ist diese Software auch mit keinem Copyright versehen, wodurch auch Quellcode bearbeitet werden darf.. 1.0

MVC
englisch: **M**odel **V**iew **C**ontroller, übersetzt: Modell-Präsentation-Steuerung, strukturiertes Architekturmuster in der Softwareentwicklung in drei Einheiten. 1.2

Open Source
eine Sammlung an Softwarelizenzen, die den Quellcode öffentlich zugänglich macht.. 3.4

OpenGL
englisch: **Open G**raphics **L**ibrary ist eine plattform- und programmiersprachenunabhängige Programmierschnittstelle für 2D und 3D Computergrafiken.. 1.1

Organisator
eine Person, die vollen Zugriff auf das System hat und alle Daten verwalten und auswerten kann. Diese Personengruppe wird auch oft als System Administrator bezeichnet.. 5.2

QR Code
englisch: **Q**uick **R**esponse, übersetzt: schnelle Antwort. 3.4

RFID
englisch: radio-frequency identification, übersetzt: Identifizierung mit Hilfe elektromagnetischer Wellen. 3.2

UML
Unified Modeling Language. 3.0

XML
englisch: E**x**tensible **M**arkup **L**anguage, übersetzt: erweiterbare Auszeichnungssprache, Darstellung hierarchisch strukturierter Daten in Form von Textdaten. 2.0

Anhang

A. Bildschirmmasken der prototypischen Implementierung

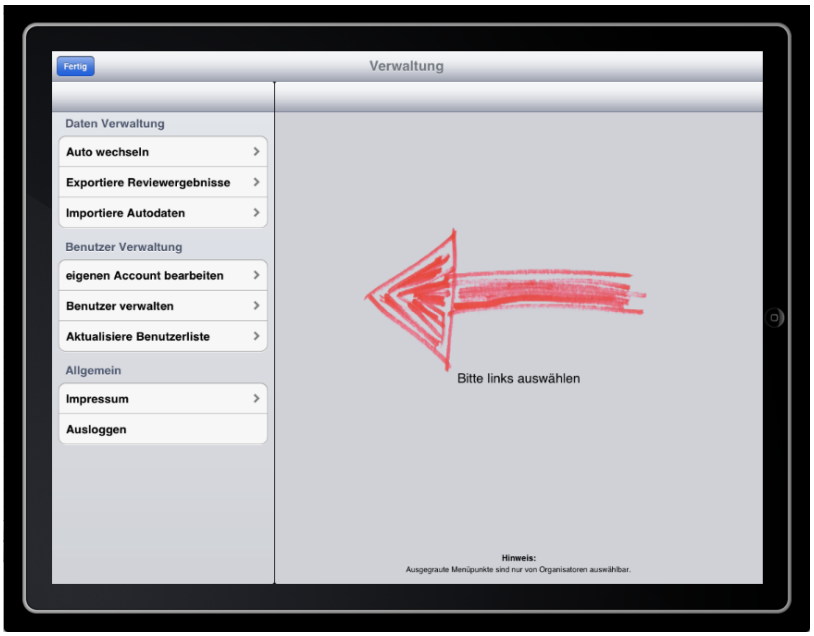


Abbildung A.1.: Verwaltungsansicht eines Organisators

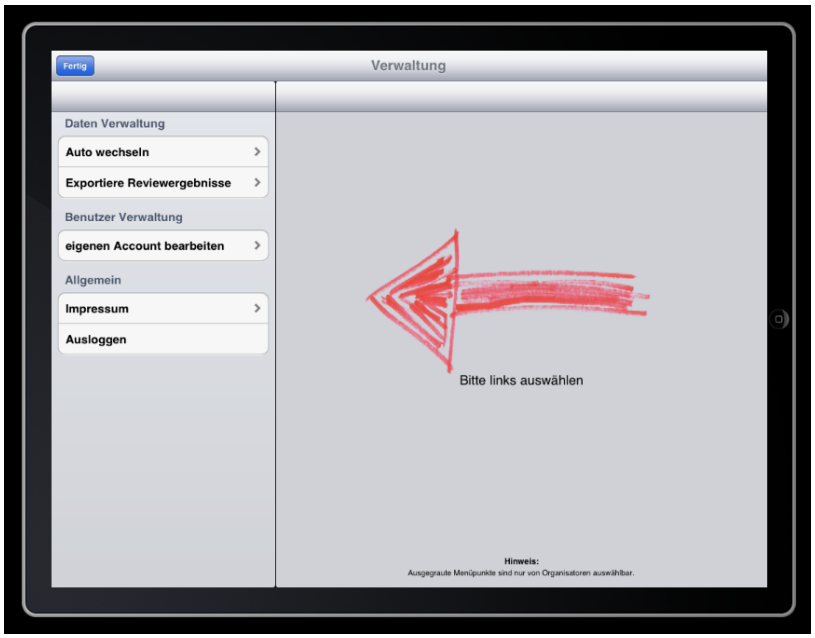


Abbildung A.2.: Verwaltungsansicht eines Reviewers

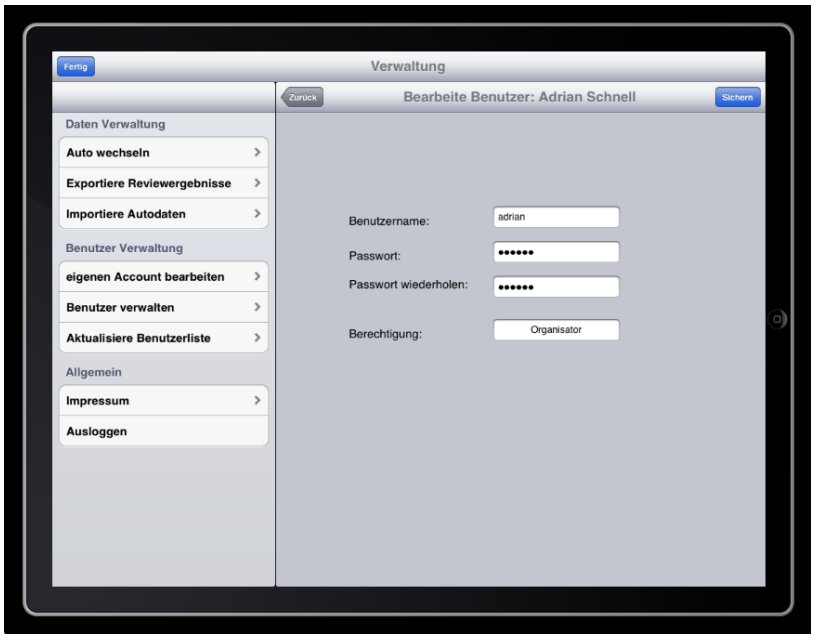


Abbildung A.3.: Eingabemaske zur Bearbeitung eines Benutzers

Abbildung A.4.: Beispielhafte Darstellung der Optionen, die für den Reifen zur Verfügung stehen

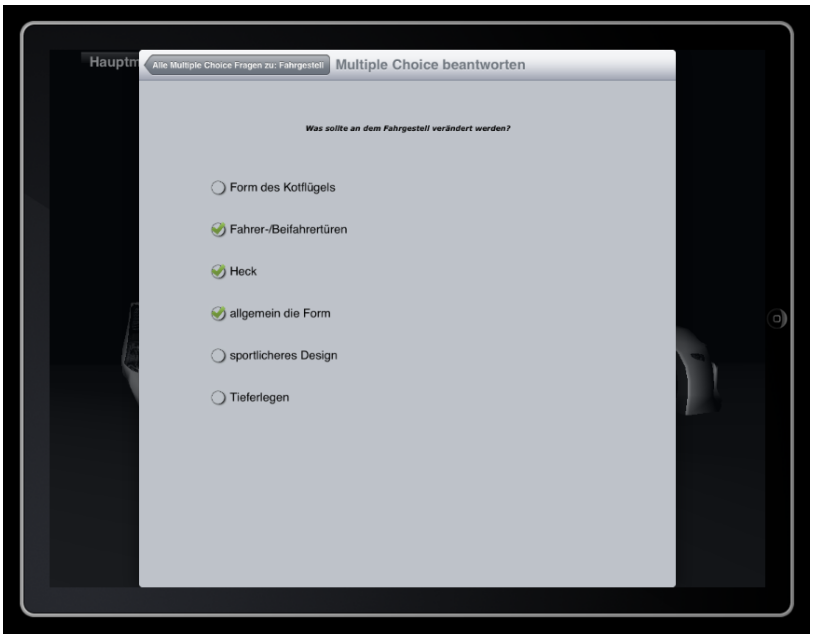


Abbildung A.5.: Eingabemaske für die Beantwortung einer Multiple Choice Frage

Abbildung A.6.: Übersicht der verfügbaren Informationen zu diesem 3D-Automodell

B. „Lines of Code“

Die grobe Komplexität eines Softwareprojektes wird durch die LOC [1] beschrieben [Hof08]. Diese Maßzahl wird durch Addition der einzelnen Code-Zeilen aller Quelldateien berechnet.

Diese Aussage ist jedoch recht fragwürdig. So sagt beispielsweise eine LOC von 5000 nichts aus, wenn davon 4500 Zeilen eine Ausgabe auf die Konsole bewirken. Sinnvoller wäre, die LOC im Verhältnis zu der Klassenanzahl und deren Methoden zu stellen. Damit lässt sich beispielsweise feststellen, ob die Länge der Methoden zu groß ist. Dies entspricht laut Martin [Mar08] einem Programmcode.

Der Prototyp wurde ausschließlich in Objective C entwickelt. - Auf eine ausführliche Dokumentation wurde mit 850 Zeilen hohen Wert gelegt. Die Implementierung selbst umfasst ca. 2950 Zeilen. Eine genaue Auflistung ist in Tabelle B.1 dokumentiert.

Sprache	**Dateien**	**Leerzeilen**	**Kommentare**	**LOC**
Objective C	26	677	654	2554
C/C++ Header	26	118	200	407
Summe	52	795	764	2961

Tabelle B.1.: LOC des Projekts „ReviewApp“ exklusiv des Grafikframeworks Cocos3D

[1] englisch: **L**ine **o**f **C**ode

C. Eingesetzte Software

C.1. Software zur Erzeugung dieses Dokumentes

MacTEX
MacTEX ist eine LaTeX Distribution, um TeX Texte auf MacOS X zu kompilieren. Es wurde als Basis zu texmaker verwendet.

texmaker
texmaker ist ein komfortabler Open Source LaTeX Editor, der neben Syntax Highlightning und Autovervollständigung von LaTeX-Befehlen viele weitere Tools zur Verfügung stellt.

Adobe Photoshop CS5
Adobe Photoshop ist eine professionelle Software zur Foto- und Bildbearbeitung. Einige in dieser Arbeit enthaltenen Fotos wurden mit Photoshop in der Version CS5 erstellt beziehungsweise bearbeitet.

C.2. Softwareentwicklung

XCode 4
XCode ist die von Apple entwickelte IDE (integrierte Entwicklungsumgebung) für die Programmiersprache „Objective-C“. Der gesamte Programmcode für diese Arbeit wurde ausschließlich mit XCode 4 erstellt.

Adobe Photoshop CS5
Adobe Photoshop ist eine professionelle Software zur Foto- und Bildbearbeitung. Die in dieser Arbeit enthaltenen grafischen Elemente wurden mit Photoshop in der Version CS5 erstellt.

Visual Paradigm
Visual Paradigm wurde eingesetzt, um das in dieser Arbeit enthaltene Sequenzdiagramm zu erstellen.

C.3. Quellcode-Verwaltung

sourceTree Als Sicherung und gleichzeitige Versionierung des Quellcodes wurde die Software sourceTree eingesetzt, um die GIT Repositories zu verwalten.