SEP- Hauptaufgabe

Projektmappe des Projektes  
SEPTrash

*Spezifikation des Projektes*

Gruppe A:

David Reuter

Florian Bongers

Matthias Buttgereit

Stephan Lehmich

Tobias Dörnemann

Vanessa Vogt

*Systemdesign des Projektes*

Gruppe B:

Christian Schmidt

Jonathan Liebers

Michael Krane

Mike van Afferden

Nils Dechant**Einleitung**

Dieses Dokument enthält alle nötigen Informationen zur Erstellung eines Software-Produktes. Die Erstellung eines Software-Produktes wird im Allgemeinen auch als Programmierung bezeichnet. Programmierung kann man aber auch dahingehend verstehen, dass ein Computer zur Durchführung eines bestimmten Verhaltens konfiguriert werden muss.[[1]](#footnote-1) Vorher müssen jedoch alle nötigen Informationen über das „bestimmte Verhalten“ zusammengetragen und dokumentiert werden. Diese Informationen bestehen aus Anforderungen (zu neudeutsch *Requirements*), Architekturbeschreibungen, etc., die im Folgenden in diesem Dokument wiedergegeben werden.

Dieses Dokument soll euch durch den gesamten Verlauf der Hauptaufgabe begleiten und dazu dienen, euer Projekt zu dokumentieren. Jeder Abschnitt beschäftigt sich mit einem Teilaspekt eurer Aufgabenstellung (Anforderungen, Architektur, Projektplanung, Testfälle etc.). An vielen Stellen findet ihr im Dokument folgendes Kästchen:

Dies ist eine Hilfestellung.

Diese Kästen dienen dazu, euch kurze Informationen über Ziele und Inhalte der jeweiligen Abschnitte zu geben. Sie sind spätestens zur finalen Abgabe der Projektmappe vollständig zu entfernen. Betrachtet dieses Dokument bitte nicht als Aufgabe, die man von oben nach unten abarbeiten soll; es soll vielmehr als durchgängige Dokumentation eurer Projektarbeit dienen und fortlaufend erweitert bzw. angepasst werden, sodass am Ende des SEPs eine Software entstanden ist, die sich in diesem Dokument wieder findet.

**Eine Anmerkung in eigener Sache**

Dieses Dokument soll keinen starren Rahmen vorgeben, sondern vielmehr eine Richtschnur für eure Arbeit sein. Wenn Ihr Abschnitte anders gestalten wollt, so könnt ihr dies gerne tun (grobe Änderungen bitte mit eurem Betreuer absprechen, außerdem nur strukturelle Änderungen auf den Ebenen unter der ersten Strukturierungsebene (1, 2, 3, …) durchführen). Ferner ist dieses Dokument keineswegs vollständig oder erhebt Anspruch auf Perfektion. Wenn ihr Anmerkungen und/oder Verbesserungsvorschläge habt, dann könnt ihr diese gerne an euren Betreuer weitergeben. Wir werden sie dann in das Vorlagedokument übernehmen.

Das SEP-Team wünscht euch  
**viel Erfolg**  
bei der Bearbeitung der Hauptaufgabe!!!

**Inhaltsverzeichnis**

[1 Projektbeschreibung (SEPTrash) 4](#_Toc361167333)

[2 Anforderungsspezifikation 5](#_Toc361167334)

[2.1 Zielmodell 5](#_Toc361167335)

[2.2 Ebene 1: Gesamtsystemebene 6](#_Toc361167336)

[2.2.1 Szenarien 6](#_Toc361167337)

[2.2.2 Textuelle Anforderungen 7](#_Toc361167338)

[2.2.3 Kontextmodell 9](#_Toc361167339)

[2.2.4 GUI-Papierprototyp 10](#_Toc361167340)

[2.3 Ebene 2: Logisches System 13](#_Toc361167341)

[2.3.1 Message Sequence Charts 13](#_Toc361167342)

[2.3.2 Datenflussdiagramm 17](#_Toc361167343)

[3 Technisches Systemdesign 19](#_Toc361167344)

[3.1 Ebene 3: Technisches System 19](#_Toc361167345)

[3.1.1 Datenmodell 19](#_Toc361167346)

[3.1.2 Technische Komponenten 21](#_Toc361167347)

[4 Testartefakte 23](#_Toc361167348)

[4.1 Modultest 23](#_Toc361167349)

[4.1.1 Testspezifikation 23](#_Toc361167350)

[4.1.2 Testergebnisse 23](#_Toc361167351)

[4.2 Systemtest 25](#_Toc361167352)

[4.2.1 Testspezifikation 25](#_Toc361167353)

[4.2.2 Testergebnisse 26](#_Toc361167354)

[5 Issue-Liste 31](#_Toc361167355)

[6 Implementierungs-protokoll 32](#_Toc361167356)

# Projektbeschreibung (SEPTrash)

Die Aufgabe „SEPTrash“ besteht in der Entwicklung einer Anwendung zur Planung eines Mülltonnenleerungsplans für beliebige Städte. Die Anwendung soll helfen, einen Leerungsplan für eine Stadt anhand der Art der Straßen, der aktuellen Verkehrssituation (z.B. durch Straßenverengungen in Baustellen oder Komplettsperrungen), der verfügbaren Fahrzeuge und Fahrer, dem Ort der Entleerung und der Tonnenart zu erstellen.

Auf Basis eines Straßennetzes von OpenStreetMaps haben unterschiedliche Straßen eine unterschiedliche Dichte von zu entleerenden Mülltonnen. Während auf Hauptstraßen die Dichte der Mülltonne evtl. höher ist (z.B. eine 120 Liter Tonne alle 40 Meter), ist die Dichte auf Anwohnerstraßen möglicherweise geringer (z.B. eine 80 Liter Tonne alle 50 Meter). Ausgangspunkt für die Räumung ist das Fahrzeugdepot, in dem nur eine begrenzte Anzahl von Fahrzeugen zur Verfügung steht. Alle Fahrzeuge können nur eine begrenzte Menge Müll abtransportieren, sodass der Müllwagen regelmäßig an einer gewissen Stelle entleert werden muss (z. B. an Müllkippen oder einer Müllverbrennungsanlage). Außerdem sollen unterschiedliche Fahrzeugtypen mit unterschiedlicher Ladekapazität und/oder Straßeneignung existieren.

Die Anwendung soll auf Basis dieser Daten automatisch einen (z.B. wöchentlichen) Entleerungsplan berechnen und den einzelnen Fahrzeugen Routen vorgeben. Dabei müssen die o. g. Punkte berücksichtigt werden. Es soll auch möglich sein, die einzelnen Pläne manuell anzupassen. Zudem muss der vorhandene Fuhrpark mit den zur Verfügung stehenden Fahrern (und zusätzlichem Personal) gemanagt werden. Auf einer Karte soll es möglich sein, das Fahrzeugdepot sowie die Position der Mülldepots zu definieren.

Die berechneten bzw. angepassten Pläne sollen auf einer Karte angezeigt werden, wobei es auch möglich sein soll, gezielt für einzelne Fahrzeuge die Routen anzeigen zu lassen.

*Anmerkungen:*

* Die Anwendung ist in der Programmiersprache Java zu entwickeln.
* Straßenkarten können von OpenStreetMaps genutzt werden. Es soll möglich sein, auch andere Kartenausschnitte mit der Anwendung zu nutzen.
* Wir veröffentlichen besonders gelungene Software auf unserer SEP-Webseite. Hierzu ist es unbedingt erforderlich, dass ein System keine urheberrechtlich geschützten Inhalte (Bilder, Musik, etc.) enthält.

# Anforderungsspezifikation

## Zielmodell

* **Z-HG-1:** Erstellung Mülltonnenleerungsplan  
  Das System erstellt einen Mülltonnenleerungsplan.
  + **Z-L-HG-1.1:** Entleerungsgewährleistung  
    Der Mülltonnenleerungsplan gewährleistet die Entleerung aller Mülltonnen aller Müllarten innerhalb einer Woche.
  + **Z-L-HG-1.2:** Routenplanung  
    Das System berechnet die von den Müllfahrzeugen zu befahrenden Routen.
    - **Z-L-HG-1.2.1:** Dateneingaben

Der Benutzer kann Eingaben machen, die das System bei der Planerstellung berücksichtigt.

* + - * **Z-L-HG-1.2.1.1:** Kartenimport

Der Benutzer muss eine Karte importieren, bevor er andere Eingaben machen kann.

* + - * + **Z-T-HG-1.2.1.1.1:** OpenStreetMap XML-Dateien

Das System soll den Import von Kartenausschnitten aus XML-Dateien des OpenStreetMap Projektes unterstützen.

* + - * + **Z-T-HG-1.2.1.1.2:** Umwandlung in eigene Struktur

Das System soll den importierten Kartenausschnitt in einen Graphen transformieren, auf dem das System im Folgenden weiterarbeitet.

* + - * **Z-L-HG-1.2.1.2:** Anfahrtspunkte

Der Benutzer muss die Position des Fahrzeugdepots und der Müllentleerungsorte festlegen.

* + - * + **Z-L-HG-1.2.1.2.1:** Anfahrtspunkte ändern

Der Benutzer kann manuelle Änderungen an den Positionen des Fahrzeugdepots und der Müllentleerungsorte vornehmen.

* + - * **Z-L-HG-1.2.1.3:** Fahrzeuge

Der Benutzer muss Fahrzeuge (mindestens ein Fahrzeug je Müllart) anlegen.

* + - * + **Z-L-HG-1.2.1.3.1:** Verfügbarkeit

Der Benutzer ist in der Lage die Verfügbarkeit von Müllfahrzeugen bzw. Teams zu bestimmen.

* + - **Z-L-SG-1.2.2:** Straßenverhältnisse  
      Zu berücksichtigen sind die Straßenverhältnisse.
      * **Z-HG-1.2.2.1:** Allgemeine Hindernisse  
        Das System berücksichtigt im Kartenmaterial hinterlegte Hindernisse bei der Planung.
      * **Z-L-HG-1.2.2.2:** Nicht erreichbare Straßenabschnitte

Das System berücksichtigt aufgrund von Hindernissen nicht erreichbare Straßenabschnitte, indem es Tonnen, die im nicht befahrbaren Straßenabschnitt stehen, auf den nächstgelegenen, anfahrbaren Knoten verlegt.

* + - **Z-L-HG-1.2.3:** Fassungsvermögen  
      Zu berücksichtigen ist das Fassungsvermögen (abhängig vom Fahrzeugtyp) des Fahrzeugs.
    - **Z-L-SG-1.2.4:** Umwege  
      Das System soll Umwege und unnütze Fahrten vermeiden.
      * **Z-L-HG-1.2.4.1:** Startpunkt  
        Jede Fahrt beginnt am Depot.
      * **Z-L-HG-1.2.4.2:** Zuweisung Müllentleerungsort  
        Für die Entleerung berücksichtigt das System den für das Fahrzeug nächstgelegenen Müllentleerungsort entsprechender Müllart.
      * **Z-L-HG-1.2.4.3:** Kurze Strecke  
        Die Route ist so zu berechnen, dass sie möglichst kurz ist.
  + **Z-L-HG-1.3:** Manuelle Änderung  
    Der Benutzer kann manuelle Änderungen am Mülltonnenleerungsplan bzw. an der Routenplanung vornehmen.
  + **Z-L-HG-1.4:** Speicherung  
    Das System soll Dateneingaben, manuelle Änderungen und den fertigen Plan speichern.
    - **Z-T-HG-1.4.1:** Speicherung der Fahrzeugdaten  
      Das System soll die vorhandenen Fahrzeuge und ihre Attribute in Form einer ser-Datei auf dem lokalen Rechner speichern.
    - **Z-T-HG-1.4.2:** Speicherung des Kartenausschnitts und der Anfahrtspunkte  
      Das System soll die jeweils aktuelle Version des Graphen in Form einer .ser-Datei auf dem lokalen Rechner speichern.
* **Z-HG-2:** Ausgabe  
  Das System soll alle Routen mitsamt Depot, Müllentleerungsorten und Tonnen auf einer Karte anzeigen.
  + **Z-L-HG-2.1:** Ausgabe aller Routen für eine bestimmte Müllart  
    Es soll möglich sein, alle Routen, die nur die Entsorgung einer einzelnen Müllart (z. B. nur Restmüll) betreffen, mit Müllentleerungsort(en) anzuzeigen.
  + **Z-L-HG-2.2:** Ausgabe einer einzelnen Route  
    Es soll möglich sein, nur eine einzelne Route von Müllfahrzeug mit Müllentleerungsort(en) auf einer Karte anzuzeigen.
  + **Z-L-HG-2.3:** Ausgabe nach Tagen

Es soll möglich sein, nur die Routen für einen einzelnen Wochentag anzuzeigen.

## Ebene 1: Gesamtsystemebene

### Szenarien

#### Szenario 1: Dateneingabe (Z-1.2.1x, Z-1.4)

1. Der Benutzer importiert die Karte, auf welcher die Berechnung stattfinden soll.

Außerdem legt er die Positionen des Fahrzeugdepots und der Müllentleerungsorte fest.

Darüber hinaus legt er Fahrzeuge, inklusive ihrer Teams, an.

1. Das System speichert diese Eingaben.

#### Szenario 2: Manuelle Änderung (Z-1.2.1.3x, Z-1.3x)

1. Das System hat einen Routenplan berechnet.
2. Der Benutzer fügt ein neues Fahrzeug hinzu.

Weiterhin bearbeitet er ein bestehendes Fahrzeug und löscht ein weiteres.

Außerdem deaktiviert der Benutzer einige Fahrzeuge, da wegen Krankheitsfällen Personal ausfällt.

1. Das System speichert diese Änderung, anschließend berechnet es die Route neu.
2. Der Benutzer passt den Verlauf einer Route manuell an.
3. Das System speichert diese Änderung.

#### Szenario 3: Routenberechnung (Z-1.1, Z-1.2x, Z-1.3x, Z-1.4)

1. Der Benutzer gibt dem System den Befehl die Route zu berechnen.
2. Anhand aller gespeicherten Eingaben und manuellen Änderungen berechnet das System den Mülltonnenleerungsplan, wobei das System alle Sorten Müll berücksichtigt.

Das System speichert den fertig berechneten Plan.

#### Szenario 4: Visualisierung (Z-1x, Z-2x)

1. Das System zeigt dem Benutzer eine visuelle Darstellung des Plans an.
2. Der Benutzer stellt ein, nur die Fahrten der Müllart *Restmüll* und damit auch den entsprechenden Müllentleerungsort anzuzeigen.
3. Das System zeigt den neuen, angepassten Plan an.
4. Weiterhin wählt der Benutzer nur ein bestimmtes Fahrzeug aus.
5. Das System zeigt nur noch die Route dieses Fahrzeugs an, die anderen blendet es aus.
6. Des Weiteren wählt der Benutzer *Montag* zur Anzeige aus.
7. Das System zeigt nur noch die Route eines Fahrzeugs am *Montag* am.

### Textuelle Anforderungen

#### QA-1: Performanz

Die Berechnung des Plans soll maximal 120 Sekunden dauern.

#### QA-2: Benutzbarkeit

Das System soll den erstellten Plan oder das unbearbeitete Kartenmaterial in der Anzeige ausgeben.

1. Unterschiedliche Routen sollen unterschiedlich gefärbt sein.
2. Kontextabhängige Visualisierung

2.1 Der Benutzer kann einzelne Fahrzeuge auswählen, um nur deren Routen angezeigt zu bekommen.

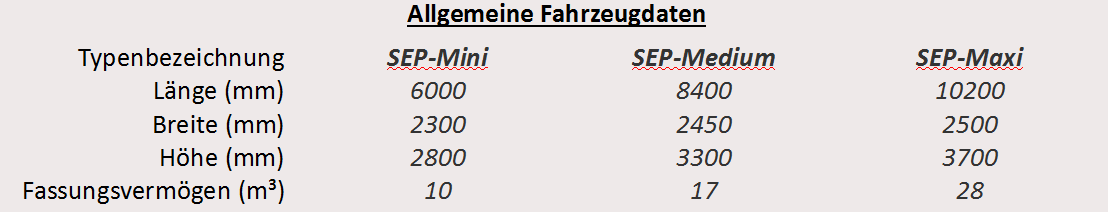
2.2 Außerdem kann er die Anzeige auf die Routen ausgewählter Tage beschränken...

2.3 ...sowie auf die Routen für ausgewählte Müllarten.

1. Die Karte zeigt ebenfalls das Fahrzeugdepot und die Müllentleerungsorte an.

#### FA-1: Fahrzeugverwaltung

Es gibt drei verschiedene Fahrzeugtypen (SEP-Mini, SEP-Medium, SEP-Maxi), welche jede Müllart entsorgen können.



1. Der Benutzer gibt den Fahrzeugtyp und ein Team für jedes Fahrzeug an.
2. Der Benutzer kann eine Liste mit allen im System befindlichen Fahrzeugen aufrufen.
3. Der Benutzer kann angelegte Fahrzeuge bearbeiten und löschen.

#### FA-2: Ausfallmanagement

Das System soll eine manuelle Änderung bei der Verfügbarkeit der Fahrzeuge (Teams) zulassen.

* 1. Der Benutzer kann jederzeit Änderungen bzw. Löschungen bei den Teams bzw. Fahrzeugen vornehmen.
  2. Das System berücksichtigt die Änderung bzw. Löschung von Fahrzeugen (Teams).

#### FA-3: Planberechnung

Der Plan soll die Entleerung aller Tonnen aller Müllarten innerhalb einer Woche gewährleisten.

1. Das System kann den Plan erst erstellen, wenn mindestens ein Fahrzeugdepot und ein Müllentleerungsort je Müllart im Planungsbereich festgelegt wurden.
2. Das System schließt bei der Berechnung des Plans ausgefallene Fahrzeuge von der Planung aus.
3. Die Routen sind so geplant, dass die Müllfahrzeuge planmäßig bei einer errechneten Füllmenge von 95% einen Müllentleerungsort anfahren.

#### FA-4: Hindernisse

Das System berücksichtigt Hindernisse bei der Planberechnung.

1. Ein Hindernis blockiert die Durchfahrt einer Straße, aber keinen Straßenabschnitt. Das System ändert die Route so, dass die Müllfahrzeuge mit den geringsten Umwegen alle Tonnen entleeren.
2. Ein oder mehrere Hindernisse blockieren einen Straßenabschnitt, sodass dieser Straßenabschnitt nicht befahrbar ist. Das System gibt eine Warnmeldung aus, dass der Benutzer die Anwohner informieren muss, dass die Anwohner vom betroffenen Straßenabschnitt die Mülltonnen zum nächstgelegenen anfahrbaren Anfahrtspunkt (=Mülltonne) bringen müssen. Das System rechnet an diesen Anfahrtspunkten dann mit einer größeren Menge Müll.

#### FA-5: Manuelle Änderungen

Es soll möglich sein, Änderungen an dem berechneten Plan bzw. an den Eingaben vorzunehmen.

1. Manuelle Änderungen betreffen Änderungen an den Routen.
2. Auf Änderungen am Plan, die nicht realisierbar sind, reagiert das System mit einer Fehlermeldung.
3. Der Benutzer kann Änderungen an Anfahrtspunkten oder Fahrzeugdaten nach der Eingabe vornehmen.

#### FA-6: Mülltonnendichte

Das System berechnet für Hauptstraßen eine 120l Mülltonne alle 40m, für Anwohnerstraßen eine 80l Mülltonne alle pro 50m. Auf Autobahnen und Landstraßen gibt es keine Mülltonnen.

#### FA-7: Müllarten

Es gibt mindestens die drei verschiedene Müllarten Restmüll, Papier und Plastik.

#### FA-8: Kartenimport

Es soll möglich sein, Kartenmaterial zu importieren.

#### FA-9: Ausgabe

Der erstellte Plan oder das Kartenmaterial soll dem Benutzer in Form einer Anzeige ausgegeben werden (siehe QA-2 Benutzbarkeit).

#### RB-1: Programmiersprache Java

Das Programm soll in der Programmiersprache Java programmiert sein.

#### RB-2: Abgabetermin

Das Programm muss am 09.07.2013 fertiggestellt sein.

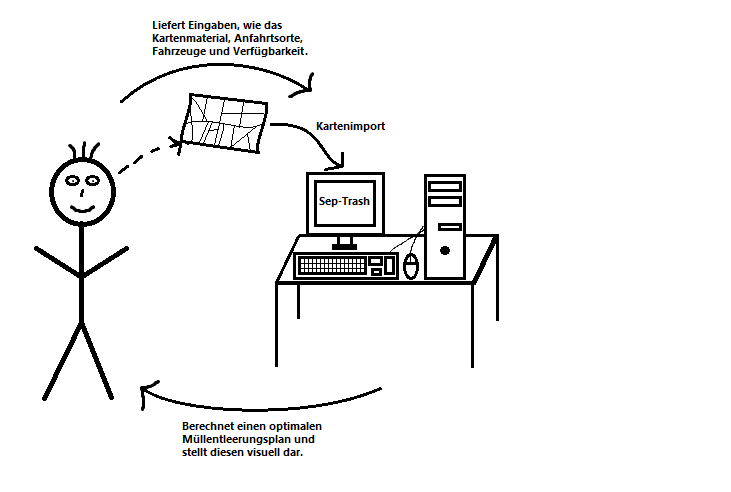
#### RB-3: Urheberrecht

Das Programm darf keine urheberrechtlich geschützten Inhalte enthalten.

#### RB-4: Testplattform

Das Programm soll auf den PCs in Raum SA202 unter Berücksichtigung der QA-1 ausführbar sein.

### Kontextmodell



#### Akteur: Benutzer

Der Benutzer ist die Person, der das System steuert. Er sorgt für die nötigen Dateneingaben und gibt den Befehl zur Routenberechnung. Der Benutzer hat die Möglichkeit seine Eingaben zu ändern bzw. löschen bzw. manuelle Änderungen an der Route vorzunehmen.

#### Akteur: System

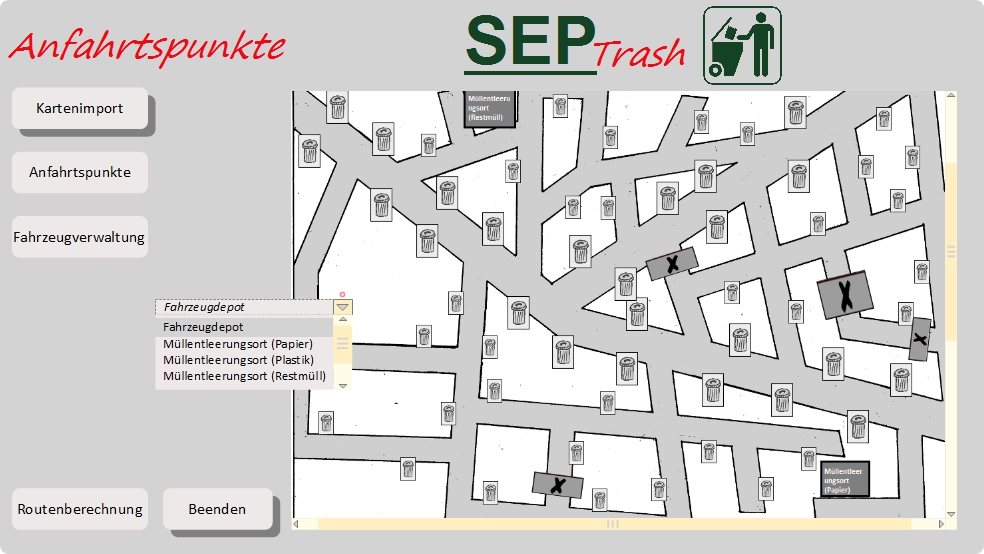
Das System ist für die Verarbeitung der Dateneingaben bzw. manuellen Änderungen des Benutzers zuständig. Es weiß, wie die Daten zu verarbeiten sind und erstellt so einen entsprechend optimalen Müllentleerungsplan. Das System speichert nach Aufforderung durch den Benutzer manuelle Änderungen bzw. Eingaben und berechnet den Plan neu.

### GUI-Papierprototyp

#### Screen „Startscreen“

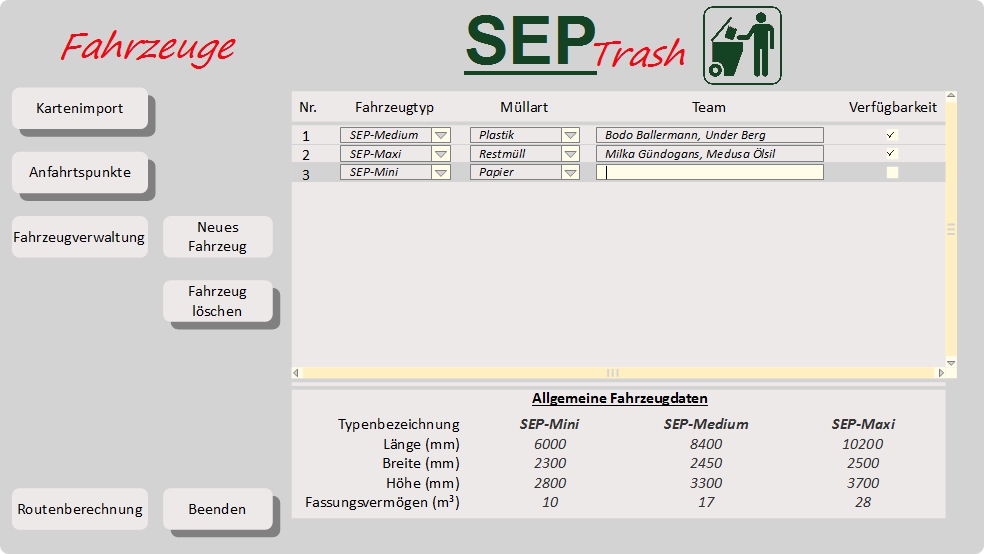


#### Screen „Anfahrtspunkte“



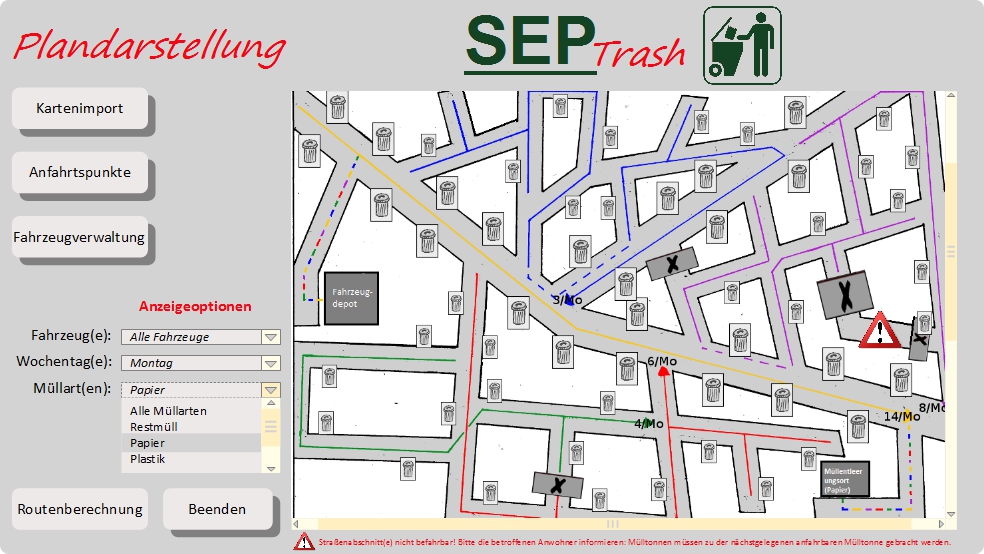
* Der Button „Kartenimport“ öffnet einen Dateibrowser zur Auswahl des Kartenmaterials.
* Anwählbare Menüpunkte sind mit Schatten hinterlegt.
* Unter „Anfahrtspunkte“ kann der Benutzer nach Wahl eines Knotens aus einem "DropDown"-Menü auswählen, ob es sich bei dem Knoten um ein Fahrzeugdepot oder um einen Müllentleerungsort (plus Müllart) handelt.
* Das System verschiebe automatisch Mülltonnen, die durch ein oder mehrere Hindernisse nicht anfahrbar sind.

#### Screen „Fahrzeugverwaltung“ 🡪 „Neues Fahrzeug“



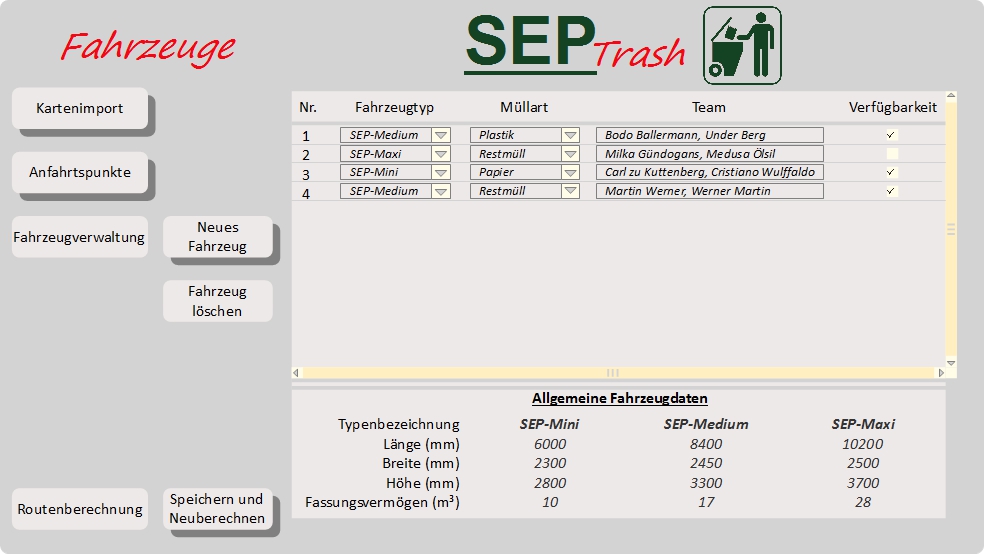
* Den Fahrzeugtypen und die Müllart kann der Benutzer aus einem "DropDown"-Menü auswählen.
* Das Feld „Team“ erlaubt beliebige Eingaben von Namen und ist nur für den Benutzer von Belang.
* Die Verfügbarkeit legt der Benutzer mittels einer "CheckBox" fest. Ein gesetzter Harken bedeutet, dass das Fahrzeug verfügbar ist.
* Nach Auswahl einer Zeile wird diese markiert und der Button "Fahrzeug löschen" steht zur Verfügung.
* Das System nummeriert die Fahrzeuge automatisch und passt sie bei der Löschung eines Fahrzeugs an.

#### Screen „Plandarstellung“ (nachdem die Route berechnet wurde)



* Der Benutzer kann manuelle Änderungen an den Routen durch das "Verschieben" von Mülltonnen vornehmen. Nach einer manuellen Änderung steht anstelle des Buttons "Beenden" der Button "Speichern und Neuberechnen" zur Verfügung.
* Sind Straßenabschnitte nicht befahrbar, stellt das System dies durch ein Symbol in der Karte und einen blinkenden Hinweis unterhalb der Karte visuell dar.

#### Screen „Fahrzeugverwaltung nach Routenberechnung“



* Nimmt der Benutzer Änderungen vor, so wird auch hier der Button "Beenden" zu "Speichern und Neuberechnen" und erlaubt die Neuberechnung des Plans anhand der aktualisierten Fahrzeugliste.

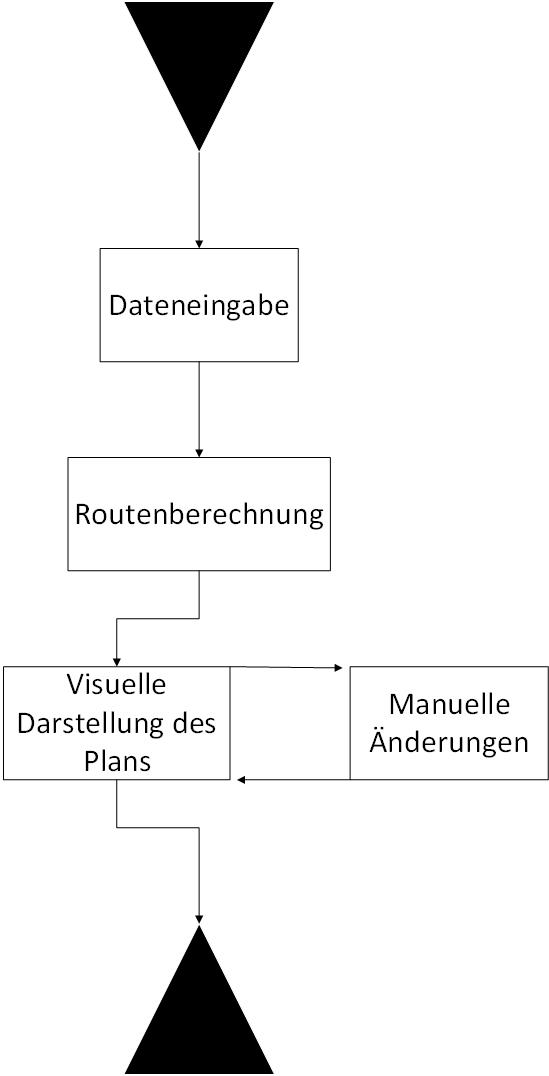
## Ebene 2: Logisches System

### Message Sequence Charts

#### **Abbildung der Szenarien auf Message Sequence Charts**

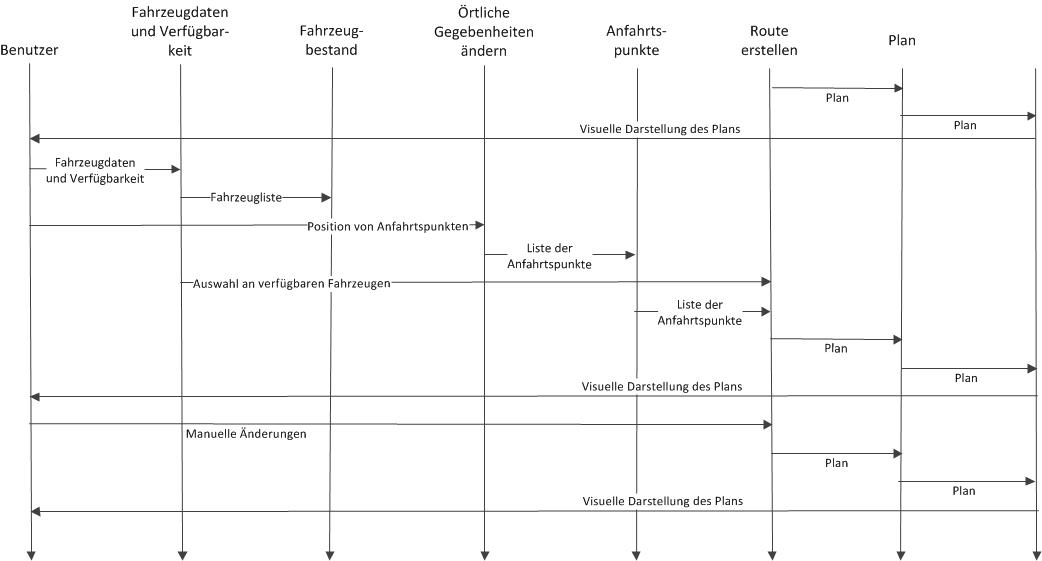
|  |  |
| --- | --- |
| Dateneingabe | bMSC-1: Dateneingabe |
| Manuelle Änderung | bMSC-2: Manuelle Änderung |
| Routenberechnung | bMSC-3: Routenberechnung |
| Visualisierung des fertigen Plans | bMSC-4: Visualisierung des fertigen Plans |

#### hMSC

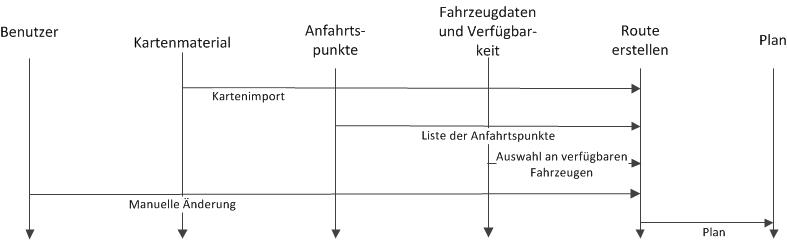


#### bmsc1_v5.jpgbMSC-1: Dateneingabe

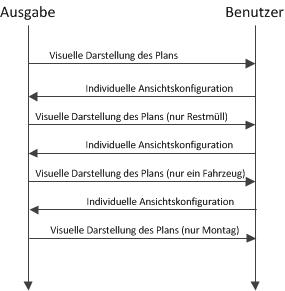
#### bmsc1_v5.jpgbMSC-2: Manuelle Änderung



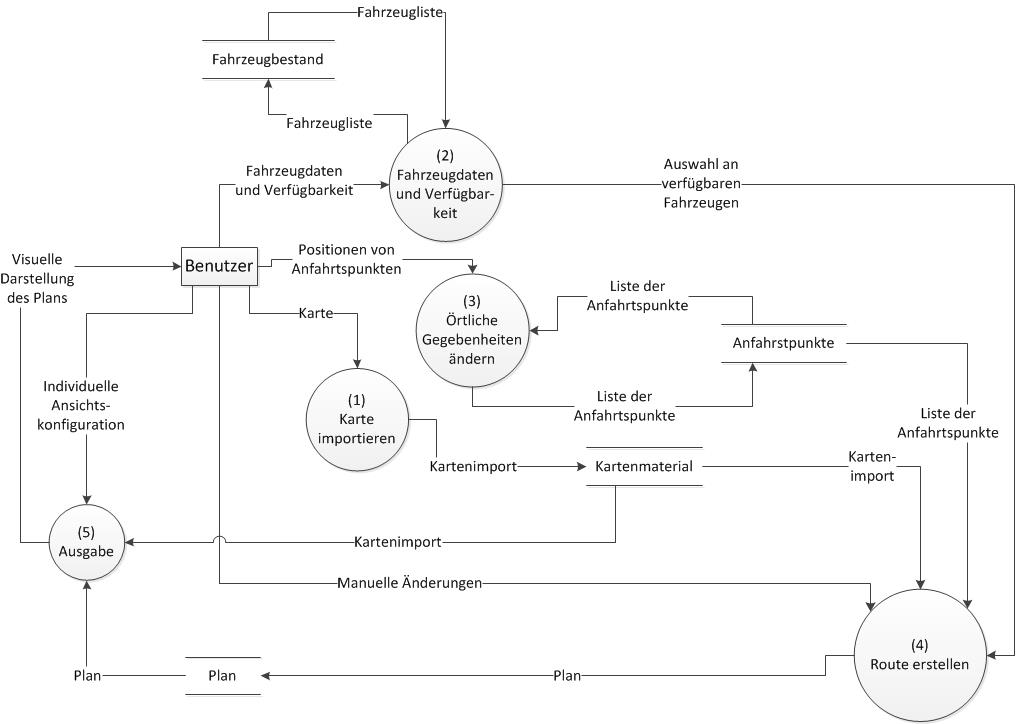
#### bMSC-3: Routenberechnung



#### bMSC-4: Visualisierung des fertigen Plans



### Datenflussdiagramm



#### P-1: Karte importieren

Der Prozess "Karte importieren" importiert eine Karte, auf der die Berechnung des Planes stattfinden soll. Er speichert das importierte Kartenmaterial dann in einem Datenspeicher mit eben diesem Namen.

#### P-2: Fahrzeugdaten und Verfügbarkeit

Der Prozess "Fahrzeugdaten und Verfügbarkeit" erlaubt dem Benutzer, sowohl vorhandene als auch neue Fahrzeugdaten zu verwalten. Der Benutzer hat die Möglichkeit, die Verfügbarkeit zu ändern.

#### P-3: Örtliche Gegebenheiten ändern

Der Prozess "örtliche Gegebenheiten ändern" bekommt vom Benutzer die Eingaben "Positionen von Anfahrtspunkten“. Der Prozess verarbeitet die Daten durch die Speicherung der Anfahrtspunkte (Depot, Müllentleerungsorte und Mülltonnen) im gleichnamigen Datenspeicher.

#### P-4: Route erstellen

Dieser Prozess ist das Kernstück des gesamten Programmes. Er liefert die Daten: Auswahl von verfügbaren Fahrzeugen (von P-2), Position von Anfahrtspunkten (von P-3), den Kartenimport (von P-1) und eventuelle manuelle Änderungen vom Benutzer. Anhand dieser Informationen erstellt er die entsprechenden Routen für einen einwöchigen Entleerungsplan, wobei er Straßenverhältnisse bzw. Hindernisse sowie die rechtzeitige Entleerung der Fahrzeuge berücksichtigt und möglichst keine Umwege einplant. Nach der Berechnung speichert der Prozess den fertigen Plan in dem Datenspeicher „Plan“.

#### P-5: Ausgabe

"Ausgabe" bekommt vom Speicher „Plan“ den fertig erstellten Plan oder das Kartenmaterial geliefert und erstellt eine Visualisierung für den Benutzer. Der Benutzer kann nach der Dateneingabe individuelle Ansichtskonfigurationen einstellen, die der Prozess verarbeitet, um so verschiedene Möglichkeiten der Darstellung zu nutzen. Entsprechend der individuellen Ansichtskonfigurationen erfolgt eine visuelle Darstellung.

#### **Abbildung der DFD-Prozesse auf funktionale Anforderungen**

|  |  |
| --- | --- |
| P-1: Karte importieren | FA-8 Karten |
| P-2: Fahrzeugdaten und Verfügbarkeit | FA-1 Fahrzeugverwaltung  FA-2 Ausfallmanagement FA-5 Manuelle Änderungen |
| P-3: Örtliche Gegebenheiten ändern | FA-5 Manuelle Änderungen |
| P-4: Route erstellen | FA-3 Planberechnung  FA-4 Hindernisse  FA-5 Manuelle Änderungen  FA-6 Mülltonnendichte |
| P-5: Ausgabe | FA-9 Ausgabe |

# Technisches Systemdesign

## Ebene 3: Technisches System

### Datenmodell



#### Müllentleerungsort

Ein Müllentleerungsort (siehe Z-1.2.4.2) ist ein Ort, an dem die Müllwagen regelmäßig ihren Müll entleeren, wobei ein Müllentleerungsort immer nur eine bestimmte Sorte von Müll aufnimmt.

[Müllart: String]

#### Fahrzeugdepot

Das Fahrzeugdepot ist der Ort, an dem alle Müllwagen stationiert sind. Somit ist es der Start- und Endpunkt jeder Route.

#### Straßenabschnitt

Ein Straßenabschnitt stellt einen bestimmten Abschnitt einer Straße aus dem aktuellen Kartenaus­schnitt dar. Für jeden Straßenabschnitt fällt eine gewisse Menge verschiedenen Mülls an, der entsorgt werden muss. Zudem ist für jeden Straßenabschnitt festgelegt, ob er gesperrt ist. Das heißt, ob hier ein Fahrzeug vorbeifahren kann. Ein Straßenabschnitt ist nicht erreichbar, falls Sperrungen keine Verbindung vom Fahrzeugdepot zu diesem Straßenabschnitt ermöglichen.

[Müllart: String, gesperrt: Boolean, erreichbar: Boolean, Müllmenge: Integer]

#### Knoten

Ein Anfahrtspunkt ist ein Knoten im Graphen, der den Kartenausschnitt repräsentiert. Bei einem Anfahrtspunkt handelt es sich entweder um eine Deponie, ein Fahrzeugdepot oder einen Straßenabschnitt.

[Tags: String-Array, Breitengrad: Double, Längengrad: Double, ID: long]

#### Plan

Ein Plan bezeichnet einen Müllentleerungsplan für einen bestimmten Kartenausschnitt für den Zeitraum von einer Woche. Da das System im Laufe der Zeit mehrere Müllentleerungspläne generiert, wird zu jedem Plan der Pfad und Dateiname des Kartenausschnitts sowie ein Beschreibungstext gespeichert.

[Kartenausschnitt: String, Beschreibung: String]

#### Route

Eine Route ist die Strecke, die von einem bestimmten Müllwagen an einem bestimmten Wochentag gemäß dem erstellten Müllentleerungsplan zurückgelegt wird.

[Wochentag: Byte]

#### Fahrzeug

Ein Fahrzeug bezeichnet einen beliebigen Müllwagen, der im System zur Umsetzung des Müllentleerungsplans zur Verfügung steht.

[Müllart: String, Fassungsvermögen: Integer, Team: String, verfuegbar: Boolean]

### Technische Komponenten



#### GUI <<Software>> (realisiert P-5)

Die GUI ist die Schnittstelle zwischen der externen Hardware und der Kartendarstellung, der Fahrzeugverwaltung und der Hindernisberechnung des Systems. Sie ist daher zuständig für die visuelle Anzeige der Daten aus diesen Modulen und zur Parameterübergabe an diese Module.

#### OSM-Parser <<Software>> (realisiert P-1)

Der OSM-Parser erhält einen beliebigen OpenStreetMap-Kartenausschnitt in Form einer XML-Datei. Aus dieser XML-Datei erzeugt er eine entsprechende Graphenstruktur, die den Kartenausschnitt der XML-Datei repräsentiert und in der bereits die entsprechenden Müllmengen den Knoten zugewiesen wurden.

#### Routenplanung <<Software>> (realisiert P-4)

Die Routenplanung generiert anhand des Graphen, der bereits von der Hindernisberechnung analysiert und ggf. modifiziert wurde, und der verfügbaren Fahrzeuge eine Menge von Routen, die zusammen einen Wochenplan zur Entsorgung aller Müllsorten im aktuellen Kartenausschnitt ergeben.

#### Kartendarstellung <<Software>> (realisiert P-5)

Die Kartendarstellung bereitet den aktuellen Graphen und den erstellten Müllentleerungsplan (falls vorhanden) gemäß der Parameter der GUI auf und liefert den aufbereiteten Plan an die GUI.

#### Fahrzeugverwaltung <<Software>> (realisiert P-2)

Die Fahrzeugverwaltung ist für das Erstellen und Löschen von Müllwagen zuständig sowie für alle Änderungen an vorhandenen Müllwagen (auch Verfügbarkeit).

#### Hindernisberechnung <<Software>> (realisiert P-3)

Die Hindernisberechnung analysiert, ob durch die Platzierung von Baustellen im Kartenausschnitt bestimmte Straßenabschnitte unerreichbar sind und verlagert in einem solchen Fall den Müll der unerreichbaren Abschnitte auf den jeweils nächsten erreichbaren Straßenabschnitt.

#### Anzeigegerät <<Hardware>> (realisiert Terminator „Benutzer“)

Das Anzeigegerät befindet sich beim Benutzer und dient der Anzeige der von der GUI des Systems übermittelten Daten.

#### Eingabegerät <<Hardware>> (realisiert Terminator „Benutzer“)

Das Eingabegerät befindet sich beim Benutzer und dient der Eingabe von Daten, die von der GUI des Systems entgegengenommen werden können.

# Testartefakte

## Modultest

### Testspezifikation

#### Modultestfall 1: OSMParser

|  |  |
| --- | --- |
| Testziel | Korrektes erkennen von gesperrten Kanten |
| Sourcecode | /testSuite/ParserTest.java |
| Vorbedingung | Graph Object != null, mehr als 0 Knoten |
| Nachbedingung | Graph Object != null, mehr als 0 Knoten |
| Bestehens Kriterien | Korrekte Anzahl an gesperrten Kanten |

#### Modultestfall 2: Kartendarstellung (Graph)

|  |  |
| --- | --- |
| Testziel | Knoten über (lat,lon)-Koordinaten finden |
| Sourcecode | /testSuite/GraphTest.java |
| Vorbedingung | Graph Object != null, mehr als 0 Knoten |
| Nachbedingung | Referenz auf einen Knoten |
| Bestehens Kriterien | Ref auf Knoten, welcher am nächsten zu den gegeben Koordinaten ist |

### Testergebnisse

#### Testprotokoll Modultestfall 1: OSM-Parser (1. Testdurchführung)

|  |  |
| --- | --- |
| Testziel | Korrektes erkennen von gesperrten Kanten |
| Sourcecode | /testSuite/ParserTest.java |
| Vorbedingung | Graph Object != null, mehr als 0 Knoten |
| Nachbedingung | Graph Object != null, mehr als 0 Knoten |
| Bestehens Kriterien | Korrekte Anzahl an gesperrten Kanten |
| Datum | 11.6.2013 |
| Tester | MK CS |
| Version der Software | SEPTrash v0.01 |
| Testtreiber | ParserTest.java (JUnit) |
| Testsystem & -umgebung | AMD X4 965, 16 GB RAM, MS Win7 Pro 64bit, Java 1.7 |
| Testurteil | Test bestanden |

#### Testprotokoll Modultestfall 1: OSM-Parser (2. Testdurchführung)

|  |  |
| --- | --- |
| Testziel | Korrektes erkennen von gesperrten Kanten |
| Sourcecode | /testSuite/ParserTest.java |
| Vorbedingung | Graph Object != null, mehr als 0 Knoten |
| Nachbedingung | Graph Object != null, mehr als 0 Knoten |
| Bestehens Kriterien | Korrekte Anzahl an gesperrten Kanten |
| Datum | 11.6.2013 |
| Tester | MK CS |
| Version der Software | SEPTrash v0.10 |
| Testtreiber | ParserTest.java (JUnit) |
| Testsystem & -umgebung | AMD X4 965, 16 GB RAM, MS Win7 Pro 64bit, Java 1.7 |
| Testurteil | Test bestanden |

#### Testprotokoll Modultestfall: Kartendarstellung (Graph) (1. Testdurchführung)

|  |  |
| --- | --- |
| Testziel | Knoten über (lat,lon)-Koordinaten finden |
| Sourcecode | /testSuite/GraphTest.java |
| Vorbedingung | Karten Object != null und mehr als 0 Knoten |
| Nachbedingung | Referenz auf einen Knoten |
| Bestehens Kriterien | Referenz auf den Knoten, welcher am nächsten zu den gegeben Koordinaten ist |
| Datum | 11.6.2013 |
| Tester | MK |
| Version der Software | SEPTrash v0.10 |
| Testtreiber | GraphTest.java (JUnit) |
| Testsystem & -umgebung | AMD X4 965, 16 GB RAM, MS Win7 Pro 64bit, Java 1.7 |
| Testurteil | Test nicht bestanden |

#### Testprotokoll Modultestfall: Kartendarstellung (Graph) (2. Testdurchführung)

|  |  |
| --- | --- |
| Testziel | Knoten über (lat,lon)-Koordinaten finden |
| Sourcecode | /testSuite/GraphTest.java |
| Vorbedingung | Karten Object != null und mehr als 0 Knoten |
| Nachbedingung | Referenz auf einen Knoten |
| Bestehens Kriterien | Referenz auf den Knoten, welcher am nächste zu den gegeben Koordinaten ist |
| Datum | 15.6.2013 |
| Tester | MK |
| Version der Software | SEPTrash v0.15 |
| Testtreiber | GraphTest.java (JUnit) |
| Testsystem & -umgebung | AMD X4 965, 16 GB RAM, MS Win7 Pro 64bit, Java 1.7 |
| Testurteil | Test bestanden |

## Systemtest

### Testspezifikation

#### Systemtestfall 1: Dateneingabe

|  |  |  |
| --- | --- | --- |
| **Szenario** | Szenario 1: Dateneingabe | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** |
| **1** | Der Benutzer importiert die Karte. | Das System lädt die OSM-Datei und berechnet die Darstellung. |
| **2** | Der User legt die Positionen des Fahrzeugdepots an. | Das System setzt die Eingabe grafisch um. |
| **3** | Der User legt den Müllentleerungsort „Plastik“ an. | Das System setzt die Eingabe grafisch um und speichert sie. |
| **4** | Der User legt den Müllentleerungsort „Papier“ an. | Das System setzt die Eingabe grafisch um und speichert sie. |
| **5** | Der User legt den Müllentleerungsort „Biomüll“ an. | Das System setzt die Eingabe grafisch um und speichert sie. |
| **6** | Der User legt den Müllentleerungsort „Restmüll“ an. | Das System setzt die Eingabe grafisch um und speichert sie. |
| **7** | Der User legt ein Fahrzeug an. | Das System speichert das neue Fahrzeug. |

#### Systemtestfall 2: Manuelle Änderungen

|  |  |  |
| --- | --- | --- |
| **Szenario** | Szenario 2: Manuelle Änderung | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** |
| **1** | Der User fügt in der Fahrzeugverwaltung ein neues Fahrzeug hinzu. | Das System zeigt das neue Fahrzeug an. |
| **2** | Der User ändert bei Fahrzeug 1 die Kapazität. | Das System zeigt das geänderte Fahrzeug 1 an. |
| **3** | Der User löscht das Fahrzeug 2. | Das System zeigt das Fahrzeug 2 nicht mehr an. |
| **4** | Der User deaktiviert Fahrzeug 3 und 4. | Das System stellt die Änderung visuell dar. |
| **5** | Der User schließt die Fahrzeugverwaltung. | Das System speichert alle Änderungen. |
| **6** |  | Das System berechnet die Route neu und passt die Ausgabe der Routen an. |

### Testergebnisse

#### Testprotokoll Systemtestfall 1 (1. Testdurchführung)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datum | | 2.7.2013 | | |
| Tester | | Vanessa Vogt | | |
| Version der Software | | 0.15 | | |
| Szenario | | Szenario 1: Dateneingabe | | |
| Schritt | Aktion (User) | Erwartete Reaktion (System) | Tatsächliche Reaktion (System) | **√ / X** |
| 1 | Der Benutzer importiert die Karte. | Das System lädt die OSM-Datei und berechnet die Darstellung. | Das System lädt die OSM-Datei und berechnet die Darstellung. | **√** |
| 2 | Der User legt die Positionen des Fahrzeugdepots an. | Das System setzt die Eingabe grafisch um. | Das Fahrzeugdepot wird angelegt, aber nicht angezeigt. | **X** |
| 3 | Der User legt den Müllentleerungsort „Plastik“ an. | Das System setzt die Eingabe grafisch um und speichert sie. | Das Fahrzeugdepot wird angelegt, aber nicht angezeigt. | **X** |
| 4 | Der User legt den Müllentleerungsort „Papier“ an. | Das System setzt die Eingabe grafisch um und speichert sie. | Das Fahrzeugdepot wird angelegt, aber nicht angezeigt. | **X** |
| 5 | Der User legt den Müllentleerungsort „Biomüll“ an. | Das System setzt die Eingabe grafisch um und speichert sie. | Das Fahrzeugdepot wird angelegt, aber nicht angezeigt. | **X** |
| 6 | Der User legt den Müllentleerungsort „Restmüll“ an. | Das System setzt die Eingabe grafisch um und speichert sie. | Das Fahrzeugdepot wird angelegt, aber nicht angezeigt. | **X** |
| 7 | Der User legt ein Fahrzeug an. | Das System erstellt und speichert das neue Fahrzeug. | Das System erstellt und speichert das neue Fahrzeug für die Dauer der Anwendung. | **√** |
| Testurteil | | Nicht bestanden. | | |

#### Testprotokoll Systemtestfall 1 (2. Testdurchführung)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datum | | 5.7.2013 | | |
| Tester | | Vanessa Vogt | | |
| Version der Software | | 0.89 | | |
| Szenario | | Szenario 1: Dateneingabe | | |
| Schritt | Aktion (User) | Erwartete Reaktion (System) | Tatsächliche Reaktion (System) | **√ / X** |
| 1 | Der Benutzer importiert die Karte. | Das System lädt die OSM-Datei und berechnet die Darstellung. | Das System lädt die OSM-Datei und berechnet die Darstellung. | **√** |
| 2 | Der User legt die Positionen des Fahrzeugdepots an. | Das System setzt die Eingabe grafisch um. | Das System stellt das Fahrzeugdepot grafisch dar. | **√** |
| 3 | Der User legt den Müllentleerungsort „Plastik“ an. | Das System setzt die Eingabe grafisch um und speichert sie. | Das System stellt den Müllentleerungs-ort „Plastik“ grafisch dar. | **√** |
| 4 | Der User legt den Müllentleerungsort „Papier“ an. | Das System setzt die Eingabe grafisch um und speichert sie. | Das System stellt den Müllentleerungs-ort „Papier“ grafisch dar. | **√** |
| 5 | Der User legt den Müllentleerungsort „Biomüll“ an. | Das System setzt die Eingabe grafisch um und speichert sie. | Das System stellt den Müllentleerungs-ort „Biomüll“ grafisch dar. | **√** |
| 6 | Der User legt den Müllentleerungsort „Restmüll“ an. | Das System setzt die Eingabe grafisch um und speichert sie. | Das System stellt den Müllentleerungs-ort „Restmüll“ grafisch dar. | **√** |
| 7 | Der User legt ein Fahrzeug an. | Das System erstellt und speichert das neue Fahrzeug. | Das System erstellt und speichert das neue Fahrzeug für die Dauer der Anwendung. | **X / √** |
| Testurteil | | bestanden | | |

#### Testprotokoll Systemtestfall 2 (1. Testdurchführung)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datum | | 2.7.2013 | | |
| Tester | | Vanessa Vogt | | |
| Version der Software | | 0.15 | | |
| Szenario | | Szenario 2: Manuelle Änderung | | |
| Schritt | Aktion (User) | Erwartete Reaktion (System) | Tatsächliche Reaktion (System) | **√ / X** |
| 1 | Der User fügt in der Fahrzeugverwaltung ein neues Fahrzeug hinzu. | Das System zeigt das neue Fahrzeug an. | Das System zeigt das neue Fahrzeug an. | **√** |
| 2 | Der User ändert bei Fahrzeug 1 die Kapazität. | Das System zeigt das geänderte Fahrzeug 1 an. | Das System zeigt das geänderte Fahrzeug 1 an. | **√** |
| 3 | Der User löscht das Fahrzeug 2. | Das System zeigt das Fahrzeug 2 nicht mehr an. | Das System zeigt das Fahrzeug 2 nicht mehr an. | **√** |
| 4 | Der User deaktiviert Fahrzeug 3 und 4. | Das System stellt die Änderung visuell dar. | Das System stellt die Änderung visuell dar. | **√** |
| 5 | Der User schließt die Fahrzeugverwaltung. | Das System speichert alle Änderungen. | Das System speichert die Änderungen nur, solange die Anwendung geöffnet ist. Bei erneutem Öffnen sind (Karte und) Fahrzeuge nicht gespeichert. | **X** |
| 6 | Der User klickt die Routenberechnung an. | Das System berechnet die Route neu und passt die Ausgabe der Routen an. | Das System gibt keine Routen aus. | **X** |
| Testurteil | | Nicht bestanden. | | |

#### Testprotokoll Systemtestfall 2 (2. Testdurchführung)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datum | | 5.7.2013 | | |
| Tester | | Vanessa Vogt | | |
| Version der Software | | 0.89 | | |
| Szenario | | Szenario 2: Manuelle Änderung | | |
| Schritt | Aktion (User) | Erwartete Reaktion (System) | Tatsächliche Reaktion (System) | **√ / X** |
| 1 | Der User fügt in der Fahrzeugverwaltung ein neues Fahrzeug hinzu. | Das System zeigt das neue Fahrzeug an. | Das System zeigt das neue Fahrzeug an. | **√** |
| 2 | Der User ändert bei Fahrzeug 1 die Kapazität. | Das System zeigt das geänderte Fahrzeug 1 an. | Das System zeigt das geänderte Fahrzeug 1 an. | **√** |
| 3 | Der User löscht das Fahrzeug 2. | Das System zeigt das Fahrzeug 2 nicht mehr an. | Das System zeigt das Fahrzeug 2 nicht mehr an. | **√** |
| 4 | Der User deaktiviert Fahrzeug 3 und 4. | Das System stellt die Änderung visuell dar. | Das System stellt die Änderung visuell dar. | **√** |
| 5 | Der User schließt die Fahrzeugverwaltung. | Das System speichert alle Änderungen. | Das System speichert die Änderungen nur, solange die Anwendung geöffnet ist. Bei erneutem Öffnen sind (Karte und) Fahrzeuge nicht gespeichert. | **X** |
| 6 | Der User klickt die Routenberechnung an. | Das System berechnet die Route neu und passt die Ausgabe der Routen an. | Das System berechnet die Routen und zeigt diese an. | **√** |
| Testurteil | | Nicht bestanden. | | |

#### Testprotokoll Systemtestfall 2 (3. Testdurchführung)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datum | | 8.7.2013 | | |
| Tester | | Vanessa Vogt | | |
| Version der Software | | 0.9 | | |
| Szenario | | Szenario 2: Manuelle Änderung | | |
| Schritt | Aktion (User) | Erwartete Reaktion (System) | Tatsächliche Reaktion (System) | **√ / X** |
| 1 | Der User fügt in der Fahrzeugverwaltung ein neues Fahrzeug hinzu. | Das System zeigt das neue Fahrzeug an. | Das System zeigt das neue Fahrzeug an. | **√** |
| 2 | Der User ändert bei Fahrzeug 1 die Kapazität. | Das System zeigt das geänderte Fahrzeug 1 an. | Das System zeigt das geänderte Fahrzeug 1 an. | **√** |
| 3 | Der User löscht das Fahrzeug 2. | Das System zeigt das Fahrzeug 2 nicht mehr an. | Das System zeigt das Fahrzeug 2 nicht mehr an. | **√** |
| 4 | Der User deaktiviert Fahrzeug 3 und 4. | Das System stellt die Änderung visuell dar. | Das System stellt die Änderung visuell dar. | **√** |
| 5 | Der User schließt die Fahrzeugverwaltung. | Das System speichert alle Änderungen. | Das System speichert alle Änderungen. | **√** |
| 6 | Der User klickt die Routenberechnung an. | Das System berechnet die Route neu und passt die Ausgabe der Routen an. | Das System berechnet die Routen und zeigt diese an. | **√** |
| Testurteil | | bestanden | | |

# Issue-Liste

In diesem Abschnitt befindet sich die finale Issue-Liste des Projektes als eingebettete Excel-Datei.



# Implementierungs-protokoll

|  |  |
| --- | --- |
| ***Package.Klasse:*** | ***Erstellt von:*** |
| **allgemein.Konstanten** | Mike van Afferden |
| **Fahrzeugverwaltung.Fahrzeug** | Mike van Afferden |
| **Fahrzeugverwaltung.FahrzeugTableListener** | Mike van Afferden |
| **Fahrzeugverwaltung.FahrzeugTableModel** | Mike van Afferden |
| **Fahrzeugverwaltung.FahrzeugVerwaltung** | Mike van Afferden |
| **Graph.Graph** | Michael Krane |
| **Graph.Kante** | Michael Krane |
| **Graph.Knoten** | Michael Krane |
| **Graph.Routenabschnitt** | Mike van Afferden |
| **GUI.Anfahrtspunkte** | Nils Dechant |
| **GUI.FahrzeugPanel** | Mike van Afferden / Christian Schmidt |
| **GUI.mainWindow** | Christian Schmidt |
| **GUI.listener** | Christian Schmidt |
| **GUI.listenerAnfahrt** | Christian Schmidt |
| **GUI.listenerRoute** | Christian Schmidt |
| **Hindernisberechnung.hindernisberechnung** | Nils Dechant |
| **Kartendarstellung.SEPSlippyMap** | Jonathan Liebers |
| **org.openstreetmap.gui.jmapviewer.\*** | OpenSource, integriert/editiert von J. Liebers |
| **OSMparser.OSMHandler** | Michael Krane |
| **OSMparser.OSMparser** | Michael Krane |
| **OSMparser.Way** | Michael Krane |
| **Routenberechnung.CPP** | “A Java library of graph algorithms and optimization”  S. 151 bis 173 ; Anpassung Michael Krane |
| **routenberechnung.Dijkstra** | vogella.com/articles/JavaAlgorithmsDijkstra/  article.html ; Anpassung Michael Krane |
| **routenberechnung.routenberechnung** | Nils Dechant |
| **testSuite.AllTests** | Michael Krane |
| **testSuite.GraphTests** | Michael Krane |
| **testSuite.ParserTests** | Michael Krane |
| **Testdurchläufe** | Michael Krane |

1. vgl. B. Kovitz: Practical Software Requirements: A Manual of Content & Style; Manning 1998 [↑](#footnote-ref-1)