Technische Universität Braunschweig

NeXT Generation Transport Tycoon

Team 0

Software-Entwicklungspraktikum (SEP)
Sommersemester 2013

Testprotokolle

Auftraggeber:
Technische Universität Braunschweig
Institut für Programmierung und Reaktive Systeme
Prof. Dr. Ursula Goltz
Mühlenpfordtstr. 23
38106 Braunschweig

Betreuer: Benjamin Mensing

Auftragnehmer:

Name	E-Mail-Adresse
Dennis Stelter	d.stelter@tu-bs.de
Henrik Lange	henrik.lange@tu-bs.de
Jochen Steiner	jochen.steiner@tu-bs.de
Markus-Björn Meißner	m-b.meissner@tu-bs.de
Patricia-Tatjana Kasulke	p.kasulke@tu-bs.de
Tessa Fabian	tessa.fabian@tu-bs.de

Braunschweig, 10. Juli 2013

Inhaltsverzeichnis

1 Testdurchführung (2013-07-03)

In diesem Abschnitt werden die einzelnen Testfälle beschrieben und deren Durchführung protokolliert.

Zunächst wird eine Aufstellung der durchgeführten Testfälle, der abgedeckten Funktionen, Komponenten, Klassen und Methoden sowie der beteiligten Test gegeben.

Ausgeführte Testfälle: $\langle T100 \rangle$, $\langle T200 \rangle$, $\langle T300 \rangle$, $\langle T400 \rangle$, $\langle T500 \rangle$, $\langle T600 \rangle$, $\langle T700 \rangle$, $\langle T800 \rangle$, $\langle T900 \rangle$, $\langle T1000 \rangle$, $\langle T1100 \rangle$, $\langle T1200 \rangle$, $\langle T1300 \rangle$, $\langle T1400 \rangle$, $\langle T1500 \rangle$, $\langle T1600 \rangle$, $\langle T1700 \rangle$, $\langle T1800 \rangle$, $\langle T1900 \rangle$, $\langle T2000 \rangle$, $\langle T2100 \rangle$, $\langle T2200 \rangle$, $\langle T2300 \rangle$, $\langle T2400 \rangle$, $\langle T2500 \rangle$, $\langle T2600 \rangle$, $\langle T2700 \rangle$, $\langle T2800 \rangle$, $\langle T2900 \rangle$, $\langle T3000 \rangle$, $\langle T3100 \rangle$, $\langle T3200 \rangle$, $\langle T3300 \rangle$
Beteiligte Tester: Henrik Lange, Dennis Stelter, Markus-Björn Meißner, Tessa Fabian, Patricia-Tajana Kasulke, Jochen Steiner
Abgedeckte Funktionen: $\langle F10 \rangle$, $\langle F20 \rangle$, $\langle F30 \rangle$, $\langle F40 \rangle$, $\langle F50 \rangle$, $\langle F60 \rangle$, $\langle F70 \rangle$, $\langle F80 \rangle$, $\langle F90 \rangle$, $\langle F100 \rangle$, $\langle F110 \rangle$, $\langle F120 \rangle$, $\langle F130 \rangle$, $\langle F140 \rangle$

1.1 Testumgebung

Die Testfälle wurden unter Windows 7 durchgeführt. Die Systemumgebung ist deutsch. Die Testfälle werden in Eclipse ausgeführt.

1.2 Testprotokoll

Im folgenden Abschnitt werden alle Testfälle detailliert in Tabellenform dokumentiert.

1.2.1 Unit-Tests

Testfall	$\langle T2900 \rangle$: Öffnen und Schließen des Hilfs-Frames, Industriestatistik-Frames und Gewinnstatistik-Frames
Tester	Tessa Fabian
Eingaben	Keine Eingaben nötig

Soll-Reaktion	Nach Drücken der Taste „F1“ oder Auswählen des Menüpunktes „Hilfe“ sollte sich das Frame öffnen und nach einem Klick auf das rote Kreuz sollte sich das Frame schließen. Nach Betätigen des Knopfes „Industriedetails“ öffnet sich ein Frame, dass Industriedetails enthält. Dieses lässt sich durch den Knopf „Industriedetails schließen“ schließen. Nach Betätigen des Knopfes „Gewinnstatistik“ öffnet sich ein Frame, welches Details zum Gewinn enthält. Das Frame lässt sich durch den Knopf „Gewinnstatistik schließen“ schließen.
Ist-Reaktion	Die Soll-Reaktion ist eingetreten
Ergebnis	Der Test konnte erfolgreich durchgeführt werden
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T2900 \rangle$: Öffnen und Schließen des Info-Fensters
Tester	Tessa Fabian
Eingaben	Keine Eingaben nötig
Soll-Reaktion	Nach dem Auswählen des Menüpunktes „Info“soll sich ein Fenster mit Informationen über das Produkt öffnen
Ist-Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich

Testfall	$\langle T2900 \rangle$: Ersteigern eines Auftrages
Tester	Tessa Fabian
Eingaben	Zunächst wird ein beliebiger, laufender Auftrag aus der Tabelle „Verfügbare Aufträge“ausgewählt. Ein Gebot, hier 15.0, wird eingeben und danach mit einem Klick auf den Button „Gebot senden“das Gebot übermittelt.
Soll-Reaktion	Nach Ablauf der Auktion soll sich ein Ereignisfenster öffnen, welches das erfolgreiche Ersteigern des Auftrages vermeldet.
Ist-Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich

Testfall	$\langle T2900 \rangle$: Abruf von Informationen über einen Knoten
Tester	Tessa Fabian
Eingaben	Keine Eingaben nötig
Soll-Reaktion	Nach dem Linksklick auf einen beliebigen Knoten im Wegenetz soll sich ein Menü mit dem Unterpunkt „Info“öffnen. Durch das Zeigen mit der Maus wird sich im Idealfall ein Pop-up mit Informationen zum gewählten Knoten öffnen.
Ist-Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich

Testfall	$\langle T2900 \rangle$: Verwerfen von falschen Zahlenwerten
Tester	Tessa Fabian
Eingaben	Ein dezimaler Zahlenwert x mit Komma als Trennzeichen anstelle des erforderlichen Dezimalpunktes
Soll-Reaktion	Nachdem der Spieler einen verfügbaren Auftrag zum Abgeben eines Gebots ausgewählt hat, gibt dieser ein Zahlenwert, hier 3,0, ein und klickt auf „Gebot senden“. Die GUI sollte darauf mit einer Fehlermeldung reagieren.
Ist-Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich

Testfall	$\langle T2900 \rangle$: Schließen der Spieloberfläche
Tester	Tessa Fabian
Eingaben	Keine Eingabedaten notwendig
Soll-Reaktion	Nach einem Klick auf das rote Kreuz sollte sich die GUI für den Spieler schließen. Das ist jedoch nicht als Beenden des Spieles zu verstehen.
Ist-Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	Unit $\langle T3000 \rangle$: Laden der Karte
Tester	Patricia Kasulke
Eingaben	Auswahl der XML-Datei „*map_weighted.xml*“.
Soll - Reaktion	Nachdem die GameAdministration gestartet wurde, kann die .xml-Datei eingelesen werden. Ist sie eingelesen, wird sie grün markiert und ein „Ok“ angezeigt.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Laden der Industriedaten
Tester	Patricia Kasulke
Eingaben	Auswahl der .xml-Datei „industry.xml““.
Soll - Reaktion	Nachdem die „industry.xml“ eingelesen wurde, erscheint der Pfad neben dem Wählen-Button und wird grün markiert.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Anzeigen der Industriedaten
Tester	Patricia Kasulke
Eingaben	Auswahl der .xml-Datei „Industrie.xml“.
Soll - Reaktion	Nachdem der Pfad „industrie.xml“ eingelesen wurde, sollen die Industriedaten aus der .xml-Datei mit dem IndustryParser eingelesen und auf der Oberfläche in Tabellenform mit Industrienamen und dem jeweiligen Standort angezeigt.
Ist – Reaktion	Nachdem der Pfad „industrie.xml“ eingelesen wurde, wurden die Industriedaten aus der .xml-Datei mit dem IndustryParser eingelesen und auf der Oberfläche in Tabellenform mit Industrienamen und dem jeweiligen Standort angezeigt.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Hinzufügen der Roboter zu einem Spiel
Tester	Patricia Kasulke
Eingaben	Keine Eingabe erforderlich.
Soll - Reaktion	Verbundene Roboter werden in der „GameAdministration“ angezeigt und werden durch Klicken auf „Roboter hinzufügen“-Button dem Spiel hinzugefügt.
Ist – Reaktion	Soll-Reaktion erfüllt. Roboter „Robot“ wird in der Tabelle angezeigt und kann dem Spiel hinzugefügt werden.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Einbinden von Spielern
Tester	Patricia Kasulke
Eingaben	Name des Spielers: Benutzer1 , Klick auf Verbinden.
Soll - Reaktion	Verbindung zum Server aufbauen und in der GameAdministration als Spieler hinzufügbar sein.
Ist – Reaktion	Benutzer1 wurde in der GameAdministration hinzugefügt und kann einem Roboter zugeordnet werden.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Hinzufügen von Robotern zu einem Team
Tester	Patricia Kasulke
Eingaben	Keine Eingaben notwendig.
Soll - Reaktion	Ein eingebundener Roboter und ein eingebundener Spieler werden jeweils markiert und zum Spiel hinzugefügt. Spieler und Roboter werden als Team im Frame angezeigt.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Schließen des InitFrames
Tester	Patricia Kasulke
Eingaben	Keine Eingaben vorhanden.
Soll - Reaktion	Durch Klicken auf das Kreuz in der rechten oberen Ecke, schließt sich das Fenster.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Anzeigen der Aufträge
Tester	Patricia Kasulke
Eingaben	Klick in das Drop-Down-Menu der Aufträge und Auswahl des jeweiligen Punktes.
Soll - Reaktion	Anzeigen der jeweiligen Information über Aufträge in der Auftragstabelle.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Informations-Popup anzeigen
Tester	Patricia Kasulke
Eingaben	Klicken auf die Objekte der Karte.
Soll - Reaktion	Öffnen des Informations-Popups und Anzeigen der jeweiligen Informationen.
Ist - Reaktion	Die korrekte Anzeige der Daten an den jeweiligen Objekten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Verwalten eines Knotens
Tester	Patricia Kasulke
Eingaben	Klick auf den Knoten 10.
Soll - Reaktion	Es werden die Punkte „Info“ und „Sperren“ angezeigt. Beim Klick auf Sperren wird das Gesperrt-Symbol angezeigt und der Knoten 10 gesperrt. Außerdem kann man ihn dann wieder Freigeben.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Verwalten einer Kante.
Tester	Patricia Kasulke
Eingaben	Klick auf Kante 9,10.
Soll - Reaktion	Es werden die Punkte „Info“ und „Sperren“ angezeigt. Beim Klick auf „Sperren“ wird das Gesperrt-Symbol angezeigt und die Kante 9,10 rot markiert. Bei erneuten Klicken auf die Kante wird statt dem Punkt „Sperren“ der Punkt „Freigeben“ angezeigt und die Kante wird wieder schwarz angezeigt.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3000 \rangle$: Entfernen eines Roboters
Tester	Patricia Kasulke
Eingaben	Markieren des Roboters und Klick auf „Roboter entfernen“.
Soll - Reaktion	Meldung, dass der Roboter entfernt wird, erscheint und der Roboter wird aus der Liste der im Spiel befindlichen Roboter entfernt.
Ist – Reaktion	Meldung über das Entfernen des Roboters wird angezeigt und der Roboter wird aus der Liste gelöscht.
Ergebnis	Der Test erzeugte anfangs immer eine NullPointerException. Der NXTManager des Roboters konnte nicht gefunden werden, um dem Roboter den kill()-Befehl zuzusenden. Der Fehler lag darin, dass auf das Roboterobjekt in der GameRegistry zuerst die deactivateRobot()-Methode angewandt wurde, wodurch der NXTManager auf NULL gesetzt wurde.

Nacharbeiten	Die Reihenfolge der Befehle wurde getauscht. Nun wird der Roboter erst deaktiviert, nachdem ihm der kill()-Befehl zugesendet wurde. Der Roboter lässt sich nun wie erwartet aus dem Spiel entfernen.

Testfall	$\langle T3000 \rangle$: Beenden des ServerFrames/Spiel beenden
Tester	Patricia Kasulke
Eingaben	Klick in das Drop-Down-Menu und Auswahl des Punktes Spiel beenden.
Soll - Reaktion	Schließen des ServerFrames. Spiel wird beendet.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3100 \rangle$: Aufträge werden von der Auktionsverwaltung empfangen und als Auktion gestartet.
Tester	Markus Björn Meißner
Eingaben	Das Serverprogramm wird gestartet. Die Karten XML-Datei „map_weighted“ wird geladen. Die Industrie XML-Datei „Industriedaten_alt“ wird geladen. Ein beliebiger Roboter wird hinzugefügt. Das Spiel wird gestartet.
Soll - Reaktion	Nach Spielstart wird unter anderem automatisch der Auftrag 0 von Knoten 0 zu Knoten 1 mit der Ware „Gefluegel“ erstellt. Die Meldung der Klasse Auction „Auktion ID:0 gestartet“ signalisiert, dass der Auftrag empfangen und die Auktion gestartet wurde.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

Testfall	$\langle T3100 \rangle$: Gebote werden registriert und nach Ablauf der Auktion wird das niedrigste Gebot ermittelt.
Tester	Markus Björn Meißner
Eingaben	Die Klasse „Auction“ wird so modifiziert, dass zwei automatische Gebote abgegeben werden. Zum einen ein Gebot von 100 auf die Auktion mit der ID 0 von Roboter ID 0 und zum anderen ein Gebot von 200 von Roboter ID 1 auf die gleiche Auktion. Das Serverprogramm wird gestartet. Die Karten XML-Datei „map_weighted“ wird geladen. Die Industrie XML-Datei „Industriedaten_alt“ wird geladen. Der Roboter "Hans“ wird hinzugefügt. Der Roboter „Heinz“ wird hinzugefügt. Das Spiel wird gestartet.
Soll - Reaktion	Fortführend zur Ist - Reaktion vom ersten Test $\langle T3100 \rangle$, werden nach Starten der Auktion in der „run()“ Methode, vor Ablauf der Zeit zum Bieten, die automatischen Gebote mittels Aufrufen von „addBid()“ von Roboter 0 und Roboter 1 ausgeführt. Die Konsole meldet „Gebot für Auktion 0: 100 erhalten“ und „Gebot für Auktion 0: 200 erhalten“. Nach Ablauf der Zeit zum Bieten, wird mit der Methode „getWinnerBid()“ das niedrigste Gebot ermittelt. Die Konsole meldet „AuktionID:0 hat gewonnen: Teilnehmer 0“.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

Testfall	$\langle T3100 \rangle$: Die Auktionsergebnisse werden an alle Teilnehmer gesendet. Dem Auktionsgewinner wird der Auftrag erteilt.
Tester	Markus Björn Meißner
Eingaben	Gleiche Eingaben wie beim vorherigen Testfall von $\langle T3100 \rangle$.
Soll - Reaktion	Fortführend zur Ist - Reaktion vom vorherigen Test$\langle T3100 \rangle$ wird von der Klasse „Auction“ nach Ermitteln des Gewinners die Methode „distributeData()“ der „GameAdministration“ aufgerufen. Diese meldet mit „<Auktionsende> an Hans gesendet.“ und „<Auktionsende> an Heinz gesendet.“, dass das Auktionsergebnis an beide Teilnehmer gesendet wird. Auf der GUI(Server) wird in der Liste für die laufenden Aufträge bei „Auftrags-ID 0“ in der Spalte „Roboter“ der Name „Hans“ angezeigt. Weiterhin meldet die Konsole des Server „Auftrag erteilt an Hans“.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.

Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

Testfall	$\langle T3200 \rangle$: Spieler und Roboter werden in der GameRegistry gespeichert.
Tester	Markus Björn Meißner
Eingaben	Das Serverprogramm wird gestartet. Das Spielerprogramm wird gestartet. Ein Roboter mit Namen „Beep“ wird auf Position 1 hinzugefügt. Ein Spieler mit Namen „Mr0wn463“ verbindet sich mit dem Server. Beide Teilnehmer werden Team 1 zugeordnet. Die übrigen Eingabefelder werden mit den Standardeinstellungen übernommen. Nachdem noch eine Karte und die Industrien geladen wurden, wird das Spiel gestartet.
Soll - Reaktion	Noch vor dem Senden der Teilnehmerliste an den Spieler bzw. Spielstart werden über die Konsole sowohl die Roboterliste, als auch die Spielerliste der GameRegistry ausgegeben. Ausgabe für die Roboterliste: „[Name:Beep ID:0 TeamID:1 Location:1 MaxCap:100 MaxOrd:2 Fuel:500 BelongsTo:1 Orders:[]]“. Ausgabe für die Spielerliste „[1 Mr0wn463]“
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

Testfall	$\langle T3200 \rangle$: Nur die bei Spielstart im Initialisierungsfenster angezeigten Industrien werden in der Industrieverwaltung gespeichert.
Tester	Markus Björn Meißner
Eingaben	Das Serverprogramm wird gestartet. Eine geeignete Kartendatei wird geladen. Ein beliebiger Roboter wird hinzugefügt. Die XML-Datei „Industriedaten_alt“ wird geladen. Übersichthalber werden der Reihe nach, durch Anwählen mit der Maus und Drücken von „Entf.“, alle Einträge der Industrietabelle des Initialisierungfensters bis auf „Fleischfabrik“ entfernt. Der Spielstart Button wird betätigt.

Soll - Reaktion	Die Konsole meldet noch vor Anzeige der GUI(Server) „[Fleischfarbik Id0 Pos:0 In:[Schweine, Gefluegel] InAmount:[10.0 10.0] Out:[Fleischwaren] outAmount[1.0] false ResIn:[0.0 0.0] ResOut:[0.0]]“. Der Benutzer öffnet das „Industriedetails“ Fenster und bekommt die Fleischfabrik als einzige Industrie angezeigt. Das „GameDisplay“ zeigt nur ein einziges Industriegebäude mit Knotenposition 0.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

Testfall	$\langle T3200 \rangle$: Die Roboter und Industrien werden auf dem Graphen platziert.
Tester	Markus Björn Meißner
Eingaben	Das Serverprogramm wird gestartet. Die XML-Datei „Industriedaten_alt“ wird geladen. Alle Industrien bis auf das „Kraftwerk“ auf Position „8“ werden aus der Industrieliste des Initialisierungsfensters mittels „Entf.“ gelöscht. Der Roboter „TheModel“ wird auf Position „0“ hinzugefügt und dem Team0 zugewiesen. Das Spiel wird gestartet.
Soll - Reaktion	Die Methoden „industryToMap()“ und „robotsToMap()“ arbeiten korrekt. Die Wegenetzverwaltung meldet für den Roboter: „Set Location(KNOTEN): Roboter 0 gesetzt auf Knoten:0“. Für die Industrie: „Industrie gesetzt“. Das GameDisplay zeigt die Industrie „Kraftwerk“ auf Knoten „8“ und auf Knoten „0“ „TheModel“ an.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

Testfall	$\langle T3200 \rangle$: Den Robotern werden Kosten für das Befahren von Strecken berechnet.
Tester	Markus Björn Meißner
Eingaben	Das Serverprogramm wird gestartet. Die Karten XML-Datei „map_weighted“ wird geladen. Die Industrie XML-Datei „Industriedaten_alt“ wird geladen. Der Roboter „NXT_24“ wird auf Position „0“ hinzugefügt und zu Team 0 zugewiesen. In das MaxOrder Feld wird eine „1“ eingetragen. Das Spiel wird gestartet.

Soll - Reaktion	Der Roboter berechnet nach Erhalt der Auktion „0“ ein Gebot und sendet dieses an den Server. Das Gebot gewinnt den Auftrag „0“. Der Roboter beginnt von Knoten „0“ aus zum Startknoten „1“ des Auftrags zu fahren. Nach jedem Verlassen eines Knotens wird dem Konto des Roboters über „addCash()“ der negative Wert des Gewichtes der Kante berechnet, auf der er sich befindet. Die erste Kante nach Knoten „0“ hat die Kosten 91. Das Konto des Roboters in der Robotertabelle zeigt einen Stand von „-91“ an. Die Gewichte der zu fahrenden Kanten entlang der Strecke betragen insgesamt 300. Nach Erreichen des Startknotens des Auftrags, hat der Roboter einen Kontostand von „-300“.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Das anfängliche Testergebnis war zwar, dass den Robotern beim Befahren von Kanten Kosten entstanden, nur waren diese immer doppelt so hoch wie erwartet. Es fiel auf, dass die Roboter ihre Knotenfreigabe immer zweimal sendeten. Da mit der Knotenfreigabe immer automatisch die Kantenbelegung erfolgt, wurde den Robotern immer der doppelte Wert pro Kante berechnet.
Nacharbeiten	Die Roboter senden ihr Knotenfeigaben nun nur noch einmal. Das Gewicht einer Kante wird den Robotern nun korrekt in Rechnung gestellt

Testfall	$\langle T3300 \rangle$: Nicht zusammenhängenden Graphen einbinden.
Tester	Dennis Stelter
Eingaben	In einer geeigneten Testumgebung (getrennt von anderen Komponenten) wird eine Instanz der Klasse „MapAdministration“ initialisiert, dieser wird ein Graph-Objekt, welches mindestens einen Knoten, der nicht mit einem anderen Knoten verbunden ist, enthält, übergeben. Dazu könnten zum Beispiel die in der Klasse vorhandenen Methoden dummyedges und dummynode benutzt werden, die mit dem Knoten „n8“ einen für diesen Testfall geeigneten Fehler liefern. Anschließend wird die Methode makeCoordinates aufgerufen.
Soll - Reaktion	Es wird eine Fehlermeldung in der Konsole ausgegeben, da der Graph nicht zusammenhängend ist („Graph nicht zulässig da nicht zusammenhängend!“).
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.

Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3300 \rangle$: Korrekten Graphen einbinden.
Tester	Dennis Stelter
Eingaben	In einer geeigneten Testumgebung (getrennt von anderen Komponenten) wird eine Instanz der Klasse „MapAdministration“ initialisiert, dieser wird ein Graph-Objekt übergeben, welches einen zusammenhängenden Graphen mit mehr als einem Knoten enthält (z.B. die durch den „GraphParser“ aus Komponente $\langle C50 \rangle$ importierte Datei „*map_weighted.xml*“). Anschließend wird die Methode „makeCoordinates“ aufgerufen.
Soll - Reaktion	In der Konsole wird mit der Meldung „Alle Knoten markiert“ die erfolgreiche Erstellung der Koordinaten für die Verwaltung bestätigt.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3300 \rangle$: Verwaltungsgraphen neu anordnen.
Tester	Dennis Stelter
Eingaben	Nach der korrekten Einbindung des Graphen wird mit dem dort erstellten Graphen die Methode „alignGraphList“ aufgerufen. Anschließend wird der zurückgegebene Graph über die Konsole ausgegeben (eine passende toString-Methode ist vorhanden).
Soll - Reaktion	Der in der Konsole ausgegebene Graph enthält keine negativen Koordinaten mehr.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3300 \rangle$: Industrie auf der Karte positionieren.
Tester	Dennis Stelter
Eingaben	Nach der erfolgreichen Einbindung und Anordnung des Graphen wird die Methode „setLocation“ mit einem passenden Industrieobjekt (z.B. einem, welches über den IndustryParser der Komponente $\langle C50 \rangle$ aus der Datei „Industriedaten_alt.xml“ ausgelesen wurde) übergeben.

Soll - Reaktion	In der Konsole wird die Meldung „Industrie gesetzt“ ausgegeben.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3300 \rangle$: Roboter auf der Karte positionieren.
Tester	Dennis Stelter
Eingaben	Nach der erfolgreichen Einbindung und Anordnung des Graphen wird die Methode „setLocation“ mit einem passenden Roboterobjekt (z.B. mit Name: NXT_19 und Position: 0) übergeben.
Soll - Reaktion	In der Konsole wird eine Erfolgsmeldung ausgegeben (z.B: „Set Location(KNOTEN): Roboter 0 gesetzt auf Knoten 0“).
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T3400 \rangle$: Die Waren am Warenausgang der Industrien werden zum Inhalt von Aufträgen, falls eine Industrie existiert, die diese Waren verarbeiten kann.
Tester	Markus Björn Meißner
Eingaben	Das Serverprogramm wird gestartet. Die Karten XML-Datei „map_weighted“ wird geladen. Die Industrie XML-Datei „Industriedaten_alt“ wird geladen. Ein beliebiger Roboter wird hinzugefügt. Im Initialisierungsfenster werden alle Industrien bis auf „Fleischfabrik“ und „Bauernhof“ entfernt. Das Programm wird gestartet.
Soll - Reaktion	Nachfolgend zur Soll - Reaktion von Testfall $\langle T3400 \rangle$ wird nach der Produktion von Waren die Methode searchForOrder() aufgerufen. Die Konsole zeigt hierbei an: „Auftrag erstellt Id:0 Start:1 End:0 PPU:100.0 PId-1 t:300“. Es wurde demnach ein Auftrag mit dem Ziel erstellt, Ware von dem Bauernhof zur Fleischfabrik zu befördern (vgl. „Out“ und „In“ der Konsolenausgaben bei Testfall T100-10). Die GameAdministraton meldet kurz darauf: „GameAdministration: Neue Order erhalten OrderID:0 Startknoten:1 Zielknoten:0 Warenwert:1100.0“.
Ist -Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

Testfall	$\langle T3400 \rangle$: Waren, die für eine Auktion reserviert wurden, werden an den Warenausgang der Industrie zurückgegeben, falls die Auktion nicht erfolgreich war.
Tester	Markus Björn Meißner
Eingaben	Das Serverprogramm wird gestartet. Die Karten XML-Datei „map_weighted“ wird geladen. Die Industrie XML-Datei „Industriedaten_alt“ wird geladen. Ein beliebiger Roboter wird hinzugefügt. Im Initialisierungsfenster werden alle Industrien bis auf „Fleischfabrik“ und „Bauernhof“ entfernt. Im „MaxOrder“ Feld wird der Wert „0“ eingegeben, damit der Roboter nicht in der Lage ist, einen Auftrag zu ersteigern. Das Programm wird gestartet. Eine Zeit von 30 Sekunden wird abgewartet.
Soll - Reaktion	Nachdem die Auktionen gestartet worden sind, läuft nach 30 Sekunden die Zeit für Gebote ab. Die Klasse Auction vermerkt mit der Nachricht „Auktion 0 nicht erfolgreich.“, dass für Auktion 0 keine Gebote empfangen wurden . Die Methode returnrderToIndustry() meldet einmal vor Ausführung „Bauernhof Id1 Pos:1 In:[Getreide, Wasser] InAmount:[0.0 0.0] Out:[Gefluegel] outAmount[0.0] false ResIn:[0.0 0.0] ResOut:[11.0]“ und nach Ausführung d.h. Rückgabe der Waren an den Warenausgang „Bauernhof Id1 Pos:1 In:[Getreide, Wasser] InAmount:[0.0 0.0] Out:[Gefluegel] outAmount[11.0] false ResIn:[0.0 0.0] ResOut:[0.0]“. Die reservierte Menge „ResOut“ wurde wieder zur Ware am Warenausgang „outAmount".
Ist -Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

1.2.2 Integrationstest

Testfall	$\langle T1900 \rangle$ | C60 -> C30 | 1
Tester	Dennis Stelter
Eingaben	Eine Instanz der Klasse NXTManager wurde mit dem Namen „NXT_19“ erstellt und die Initialisierungsmethode mit geeigneten Daten (dem Namen und Position 0) für den Roboter aufrufen. Der dazugehörige Roboter wurde vor dem Aufruf der Methode angeschaltet, mit dem Computer per Bluetooth gekoppelt und das Empfangsmodul ($\langle C30 \rangle$) wurde gestartet.
Soll - Reaktion	Der Roboter zeigt auf seinem Display durch den Hinweistext „Robot Data received“ an, dass er korrekt initialisiert wurde.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T1900 \rangle$ | C60 -> C30 | 2
Tester	Dennis Stelter
Eingaben	Nach der Ausführung der Eingaben von Punkt 1 wurde die Methode „sendGraph“ aufgerufen, hier wurde die durch die Klasse „GraphParser“ aus Kompontente $\langle C50 \rangle$ importierte Datei „map_weighted.xml“ aus dem config-Ordner verwendet.
Soll - Reaktion	Der Roboter zeigt auf seinem Display zuerst die übertragene Soll-Größe 20 und nach der Übertragung die tatsächliche Größe des Graphen an, die mit der Soll-Größe übereinstimmt.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨T1900⟩ | C60 -> C30 | 3
Tester	Dennis Stelter
Eingaben	Nach der Initialisierung wurde die Methode „sendOrder“ mit dem Auftrag Start: 0, Ziel: 1, Preis: 50, Ware: (Holz, 10, 90) aufgerufen. Da dies der einzige Auftrag war, bekam er die ID 0.
Soll - Reaktion	Der Roboter empfängt den Auftrag und bestätigt den Empfang mit der Ausgabe „Auftrag 0 empfangen“ auf seinem Display.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨T1900⟩ | C60 -> C30 | 4
Tester	Dennis Stelter
Eingaben	Nach der Ausführung der Eingaben von Punkt 3 wurde die Methode „sendAuctionResult“ mit der ID 0 und einem „true“ zur Bestätigung der erfolgreichen Auktion gesendet.
Soll - Reaktion	Der Roboter empfängt die Auftragsbestätigung und bestätigt den Empfang mit der Ausgabe „Auftragsbestätigung zu Auftrag 0 empfangen“ auf seinem Display.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨T1900⟩ | C60 -> C30 | 5
Tester	Dennis Stelter
Eingaben	Die Methode „approveEdge“ wird aufgerufen, die Knoten 3 und 4 wurden gefolgt von einem „true“ für die Bestätigung der Freigabe übergeben.
Soll - Reaktion	Der Roboter empfängt die Bestätigung und zeigt auf dem Display die Meldung „Kante von Knoten 3 nach Knoten 4 frei“ an.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨*T*1900⟩ | C60 -> C30 | 6
Tester	Dennis Stelter
Eingaben	Die Methode „kill“ wurde aufgerufen.
Soll - Reaktion	Der Roboter empfängt den Befehl, stellt alle Aktionen ein und beendet das ausgeführte Programm.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨*T*1900⟩ | C60 -> C30 | 7
Tester	Dennis Stelter
Eingaben	Dem Roboter wurde mit der Methode sendRoute die Route 0 -> 3 -> 4 -> 1 übertragen.
Soll - Reaktion	Der Roboter empfängt die Route und zeigt die Knotenfolge „0, 3, 4, 1“ auf seinem Display an.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨*T*1900⟩ | C30 -> C60 | 1
Tester	Dennis Stelter
Eingaben	Der Roboter NXT_19 wurde angeschaltet, mit dem Computer gekoppelt und durch das Standardprogramm des Roboters wurde eine Instanz der Klasse BluetoothSender initialisiert. Passend dazu wurde in einer Testumgebung (die restlichen Komponenten des Servers wurden ausgeschlossen und durch eine Testklasse ersetzt) die Klasse NXTReceiver zum Empfang der Daten instanziiert.
Soll - Reaktion	Die Verbindung wird initialisiert und der Roboter bestätigt dies mit der Anzeige „connected“ auf seinem Display.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨*T*1900⟩ | C30 -> C60 | 2
Tester	Dennis Stelter
Eingaben	Die Methode „sendLocation“ wurde mit der Zahl 0 für den aktuellen Knoten des Roboters aufgerufen.
Soll - Reaktion	Der Server gibt die empfangene Position durch die Meldung „Roboter NXT_19 befindet sich an Position 0“ in der Konsole aus.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨*T*1900⟩ | C30 -> C60 | 3
Tester	Dennis Stelter
Eingaben	Die Methode „requestEdgeStatus“ wurde mit den anliegenden Knoten 3 und 4 aufgerufen.
Soll - Reaktion	In der Konsole des Servers wird die Anfrage mit den beiden übergebenen Knoten ausgegeben („Roboter NXT_19 hat die Kante zwischen 3 und 4 angefragt“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨*T*1900⟩ | C30 -> C60 | 4
Tester	Dennis Stelter
Eingaben	Die Methode „releaseEdge“ wurde mit den anliegenden Knoten 3 und 4 aufgerufen.
Soll - Reaktion	In der Konsole des Servers wird die Freigabe der Kante mit den beiden übergebenen Knoten ausgegeben („Roboter NXT_19 hat die Kante zwischen 3 und 4 wieder freigegeben“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T1900$⟩ | C30 -> C60 | 5
Tester	Dennis Stelter
Eingaben	Die Methode „releaseNode“ wurde mit dem freizugebenden Knoten 4 und dem Knoten 1 als nächsten zu besuchenden aufgerufen.
Soll - Reaktion	In der Konsole des Servers wird die Freigabe des Knotens mit den beiden übergebenen Knoten ausgegeben („Roboter NXT_19 hat den Knoten 4 freigegeben und ist auf dem Weg zu Knoten 1“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T1900$⟩ | C30 -> C60 | 6
Tester	Dennis Stelter
Eingaben	Die Methode „sendBid“ wurde mit der ID 0 und dem Gebot 300 aufgerufen.
Soll - Reaktion	In der Konsole des Servers wird das empfangene Gebot mit der zugehörigen ID ausgegeben („Roboter NXT_19 hat für den Auftrag 0 ein Gebot von 300 abgegeben“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T1900$⟩ | C30 -> C60 | 7
Tester	Dennis Stelter
Eingaben	Die Methode „sendRoute“ wurde mit der Route 0 -> 3 -> 4 -> 1 aufgerufen.
Soll - Reaktion	In der Konsole des Servers wird die empfangene Route in Form von abzufahrenden Knoten ausgegeben („Roboter NXT_19 fährt die Route 0,3,4,1 ab“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T1900$⟩ | C30 -> C60 | 8
Tester	Dennis Stelter
Eingaben	Die Methode „sendOrderPickup“ wurde mit der ID 0 aufgerufen.
Soll - Reaktion	In der Konsole des Servers wird die Aufnahme des Auftrages mit der passenden ID angezeigt („Roboter NXT_19 hat den Auftrag 0 abgeholt“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T1900$⟩ | C30 -> C60 | 9
Tester	Dennis Stelter
Eingaben	Die Methode „sendOrderDeposit“ wurde mit der ID 0 aufgerufen.
Soll - Reaktion	In der Konsole des Servers wird die Abgabe des Auftrages mit der passenden ID angezeigt („Roboter NXT_19 hat den Auftrag 0 abgeholt“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T2100$⟩ | C10 -> C30 | 1
Tester	Jochen Steiner
Eingaben	Die Methode „sendBid“ wurde mit einem Gebot aufgerufen.
Soll - Reaktion	Auf der Konsole wird das Gebot zum Auftrag wie folgt ausgegeben („Auftrags-ID: 0 mit Gebot 100 erhalten“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T2100$⟩ | C30 -> C10 | 1
Tester	Jochen Steiner
Eingaben	Die Methode „calculateQuotation“ wurde mit einem Auftragsobjekt aufgerufen.
Soll - Reaktion	Auf der Konsole wird das errechnete Gebot zum Auftrag wie folgt ausgegeben („Für die Auftrags-ID: 0 wurde das Gebot 100 errechnet“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T2100$⟩ | C10 -> C120 | 1
Tester	Jochen Steiner
Eingaben	Die Methode „getOrderList“ wurde mit einem Auftragsobjekt aufgerufen.
Soll - Reaktion	Auf dem Display, des Roboters wird die OderList wie folgt ausgegeben („OderID: 0“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T2100$⟩ | C120 -> C10 | 1
Tester	Jochen Steiner
Eingaben	Die Methode „getNextNode“ wird aufgerufen.
Soll - Reaktion	Auf dem Display, des Roboters wird der nächste Knoten wie folgt ausgegeben („Nächster Knoten : 3“).
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T2200$⟩ | C40 -> C60 | 1
Tester	Dennis Stelter
Eingaben	Der Methode „sendData“ der Klasse CommunicationClient wurde das Auftragsobjekt: Start: 0, Ziel: 1, Preis: 50, Ware: (Holz, 10, 90) übergeben.
Soll - Reaktion	In der Konsole wird vom Server der genannte Auftrag ausgegeben.

Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T2200 \rangle$ | C40 -> C60 | 2
Tester	Dennis Stelter
Eingaben	Der Methode „sendData“ der Klasse CommunicationClient wurde der String „test“ übergeben.
Soll - Reaktion	In der Konsole wird vom Server der String „test“ ausgegeben.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T2200 \rangle$ | C60 -> C40 | 1
Tester	Dennis Stelter
Eingaben	Der Methode „sendData“ der Klasse CommunicationServer wurde das Auftragsobjekt: Start: 0, Ziel: 1, Preis: 50, Ware: (Holz, 10, 90) übergeben.
Soll - Reaktion	In der Konsole wird vom Client der genannte Auftrag ausgegeben.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T2200 \rangle$ | C60 -> C40 | 2
Tester	Dennis Stelter
Eingaben	Der Methode „sendData“ der Klasse CommunicationServer wurde der String: „test“ übergeben.
Soll - Reaktion	In der Konsole wird vom Client der String „test“ ausgegeben.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T2300$⟩ | C20 -> C40 | 1
Tester	Dennis Stelter
Eingaben	Der Methode „sendData“ der Klasse CommunicationServer wurde das Auftragsobjekt: Start: 0, Ziel: 1, Preis: 50, Ware: (Holz, 10, 90) übergeben.
Soll - Reaktion	In der Konsole wird der vom Server der genannte Auftrag ausgegeben.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T2300$⟩ | C20 -> C40 | 2
Tester	Dennis Stelter
Eingaben	Der Methode „sendData“ der Klasse CommunicationClient wurde der String: „test“ übergeben.
Soll - Reaktion	In der Konsole wird vom Server der String „test“ ausgegeben.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	⟨$T2300$⟩ | C40 -> C20 | 1
Tester	Dennis Stelter
Eingaben	Der Methode „sendData“ der Klasse PlayerAdministration wurde das Auftragsobjekt: Start: 0, Ziel: 1, Preis: 50, Ware: (Holz, 10, 90) übergeben.
Soll - Reaktion	In der Konsole wird der vom Server der genannte Auftrag ausgegeben.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T2300 \rangle$ | C40 -> C20 | 2
Tester	Dennis Stelter
Eingaben	Der Methode „sendData“ der Klasse PlayerAdministration wurde der String: „test“ übergeben.
Soll - Reaktion	In der Konsole wird vom Server der String „test“ ausgegeben.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T2400 \rangle$
Tester	Dennis Stelter
Eingaben	Nach dem gestarteten Spiel wurde der RadioButton für das manuelle Erstellen von Aufträgen angewählt und in den entsprechenden Feldern der Auftrag: Start: 0, Ziel: 1, Preis: 50, Ware: (Holz, 10, 90) eingefügt, anschliessend wurde der Auftrag abgeschickt.
Soll - Reaktion	Der erstellte Auftrag wird in der Auktionsliste angezeigt.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T2500 \rangle$
Tester	Dennis Stelter
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurden geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde das Spiel gestartet mit „Spiel starten“.
Soll - Reaktion	Die Industrien werden in der Konsole ausgegeben.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T2600 \rangle$
Tester	Dennis Stelter
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde das Spiel gestartet mit „Spiel starten“.
Soll - Reaktion	Die Karte wird korrekt in der GUI(Server) angezeigt.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T2700 \rangle$
Tester	Dennis Stelter
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde das Spiel gestartet mit „Spiel starten“.
Soll - Reaktion	Die Industrien sowie die Roboter werden in der GUI(Server) korrekt angezeigt.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T2800 \rangle$
Tester	Dennis Stelter
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde das Spiel gestartet mit „Spiel starten“. Dann wurde dem Roboter einen Auftrag gesendet welcher beim Knoten 1 startet und beim Knoten 0 endet.
Soll - Reaktion	Es wird angezeigt das der Roboter die Ware an der Startindustrie aufnimmt und diese an der korrekten Industrie wieder abliefert. Danach soll der Auftrag dem Roboter angerechnet und aus der Auftragsliste gelöscht werden.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

1.2.3 Abnahmetest

Alle folgenden Test, in diesem Unterkapitel, sind dafür da, um zu testen, ob das Produkt alle Anforderungen erfüllen kann.

Testfall	$\langle T100 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Ein beliebiger Roboter wurde dem Spiel hinzugefügt. Zudem wurd die maximale Ladekapazität der Roboter auf „50“ gesetzt und der Tankstand der Roboter auf „300“. Danach wird das Spiel gestartet mit dem Button „Spiel starten“.
Soll - Reaktion	Die Serveroberfläche soll den Graphen und die Industrien korrekt anzeigen. Zudem soll in der Tabelle erkennbar sein, dass die Werte übernommen wurden.
Ist - Reaktion	Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T200 \rangle$
Tester	Henrik Lange
Eingaben	Für den Namen des Spielers wurde „Player“ eingegeben und die IP des Servers wurde freigelassen, da der Standardwert auf „localhost“ gesetzt ist.
Soll-Reaktion	Nachdem sich der Spieler mit seinem Namen und der IP des Servers am Spiel angemeldet hat, soll sich die GUI öffnen, sobald das Spiel serverseitig gestartet wurde.
Ist-Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich

Testfall	$\langle T300 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet.
Soll - Reaktion	Der Roboter zeigt auf seinem Display durch den Hinweistext „Robot Data received“ an, dass er korrekt initialisiert wurde.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test mit nur einem Roboter war erfolgreich. Wurden jedoch mehrere Roboter ins Spiel eingebunden, fiel auf, dass alle Roboter den gleichen Standort meldeten, obwohl die Daten in der Registrierung korrekt waren. Die Roboter waren immer mehrfach initialisiert worden.
Nacharbeiten	Der Aufruf, mit dem jeder Roboter einzeln initialisiert werden sollten, enthielt einen Fehler, so dass immer allen Robotenn alle Initialisierungsdaten zugesenden wurden. Der Aufruf wurde korrigiert. Jedem Roboter werden nun nurnoch einmal die entsprechenden Konfigurationsdaten zugesendet.

Testfall	$\langle T400 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurden geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter einem Team zugewiesen und das Spiel wurde gestartet.
Soll - Reaktion	Die Verbindung wird initialisiert und der Roboter bestätigt mit der Anzeige „ready“ auf seinem Display.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T500 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter einem Team zugewiesen und das Spiel wird gestartet. Dem Roboter wurde ein Auftrag mit ID 0, welcher beim Knoten 1 beginnt und beim Knoten 0 endet, vom Server zugesandt.
Soll - Reaktion	Die KI des Roboters berechnet ein Gebot zu dem erhaltenen Auftrag und sendet sein Gebot an den Server und gibt auf der Konsole „Gebot zu Auftrag 0 abgegeben.“ aus. Falls der Server mitteilt, dass der Roboter gewonnen hat, speichert die KI den Auftrag in einer Liste und gibt auf der Konsole „Auftrag 0 wurde gespeichert“ aus.
Ist - Reaktion	Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T600 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter einem Team zugewiesen und das Spiel wird gestartet. Dem Roboter wurde einen Auftrag mit ID 0, welcher beim Knoten 1 beginnt und beim Knoten 0 endet, vom Server zugesandt.
Soll - Reaktion	Der Roboter soll dem Wegenetz folgen und Kreuzungen erkennen.
Ist -Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

Testfall	$\langle T700 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter einem Team zugewiesen und das Spiel wird gestartet. Dem Roboter wurde einen Auftrag mit ID 0, welcher beim Knoten 1 beginnt und beim Knoten 0 endet, vom Server zugesandt.
Soll - Reaktion	Der Tankstand muss mit fortlaufender Zeit, in der Tabelle für die Informationen über die Roboter sinken.
Ist – Reaktion	Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T800 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter einem Team zugewiesen und das Spiel wird gestartet. Dem Roboter wurde einen Auftrag mit ID 0, welcher beim Knoten 1 beginnt und beim Knoten 0 endet, vom Server zugesandt.
Soll - Reaktion	Der Roboter kommt am Knoten 0 an.
Ist – Reaktion	Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T900 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter einem Team zugewiesen und das Spiel wird gestartet. Dem Roboter wurde einen Auftrag mit ID 0, welcher beim Knoten 1 beginnt und beim Knoten 0 endet, vom Server zugesandt. Es wurde die Kante zum Knoten 0 gesperrt.
Soll - Reaktion	Der Roboter soll stehen bleiben, da er keine Freigabe erhält.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T1000 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurden geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Dem Roboter wurde einem Transportvolumen von „20“ zugewiesen. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter einem Team zugewiesen und das Spiel wird gestartet. Dem Roboter wurde einen Auftrag mit ID 0, welcher bei 1 beginnt und bei 0 endet und „30“ Einheiten groß war, vom Server zugesandt.
Soll - Reaktion	Der Roboter bietet nicht für den Auftrag.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T1100 \rangle$
Tester	Henrik Lange
Vorbedingung	Der Server ist in Betrieb mit einer Verbindung zu einem Roboter
Eingaben	Login-Name: Test, IP-Adresse: 127.0.0.1
Soll-Reaktion	Bei zuvor gestartetem Server sollte nach Eingabe der angegebenen Daten die Spieler-GUI gestartet werden.
Ist-Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T1200 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter einem Team zugewiesen und das Spiel wird gestartet. Dem Roboter wurde einen Auftrag mit ID 0, welcher beim Knoten 1 beginnt und beim Knoten 0 endet, vom Server zugesandt. Es wurde die Kante zum Knoten 0 gesperrt. Danach wird auch die letzte anliegende Kante gesperrt.
Soll - Reaktion	Der Roboter versucht erst eine alternative Route zu finden, wenn er keine findet soll er auf eine anliegende Kante ausweichen. Nachdem die letzte freie Kante gesperrt wird, dreht der Roboter um und versucht dort auszuweichen.
Ist – Reaktion	Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T1300 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter ein Team zugewiesen und das Spiel wird gestartet. Dem Roboter wurde einen Auftrag mit ID 0, welcher beim Knoten 1 beginnt und beim Knoten 0 endet, vom Server zugesandt.
Soll - Reaktion	Die Positionsdaten des Roboters sollen auf der Serveroberfläche erkennbar sein.
Ist – Reaktion	Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T1400 \rangle$
Tester	Henrik Lange
Vorbedingung	Der Server ist in Betrieb mit einer Verbindung zu einem Roboter.
Eingaben	Es wurde ein Roboter dem Spieler hinzugeführt und ihm wurden Aufträge zugewiesen.
Soll - Reaktion	In der verbundene Spieleroberfläche sollen die Roboter mit ihrer ID, ihrem Namen, ihrer Füllstand des Tanks, ihren Aufträgen, ihrer Fracht, ihrer Kapazität, ihrem Standort und ihrem Status in einer Tabelle aufgeführt werden und aktualisiert werden.
Ist - Reaktion	Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T1500 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter einem Team zugewiesen und das Spiel wird gestartet. Dem Roboter wurde einen Auftrag mit ID 0, welcher beim Knoten 1 beginnt und beim Knoten 0 endet, vom Server zugesandt.
Soll - Reaktion	Die Statistiken in der Tabelle ändern sich mit der Zeit.
Ist – Reaktion	Soll-Reaktion ist eingetreten.
Ergebnis	Der Test war erfolgreich.
Nacharbeiten	Keine Nacharbeiten erforderlich.

Testfall	$\langle T1600 \rangle$
Tester	Markus Björn Meißner
Eingaben	Das Serverprogramm wird gestartet. Eine beliebige Karten XML-Datei wird geladen. Eine beliebige Industrie XML-Datei wird geladen. Der Roboter „LittleG“ wird hinzugefügt und zu Team 0 zugewiesen. In das Feld „Tankvolumen“ wird der Wert „10“ eingetragen. Das Spiel wird gestartet. Es wird eine Zeit von 10 Sekunden abgewartet.
Soll - Reaktion	Nach Spielstart verringert sich LittleG's Tank jede Sekunde um eins. In der Robotertabelle der GUI(Server) kann der Tankstand mitverfolgt werden. Wenn dieser nach 10 Sekunden leer ist, erscheint ein Hinweisfenster: „Regelverletzung: Treibstoff. Bitte den Roboter LittleG vom Spielfeld entfernen.“ . Der Roboter wird von nun an weder auf dem GameDisplay noch in der Robotertabelle angezeigt.
Ist – Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

Testfall	$\langle T1700 \rangle$
Tester	Markus Björn Meißner
Eingaben	Das Serverprogramm wird gestartet. Die Karten XML-Datei „map_weighted“ wird geladen. Die Industrie XML-Datei „Industriedaten_alt“ wird geladen. Ein beliebiger Roboter wird hinzugefügt. Im Initialisierungsfenster werden alle Industrien bis auf „Fleischfabrik“ und „Bauernhof“ entfernt. Das Programm wird gestartet.

Soll - Reaktion	Nach der Initialisierung wird zunächst die Industrieliste auf der Konsole ausgegeben. „[Fleischfarbik Id0 Pos:0 In:[Schweine, Gefluegel] InAmount:[10.0 10.0] Out:[Fleischwaren] outAmount[1.0] false ResIn:[0.0 0.0] ResOut:[0.0], Bauernhof Id1 Pos:1 In:[Getreide, Wasser] InAmount:[10.0 10.0] Out:[Gefluegel] outAmount[1.0] false ResIn:[0.0 0.0] ResOut:[0.0]]“ .Zu Spielbeginn wird dann mit der GameAdministration für jede Industrie die Methode produce() aufgerufen. Es erscheinen die Meldungen „Industrie Bauernhof 1 kann 10.0 Einheiten [Gefluegel] produzieren.üind „Industrie Fleischfarbik 0 kann 10.0 Einheiten [Fleischwaren] produzieren “. Nach Ablauf der jeweiligen Produktionszeit werden von produce() die Konsolenmeldungen „Bauernhof Id1 Pos:1 In:[Getreide, Wasser] InAmount:[0.0 0.0] Out:[Gefluegel] outAmount[11.0] false ResIn:[0.0 0.0] ResOut:[0.0]“ und „Fleischfarbik Id0 Pos:0 In:[Schweine, Gefluegel] InAmount:[0.0 0.0] Out:[Fleischwaren] outAmount[11.0] false ResIn:[0.0 11.0] ResOut:[0.0]“ erzeugt. Man erkennt, dass bei beiden Industrien der „outAmount“ von „1.0“ auf „11.0“ gewechselt hat. Bei beiden Industrien wurden jeweils 10.0 Einheiten neue Ware produziert. Der „inAmount“ hat sich dementsprechend verringert.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

Testfall	$\langle T1800 \rangle$
Tester	Henrik Lange
Eingaben	Das Serverprogramm wurde gestartet. Die Karten-XML „map_weighted“ wurde geladen. Die Industrie-XML „Industriedaten_alt“ wurde geladen. Dann wurde ein Roboter mit dem Namen „NXT_19“ und der Position 1 hinzugefügt. Der Roboter „NXT_19“ wurde vorher gestartet und das gespeicherte Programm gestartet. Danach wurde dem Roboter ein Team zugewiesen. Dann wurde ein Spieler hinzugefügt, dem der Roboter zugeteilt wurde. Der Spieler bot für einen Auftrag von Knoten 1 zu 0, welchen er gewann. Der Spieler hat die Route vom Roboter abgelehnt und wählte eine eigene Route und sendete diese an den Roboter.

Soll - Reaktion	Der Roboter soll ein Route vorschlagen die er nicht sofort ausführt, sondern erst vom Benutzer ein OK erhalten muss. Der Roboter soll die vorgeschlagene Strecke abfahren.
Ist - Reaktion	Die Soll-Reaktion ist eingetreten.
Ergebnis	Der Test konnte erfolgreich durchgeführt werden.
Nacharbeiten	Keine Nachbearbeitung erforderlich.

1.3 Zusammenfassung

Zusammenfassend stellt man fest, dass fast alle Testergebnisse zu Beginn positiv ausgefallen sind; die, die nicht positiv ausfielen, wurden korrigiert. Dies liegt daran, dass während der Programmierung Fehler sofort ausgebessert, aber nicht dokumentiert wurden und daher zum jetzigen Zeitpunkt kaum Nachbesserungen mehr nötig sind. Das Softwareprojekt hat eine hohe Softwarequalität, da alle Tests bestanden wurden und diese das gesamte Spektrum des Projekts abdecken.