Leitzentrale-Modul-Design

Für das studentische Projekt *Sichere Eisenbahnsteuerung*

Datum	17.06.2010
Quelle	Google Code → Dokumente → 02_Design → 02.02_Moduldesign
Autoren	Kai Dziembala
	Norman Nieß
Version	2.0
Status	freigegeben

1 Historie

Version	Datum	Autor	Bemerkung
0.1	30.11.2009	Jan Kremer	Initial
0.2	08.12.2009	Jan Kremer	Aktivitätsdiagramm erstellt
0.3	15.12.2009	Jan Kremer	Aktivitätsdiagramm überarbeitet und in mehrere Diagramme aufgeteilt, Einleitung, Architektur
0.4	21.12.2009	Jan Kremer	Zustandsdiagramm erstellt
1.0	10.01.2010	Jan Kremer	Beschreibung der lokalen Funktionen (Kap. 5.6) erweitert, Datenkapitel ergänzt (Kap. 5.7)
1.1	11.01.2010	Jan Kremer	Namen der modulglobalen Variablen geändert und ins Deutsche übersetzt, ebenso die Diagramme (Kap. 5.7), Typ für die Fahranweisung ergänzt (Kap. 5.5.1), Nachbedingungen bei checkBefahrbarkeit geändert.
1.2	02.02.2010	Jan Kremer	Anpassung des Aktivitätsdiagramm in Kap. 6.3., Entfernen der Weichensperrung (Kap. 5.2), Änderung der lokalen Funktionen und Daten (Kap. 5.6 und 5.7), Hinzufügen der Anbindung ans Auditing-System (Kap. 5.5.3) und den Watchdog (Kap. 5.5.4)
1.3	06.05.2010	Kai Dziembala Norman Nieß	Korrektur der Rechtschreibfehler
1.4	18.05.2010	Kai Dziembala Norman Nieß	Referenzierung des Modul-Design Fahrprogramm (Kapitel 4); Aktualisierung der Referenzen im Kapitel 5
1.5	20.05.2010	Kai Dziembala Norman Nieß	Aktualisierung der Verweise im Kapitel 5.3
2.0	17.06.2010	Kai Dziembala	Freigabe des Moduldesigns 'Leitzentrale'

2 Inhaltsverzeichnis

3 Einleitung

Das Modul *Leitzentrale* wird im Rahmen des studentischen Projektes *Sichere Eisenbahnsteuerung* im Wintersemester 09/10 an der Hochschule Bremen entwickelt. Es ist Teil der Anwendungsschicht und erzeugt aus den von den Fahrprogrammen abgerufenen Fahranweisungen Streckenbefehle, die jede der beiden Loks ihre im Pflichtenheft vorgegebene Fahraufgabe kollisionsfrei erledigen lassen sollen.

Die erzeugten Streckenbefehle werden an das Modul Befehlsvalidierung der Sicherheitsschicht zur Überprüfung weitergegeben. Von diesem Modul werden auch die Sensordaten empfangen, die Aufschluss über die derzeitige Position der Loks und Wagons auf der Modelleisenbahnstrecke geben. Zur Vermeidung von Redundanzen wird die Streckentopologie sowie Gleisabschnitt-/ Weichenbelegung und Zugposition über das Shared Memory von der Befehlsvalidierung zur Verfügung gestellt.

Die Leitzentrale soll nicht nur Kollisionen vermeiden, sondern auch mögliche Fehler beim An- oder Abkoppeln erkennen und diese durch Wiederholung des Vorgangs beheben. Kritische Zustände, wie zwei Loks auf einem Gleisabschnitt, sollen durch vorausschauendes Sperren von Gleisabschnitten für den jeweils anderen Zug vermieden werden.

4 Referenzierte Dokumente

Pflichtenheft: Google Code → Dokumente → 01_Anforderungsanalyse → 01.00_Pflichtenheft → Pflichtenheft

Hardware-Design: Google Code → Dokumente → 02_Design → 02.01_Subsystemdesign → Hardware-Design

Software-Design: Google Code → Dokumente → 02_Design → 02.01_Subsystemdesign → Software-Design

Modul-Design 'Befehlsvalidierung' Google Code → Dokumente → 02_Design → 02.02_Moduldesign → Modul-Design_Befehlsvalidierung

Modul-Design 'Fahrprogramm': Google Code → Dokumente → 02_Design → 02.02_Moduldesign → Modul-Design_Fahrprogramm

5 Architektur

5.1 Übersicht

Ein grundlegender Teil der Architektur baut auf den von der Sicherheitsschicht übermittelten Strukturen auf. Dazu gehören die Streckentopologie, die Gleisabschnitt-/ Weichenbelegung sowie die Zugpositionen. Hinzu kommen von Seiten der Leitzentrale Informationen über die Sperrung von Gleisabschnitten durch Züge sowie ein Ringpuffer, um die von den Fahrprogrammen geholten Fahranweisungen bei einem misslungenen Koppelversuch wiederholen zu können.

5.2 Gleissperrung

Die Gleissperrung wird im ähnlichen Format wie die Gleisabschnitts-/ Weichenbelegung der Befehlsvalidierung (siehe Kapitel 5.3, Befehlsvalidierung-Modul-Design) vorgenommen. Es gibt ein Array *gleisSperrung*: Der Index steht für die Gleisabschnittsnummer (siehe Kapitel 5.2, Befehlsvalidierung-Modul-Design). Der Wert eines Array-Elements gibt die sperrende Lok an: Zum Beispiel:

- gleisSperrung[1] = 0 (Gleis 1 von Lok #1 gesperrt)

5.3 Fahranweisungen

Die Fahranweisungen werden immer einzeln von den Fahrprogrammen abgefragt. Allerdings werden sie intern in einen Ringpuffer, bestehend aus drei Elementen gespeichert. So können die letzten beiden Fahranweisungen (Rangieren, Verlassen des Gleisabschnitts) bei einem misslungenen An- oder Abkoppeln wiederholt werden. Falls eine Aktion wiederholt werden muss, so wird der Index des Ringpuffers um zwei Positionen dekrementiert und es wird nur noch lesend zugegriffen, bis die Wiederholung durchgeführt wurde. Für jede Lok wird ein Ringpuffer angelegt und diese werden zusammen gelegt in einem zweidimensionalen Array *fahranweisungRingpuffer[ANZAHL_LOKS][3]*.

5.4 Abhängigkeiten von anderen Modulen

Die Leitzentrale bekommt von der Befehlsvalidierung über das Shared Memory Informationen über Gleistopologie, Gleisabschnitt-/ Weichenbelegung sowie die Zugpositionen.

5.5 Schnittstellenbeschreibung

5.5.1 Fahrprogramme

Die Schnittstelle zu den Fahrprogrammen ist im Software-Design beschrieben (siehe Kapitel 6.1.1). Um eine Fahranweisung aus 2 Byte zu beschreiben, wird der Dateityp F*ahranweisung* eingeführt. Die *Fahranweisung* ist ein Strukturtyp der zwei jeweils Byte-große Member hat. Die Member-Variable *fahrbefehl* enthält den Fahrbefehl und die Lok, für die der Fahrbefehl gilt. Die Member-Variable *gleisabschnittNr* enthält die Nummer des Gleisabschnitts für die die Fahranweisung gilt:

struct Fahranweisung

- byte fahrbefehl
- byte gleisabschnittNr

5.5.2 Shared Memory

Über das Shared Memory wird mit der Befehlsvalidierung kommuniziert. Von ihr bekommt es die Informationen über Streckentopologie, Gleis- und Weichenbelegung sowie Zugpositionen (s.o). Weiterhin werden die Sensordaten von ihr weitergereicht (Syntax, siehe Software-Design, Kapitel 6.2.1) und Streckenbefehle abgesendet.

5.5.3 Auditing-System

Zur Protokollierung von Fehlerfällen und Informationen, die für das Debugging interessant sein könnten, schickt die Leitzentrale in verschiedenen Situationen 6 Byte-große Nachrichten ans Auditing-System. Es gibt acht verschiedene Fehlercodes, für die jeweils verschiedene Nachrichten generiert werden:

Fehlercode	Beschreibung
FEHLER_ZUSTAND = 0	Sprung in einen nicht existenten Zustand
FEHLER_FAHRBEFEHL = 1	Ausführung eines nicht definierten Fahrbefehls
FEHLER_WEICHENSTELLUNG = 2	Weichenstellung lässt sich nicht bestimmen
ZIEL_ERREICHT = 3	Die angestrebte Zielposition wurde erreicht
KUPPELN_FEHLGESCHLAGEN = 4	Ein Kuppelversuch schlug fehl
ANKUPPEL_VERSUCH = 5	Ein Ankuppelversuch wurde gestartet
ABKUPPEL_VERSUCH = 6	Ein Abkuppelversuch wurde gestartet
NICHT_BEFAHRBAR = 7	Ein Gleisabschnitt ist nicht befahrbar

Die ersten fünf Bytes sind bei allen Fehlercodes gleich kodiert:

Fehlercode	**Alle**
Byte 0	Die untersten 4 Bit speichern den Fehlercode selbst, die obersten 4 Bit speichern die Lok, die gerade gesteuert wird.
Byte 1	Der Zustand, in dem sich Lok #1 befindet (mögliche Zustände s. Kap. 5.7)
Byte 2	Der Zustand von Lok #2
Byte 3	Die Position von Lok #1
Byte 4	Die Position von Lok #2

Das letzte Byte wird unterschiedlich kodiert:

Fehlercode	**FEHLER_ZUSTAND, FEHLER_FAHRBEFEHL, FEHLER_WEICHENSTELLUNG, ZIEL_ERREICHT, KUPPELN_FEHLGESCHLAGEN, ANKUPPEL_VERSUCH, ABKUPPEL_VERSUCH**
Byte 5	Der Fahrbefehl, den die gerade angesteuerte Lok ausführen soll.

Fehlercode	**NICHT_BEFAHRBAR**
Byte 5	Der Gleisabschnitt, in den die gerade angesteuerte Lok einfahren soll.

5.5.4 Software Watchdog

Kann ein Modul während eines Aufrufs seine Aufgabe nicht komplett erfüllen, so soll eine Nachricht an den Software Watchdog geschickt werden. Dabei soll sich jede Nachricht von der vorhergehenden unterscheiden, um sicherzustellen, dass ein Modul mit der Abarbeitung seiner Aufgabe vorangeschritten ist.

Dazu sendet die Leitzentrale den Zustand, die aktuell angesteuerte Lok und einen Zähler, kodiert in einem Status-Byte. Der Zähler wird bei jedem Versand inkrementiert, um sicherzustellen, dass bei normalem Betrieb kein Not-Aus vom Watchdog gesendet wird. Eine Ausnahme bildet der Zustand HOLT_FAHRANWEISUNG. Da dieser Zustand nicht in sich selbst übergehen darf, wird hier kein Zähler definiert. Da sich beim nächsten Versand an den Watchdog der Zustand geändert haben muss, muss sich so auch das Status-Byte geändert haben.

5.6 Lokale Funktionen

Name	**getFahranweisung**
Beschreibung	Holt eine Fahranweisung aus dem Ringpuffer. Je nachdem, ob eine Folge von Fahranweisungen wiederholt werden soll, wird eine neue Fahranweisung aus dem Fahrprogramm in den Ringpuffer geschrieben oder nur lesend darauf zugegriffen.
Vorbedingung	Richtige Lok ist ausgewählt
Parameter	Keine
Rückgabe	Fahranweisung
Nachbedingung	Evtl. neue Fahranweisung vom Fahrprogramm geholt, welches nun auf die nächste zeigt.

Name	**getSensordaten**
Beschreibung	Holt die Sensordaten vom Shared Memory.
Vorbedingung	Neue Sensordaten sind auf dem Shared Memory vorhanden
Parameter	keine
Rückgabe	Sensordaten
Nachbedingung	Sensordaten werden zurückgesetzt (Alle Bytes auf 255).

Name	checkBefahrbarkeit
Beschreibung	Prüft, ob der nächste Gleisabschnitt befahrbar ist.
Vorbedingung	Richtige Lok ist ausgewählt
Parameter	byte gleisabschnittNr
Rückgabe	TRUE, falls der Gleisabschnitt befahrbar ist, sonst FALSE
Nachbedingung	Wenn Rückgabe TRUE ist, ist der Gleisabschnitt gesperrt, ist eine Weiche vor dem Gleisabschnitt, so wird sie gestellt

Name	checkZielpositionErreicht
Beschreibung	Prüft, ob die Zielposition erreicht ist
Vorbedingung	Fahrbefehl wurde erteilt, richtige Lok ist ausgewählt
Parameter	byte zielNr
Rückgabe	TRUE, wenn die Zielposition erreicht ist, sonst FALSE
Nachbedingung	keine

Name	checkZeit
Beschreibung	Prüft, ob Lok #2 noch warten muss
Vorbedingung	Lok #2 wartet, Zeit ist initialisiert
Parameter	keine
Rückgabe	TRUE, wenn noch Zeit ist, sonst FALSE
Nachbedingung	Zeit wurde dekrementiert

Name	checkBelegt
Beschreibung	Prüft, ob der angeforderte Gleisabschnitt belegt ist
Vorbedingung	Gleisabschnitt wurde angefordert, richtige Lok ausgewählt
Parameter	byte gleisabschnittNr
Rückgabe	TRUE, wenn Gleisabschnitt belegt ist, sonst FALSE
Nachbedingung	keine

Name	checkGesperrt
Beschreibung	Prüft, ob der angeforderte Gleisabschnitt durch einen anderen Zug bereits gesperrt ist.
Vorbedingung	Gleisabschnitt wurde angefordert, richtige Lok ausgewählt
Parameter	byte gleisabschnittNr
Rückgabe	TRUE, wenn Gleisabschnitt belegt ist, sonst FALSE
Nachbedingung	keine

Name	setGleisabschnittGesperrt
Beschreibung	Sperrt und entsperrt einen Gleisabschnitt
Vorbedingung	Richtige Lok ausgewählt
Parameter	boolean gesperrt (=0 entsperrt, =1 gesperrt), byte gleisabschnittNr
Rückgabe	keine
Nachbedingung	Gleisabschnitt ist gesperrt oder freigegeben

Name	**setStreckenbefehl**
Beschreibung	Schreibt einen neuen Streckenbefehl ins Shared Memory
Vorbedingung	Richtige Lok ausgewählt
Parameter	byte[] streckenbefehl: 2-Byte Array, das den Streckenbefehl enthält
Rückgabe	keine
Nachbedingung	Streckenbefehl steht im Shared Memory

Name	**checkAnzahlWagons**
Beschreibung	Prüft, ob die Anzahl der Wagons auf einem Gleisabschnitt mit der erwarteten Anzahl übereinstimmt
Vorbedingung	keine
Parameter	byte gleisabschnittNr
Rückgabe	TRUE, wenn die Anzahl mit der erwarteten übereinstimmt, sonst FALSE
Nachbedingung	keine

5.7 Daten

Die oben genannten Funktionen benötigen folgende Variablen:

Name	Verwendung
byte[ANZAHL_LOKS] wiederholen	Gibt an, ob die letzte Fahranweisung wiederholt werden soll, falls ein Kuppelmanöver misslungen ist. Dabei kann wiederholen Werte vom Typen boolean annehmen. Das Array ist vom Typ byte, da der C-Compiler keine Bit-Arrays erlaubt.
Fahranweisung[ANZAHL_LOKS] [3] fahranweisungRingpuffer	Ringpuffer mit 3 Elementen, der die geholten Fahranweisungen (2 Byte) speichert.
Fahranweisung[ANZAHL_LOKS] fahranweisung	Speichert die aktuelle Fahranweisung für die ausgewählte Lok
Zustand[ANZAHL_LOKS] zustand	Zustand in den das Programm für die angegebene Lok beim nächsten Aufruf springen soll. Mögliche Zustände sind: FAHREND, WARTEND, ANGEHALTEN, ANKUPPELND, ABKUPPELND, HOLT_FAHRANWEISUNG
int zeit	Speichert die Pseudo-Zeit, die Lok #2 noch warten muss, falls sie im Zustand WARTEND ist.
Lok lok	Speichert die Lok, für die gerade das Fahrprogramm ausgeführt wird. Mögliche Werte sind LOK1, LOK2 oder KEINE_LOK
KuppelAktion kuppelAktion	Speichert, ob ein An- oder ein Abkuppelvorgang gestartet wurde. Mögliche Werte sind: KEINE, ANKUPPELN, ABKUPPELN
CheckKuppelAktion checkKuppelAktion	Speichert, wann geprüft werden soll, ob das An-/ Abkuppeln geklappt hat. Mögliche Werte sind: JETZT, BEIM_NAECHSTEN_MAL, NIE
Lok[ANZAHL_GLEISE] gleisSperrung	Speichert, welches Gleis von welcher Lok gesperrt ist. Mögliche Werte sind hier: LOK1, LOK2, KEINE_LOK. ANZAHL_GLEISE ist hier die Anzahl der Gleisabschnitte + 1, da die Zählung bei 1 beginnt.
byte kuppelSensor	Speichert, ob der Sensor, der zur Überprüfung des Ankuppelvorgangs benötigt wird, belegt ist.
byte wdStatus	Speichert den aktuellen Status, der an den Watchdog bei

	Verlassen des Moduls gesendet wird.

6 Dynamisches Verhalten

Eine Gesamtübersicht des dynamischen Verhaltens ist im Aktivitätsdiagramm in Abbildung 1 zu sehen:

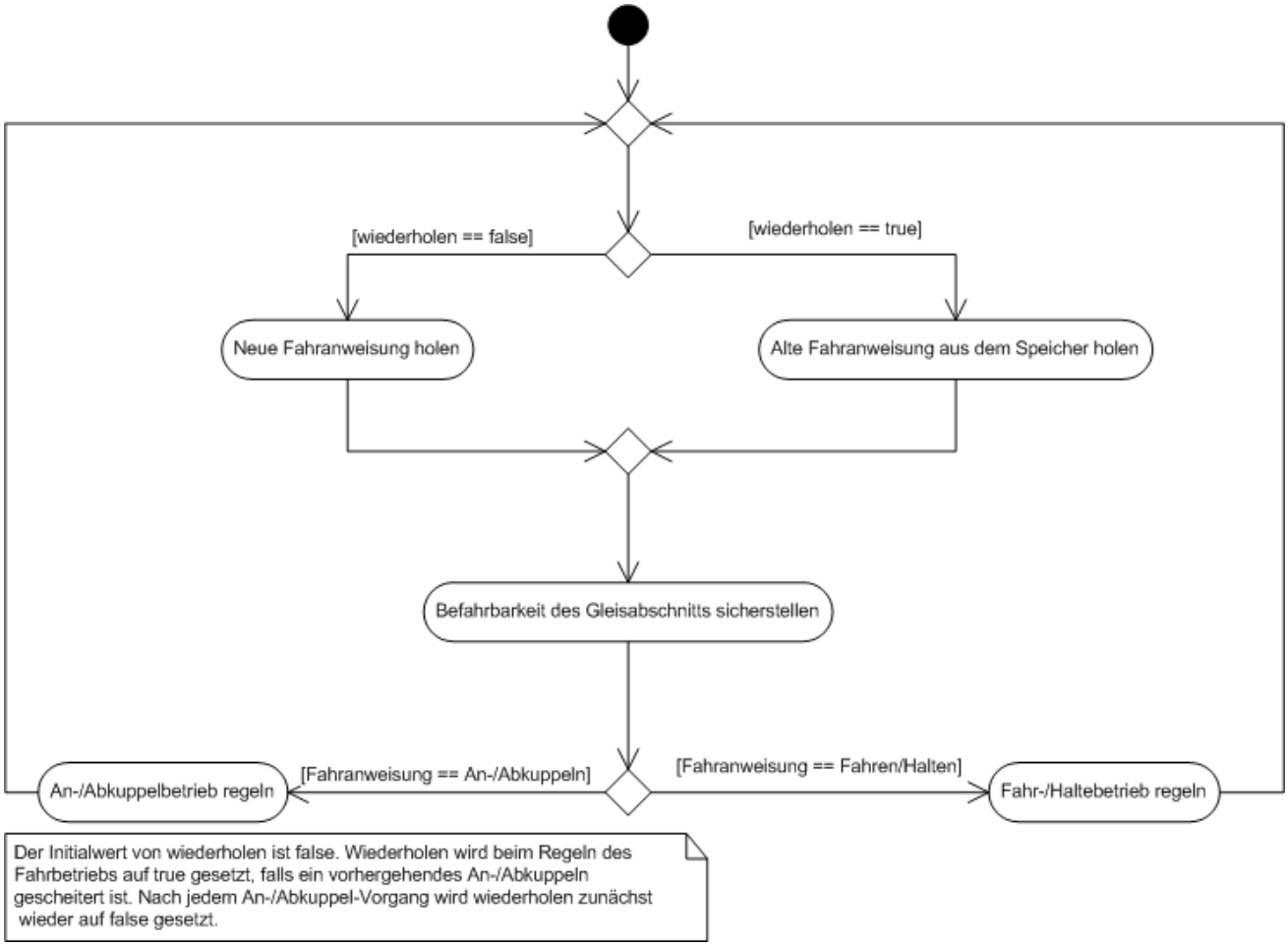

Abbildung 1: Aktivitätsdiagramm - Gesamtübersicht

Im Diagramm sind die fünf wesentlichen Aktivitäten der Leitzentrale aufgeführt. Zunächst wird geprüft, ob eine Wiederholung auf Grund eines misslungenen An- oder Abkoppelvorgangs nötig ist. Danach wird sichergestellt, dass der Gleisabschnitt auch befahrbar ist. Schließlich wird in Abhängigkeit von der Fahranweisung der Fahr- oder Rangierbetrieb geregelt.

Ist eine Wiederholung des An- bzw. Abkoppelns nötig, so werden die letzten beiden Fahranweisungen (Rangieren und Verlassen des Gleisabschnitts) noch einmal ausgeführt.

6.1 Befahrbarkeit des Gleisabschnitts sicherstellen

Damit der Zug die für ihn vorgesehene Fahranweisung ausführen kann, muss zunächst sichergestellt sein, dass er in den gewünschten Gleisabschnitt auch einfahren darf. Abbildung 2 zeigt das Aktivitätsdiagramm hierfür:

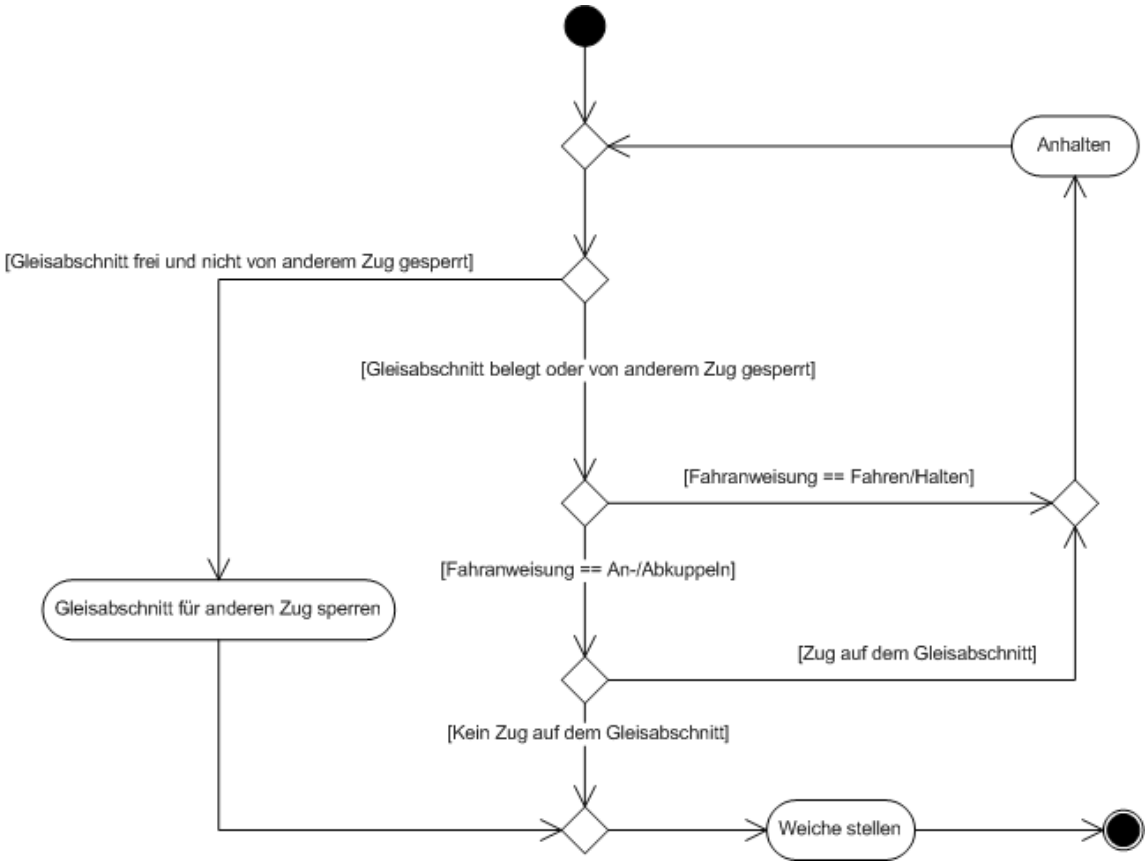

Abbildung 2: Aktivitätsdiagramm - Befahrbarkeit sicherstellen

Sollte der gewünschte Gleisabschnitt frei und nicht gesperrt sein, so wird er selbst für den anderen Zug gesperrt und evtl. die Weiche gestellt. Sollte dies nicht der Fall sein, so wird noch geprüft, ob in den entsprechenden Gleisabschnitt mit hoher Geschwindigkeit eingefahren werden soll (das ist bei den Fahranweisungen Fahren oder Halten der Fall). Hier wird der Zug sofort angehalten. Ist der Zug im Rangierbetrieb, wird noch überprüft ob sich ein anderer Zug oder nur Wagons auf dem Gleisabschnitt befinden. Sind es nur Wagons, darf der Zug in den Gleisabschnitt einfahren.

6.2 An-/ Abkoppelbetrieb regeln

Abbildung 3 zeigt das Aktivitätsdiagramm zur Regelung des An-/ Abkoppelbetriebs:

Abbildung 3: Aktivitätsdiagramm – An-/ Abkoppelbetrieb regeln

Zunächst wird in Abhängigkeit der Fahranweisung, die benötigte Geschwindigkeit ausgewählt. Daraufhin wird so lange gewartet, bis die Zielposition erreicht und abgekoppelt werden kann bzw. bis angekoppelt wurde. Danach gibt man noch den gerade verlassenen Gleisabschnitt frei und setzt die beiden Variablen *checkKuppelAktion* und *wiederholen; checkKuppelAktion* gibt an, wann der Abkoppelvorgang überprüft werden soll.

Die drei Werte BEIM_NAECHSTEN_MAL, JETZT und NIE sind hier möglich. BEIM_NAECHSTEN_MAL bedeutet, dass bei Anforderung der übernächsten Fahranweisung bis zum Erreichen des dadurch angeforderten Gleisabschnitts geprüft werden soll. JETZT bedeutet, dass bis zum Erreichen des aktuell angeforderten Gleisabschnitts geprüft werden soll. NIE heißt, dass gar nicht geprüft werden soll. Diese Variable ist deswegen nötig, da man erst aus einem Gleisabschnitt, in dem man abgekoppelt hat, hinausfahren muss, bis man feststellen kann, ob nun wirklich alles funktioniert hat.

wiederholen gibt an, ob die Koppelaktion wiederholt werden muss oder nicht. Zunächst wird angenommen, dass dies nicht der Fall ist, solange bis ein Fehler beim Kuppeln festgestellt wurde.

Da sich die Strategien zur Erkennung von Fehlern zwischen An- und Abkoppeln unterscheiden, wird in der Variable *kuppelAktion* gespeichert, ob als letztes an- (ANKUPPELN), ab- (ABKUPPELN) oder gar nicht (KEINE) gekoppelt wurde. Diese Variable wird später beim Regeln des Fahr- und Haltebetriebs noch einmal abgefragt, um die entsprechenden Maßnahmen treffen zu können, falls ein Fehler aufgetreten sein sollte.

Die Zielposition beim Ankoppeln ist erreicht, sobald der Prüfsensor am Ende des Kupplungsgleises eine Belegung meldet. Beim Abkuppeln ist die Zielposition erreicht, sobald die Lok den Sensor vor dem Entkoppler passiert hat.

6.3 Fahr-/ Haltebetrieb regeln

Abbildung 4 zeigt das Aktivitätsdiagramm für den Fahr-/Haltebetrieb:

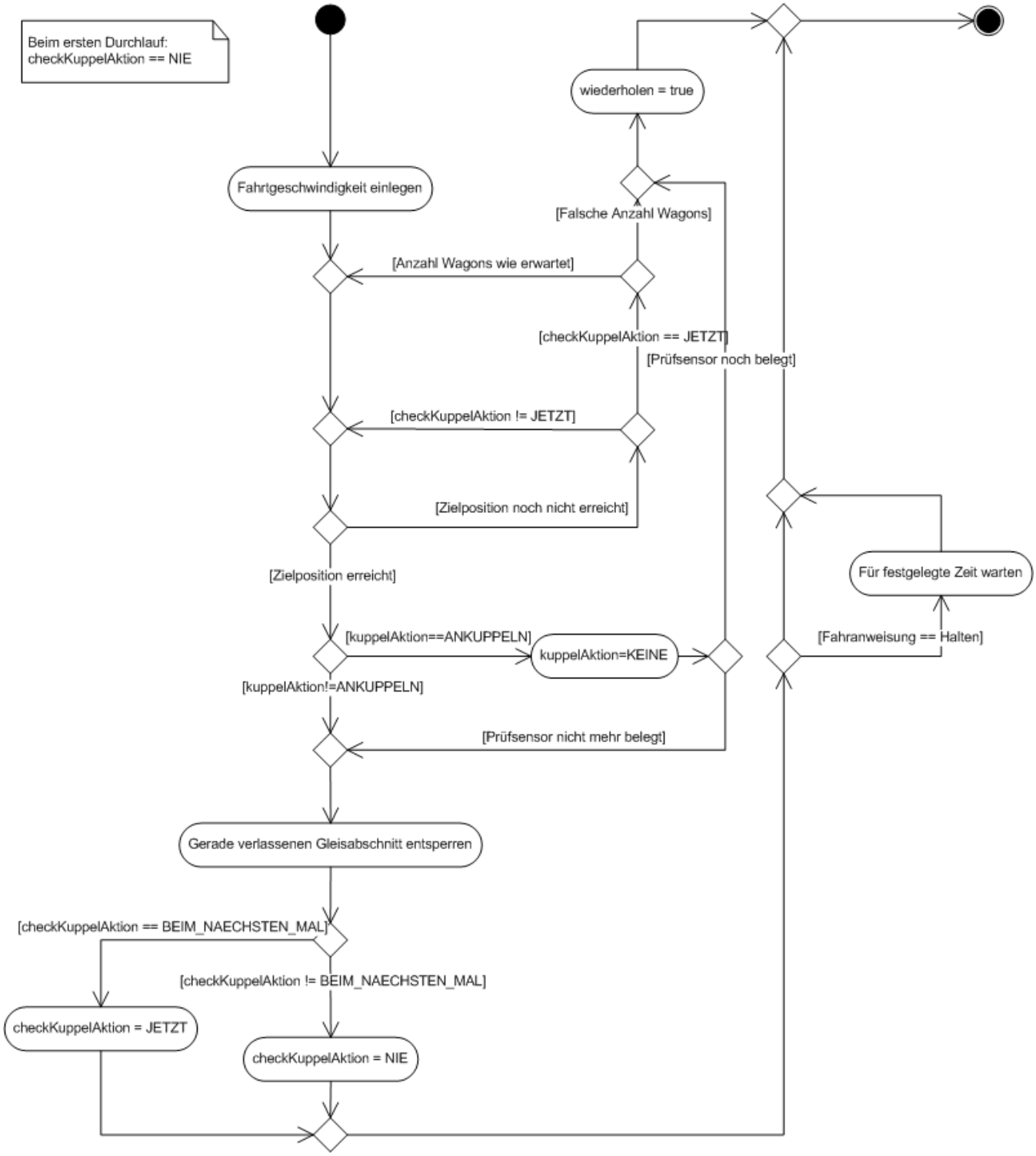

Abbildung 4: Aktivitätsdiagramm - Fahr-/Haltebetrieb regeln

Auch hier wird zunächst die benötigte Geschwindigkeit geregelt. Solange die Zielposition nicht erreicht ist, wird geprüft, ob sich schon feststellen lässt, ob eine evtl. Ankoppelanweisung fehlgeschlagen ist. Ist dies möglich und eine andere Anzahl an Wagons auf dem Gleisabschnitt als die erwartete, so wird durch *wiederholen=true* eine Wiederholung des Vorgangs erzwungen. Andernfalls wird bei Erreichen der Zielposition der verlassene Gleisabschnitt entsperrt und evtl. die *checkKuppelAktion*-Variable angepasst.

Zum Schluss wird noch überprüft, ob die Fahranweisung die Aktion Halten enthielt, um dann für die in der Anweisung festgelegte Zeit auf dem Gleisabschnitt zu warten. Ist die Zielposition erreicht, so wird über die Variable kuppelAk*tion* abgefragt, ob vorher angekoppelt wurde. Sollte dies der Fall sein, wird überprüft, ob sich ein Wagon noch auf dem Prüfsensor am Ende des Kupplungsgleises befindet. Bei einer Belegung des Sensors ist offensichtlich ein Fehler beim Ankoppeln passiert und der Vorgang muss wiederholt werden.

6.4 Zustandsautomat

Aus den Aktivitätsdiagrammen ergeben sich fünf Zustände, in denen auf ein neues Ereignis gewartet werden muss. Um an diesen Stellen nicht durch *Busy Waiting* den Prozessor für die anderen Funktionen zu blockieren, wird sobald einer dieser Zustände erreicht ist, dieser gespeichert und die Kontrolle an die Betriebsmittelverwaltung zurückgegeben. Abbildung 5 zeigt das Zustandsdiagramm für diesen Automaten:

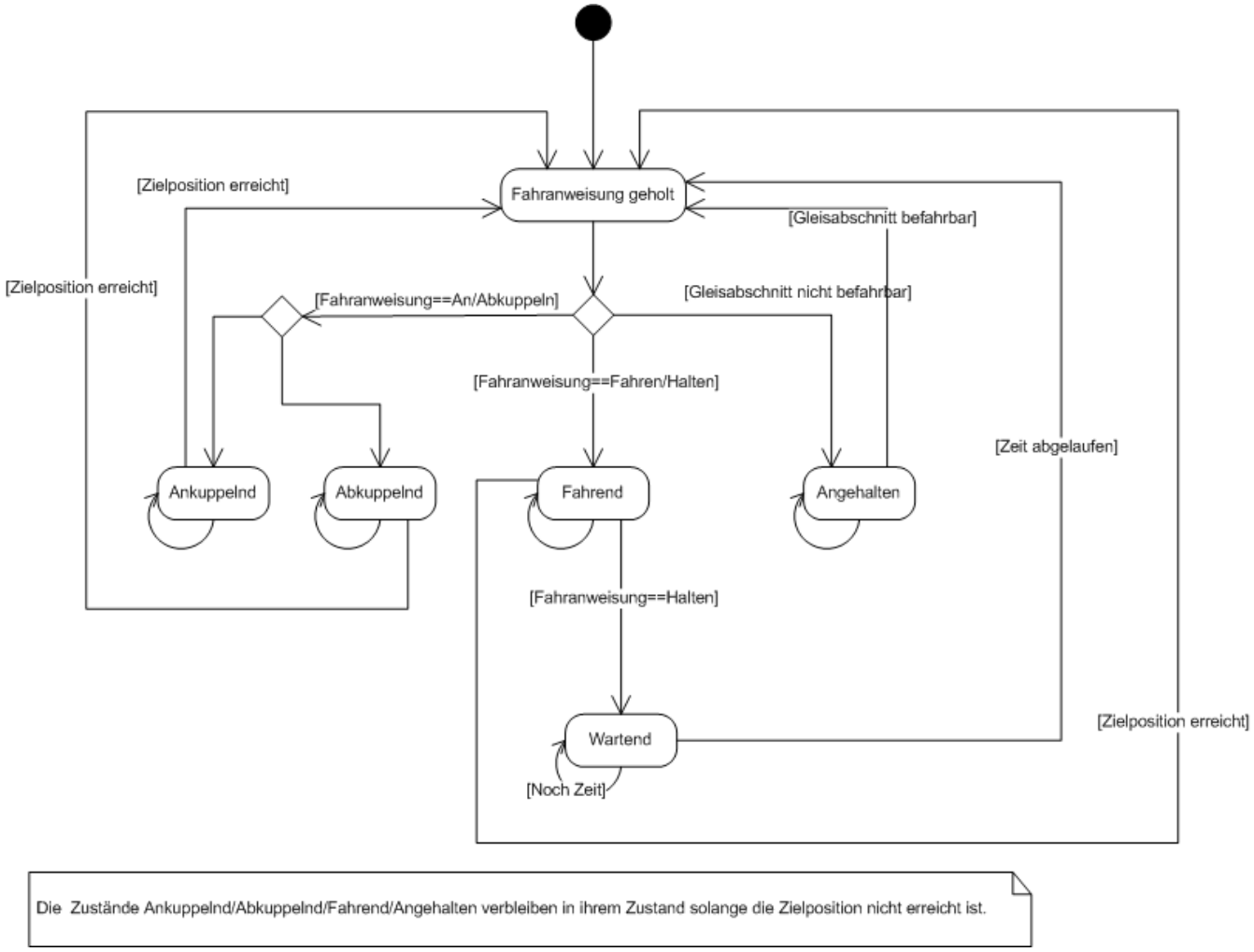

Abbildung 5: Zustandsdiagramm - Gesamtübersicht