Requirements and Design

PR Anwendungsentwicklung i.d. Medieninformatik

WS 2015/16 – Gruppe WK2

Document Control

Contributors

Person	Role	Company	Contribution
Christoph Pressler	Project Member	UNIVIE	Modelling and implementation
Maximilian Steiner	Project Member	UNIVIE	Modelling and implementation
Thomas Schönmann	Project Member	UNIVIE	Modelling and implementation

Revision History

Issue	Author	Date	Description
v0.1	Christoph Pressler	31-10-2016	Git project Initialisierung, leichte Anpassungen der Vorlage
v0.2	Thomas Schönmann	31-10-2016	Weitere Anpassungen der Vorlage, mehrere Use-Case Diagramm Prototypen hinzugefügt
v0.2.1	Christoph Pressler	01-11-2016	Ein Use-Case Diagramm hinzugefügt
v0.2.2	Maximilian Steiner	02-11-2016	Mehrere Use-Case Diagramme hinzugefügt
v0.2.3	Maximilian Steiner	03-11-2016	Zusammenfassung der verschiedenen Use-Case Diagramme zu einem Sinnvollen.
v0.2.4	Christoph Pressler	04-11-2016	„Application Overview and Objectives“ hinzugefügt
v0.4	Thomas Schönmann	05-11-2016	Komponentendiagramm hinzugefügt
v0.5	Thomas Schönmann	06-11-2016	Beschreibungen, für die Komponenten hinzugefügt
v0.5.1	Christoph Pressler	06-11-2016	Einzelnes Klassendiagramm hinzugefügt
v0.8	Thomas Schönmann	07-11-2016	Alle restlichen Klassendiagramme hinzugefügt, Komponentendiagramm aktualisiert, Dokumentation der einzelnen Komponenten verbessert
v0.9	Maximilian Steiner	07-11-2016	Projektmanagement hinzugefügt
v0.9.1	Christoph Pressler	07-11-2016	Komponentenbeschreibung einer Komponente hinzugefügt
v0.9.2	Maximilian Steiner	07-11-2016	Kleine Änderungen / Fehler ausgebessert
v1.0	Christoph Pressler	07-11-2016	Kleine Änderungen / Fehler ausgebessert
V1.1	Maximilian Steiner	19-11-2016	Use-Case Diagramm geändert und textuelle Beschreibung hinzugefügt.

Table of Contents

1. Application Overview and Objectives

1.1. Application Overview

Das grundlegende Ziel der Applikation ist es den EV3 Lego MindStorm Roboter anzusprechen und dem Benutzer verschiedene Möglichkeiten zur Nutzung des Roboters zu geben.

Einerseits soll der Nutzer den Roboter, idealerweise über ein Graphical User Inferface, direkt steuern, sprich ihn vorwärts- und rückwärtsfahren, sowie nach links und rechts drehen lassen können. Neben diesen grundlegenden Funktionalitäten, soll es dem Nutzer auch möglich sein den Roboter kompliziertere Aufgaben erfüllen zu lassen. So ist geplant, den Roboter farbige DUPLO Blöcke sortieren zu lassen, sollte die Zeit reichen, soll der Roboter auch einen Raum ausmessen, sowie durch ein Labyrinth finden können-

Die Zielnutzer des Programms sind hauptsächlich technikinteressierte bzw. roboterinteressierte Menschen. Die Applikation selber soll vor allem zeigen was möglich ist mit einem Lego MindStorm Roboter und sich so das Staunen verschiedenster Nutzergruppen sichern.

1.2. Objectives

- ➔ Verbindung zwischen dem Roboter und einem Computer aufbauen
- ➔ Direkte Steuerung des Roboters über einen Computer
- ➔ Ausführen vorprogrammierter „Kunststücke“ (Raum vermessen, DUPLO Blöcke sortieren, Labyrinth durchfahren)
- ➔ Interesse des Nutzers wecken

2. Use Cases

Textuelle Beschreibung:

Die beschriebenen Anwendungsfälle bestehen vor allem aus den zwei Hauptakteuren „Nutzer“ und „Roboter“, wobei Zweitgenannter den Großteil aller Aktionen umsetzt.
Unsere Use-Cases beschreiben ein System in dem der Roboter in einem ihm unbekannten Raum hin und herfährt.
Er soll in dieser Testumgebung Hindernisse erkennen und finden. Dies erfolgt mit seinem Ultraschallsensor. Sobald er ein Hindernis gefunden hat fährt er an es heran bis er die Farbe mit dem Farbsensor erkennen kann. Dann führt er gewisse Aktionen anhand der erkannten Farbe aus. Wir werden in der Testumgebung rote, gelbe und blaue Blöcke verteilen. An einem roten Block angekommen muss der Roboter umdrehen, bei einem gelben macht er eine n Grad Wende nach links und bei einem blauem eine Wende nach rechts. Falls er an ein anderes Hindernis gerät (z.B.: weiße Wand) dann dreht er sich um n Grad. Danach versucht dann wieder geradeaus zu fahren. Falls dies nicht möglich ist weil wieder ein Hindernis erkannt wurde, dreht er sich nochmal um diese n Grad. Diesen Vorgang wiederholt er bis kein Hindernis mehr erkannt wird. Falls es sich um eine Wand handelt würde der Roboter nun ungefähr im 90 Grad Winkel zur Wand stehen.
So fährt der Roboter in unserem Testbereich hin und her. Mithilfe der bunten Blöcke kann man dem Roboter auch eine Spur legen der er folgt.
Nun soll der Roboter aber nicht ewig hin und her fahren. Also brauchen wir eine Bedingung bei der der Roboter aufhört.
Allerdings muss sich der Roboter dann auch merken was bisher passiert ist. Beispielweise wenn wir bestimmen das sobald 3mal hintereinander ein gleichfarbiger Block erkannt wurde er stehen bleibt und das Programm beendet. Dafür muss sich der Roboter allerdings immer die 3 vorherigen Blöcke merken. Optional könnte der Roboter sich auch den bisherigen Weg merken und aufzeichnen.
Anhand der Aufzeichnung könnte auch eine Terminierungsbedingung gefunden werden.
Entweder ein bestimmter zurückgelegter Weg oder andere Muster. Wir definieren also der Roboter beendet den Vorgang falls der Fall von n abgefahrenen Blöcken, n gefahrene Zeit oder n zurückgelegte Meter eingetreten ist.

2.1. Roles and Actors

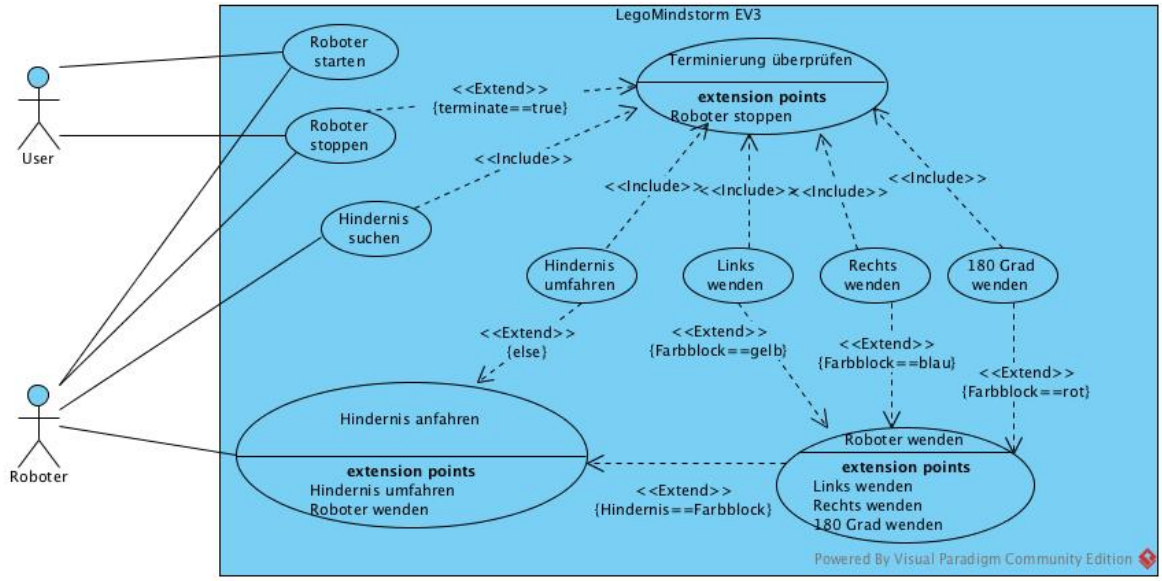
2.1.1. Nutzer

Der Nutzer initiiert den Prozess bzw. hat die Möglichkeit, diesen jederzeit zu beenden (remote kill).

2.1.2. Roboter

Der Roboter setzt den vom Nutzer bereitgestellten Algorithmus in die Tat um, um vordefinierte Probleme zu lösen. Er wird durch ein Lego „EV3“ MindStorm Bauset physikalisch repräsentiert.

2.2. Use Case Diagrams



2.3. Textual Use Case Descriptions

Category	Entry
Use Case ID	***1***
Title	***Roboter starten***
Scope	
Actors	*User, Roboter*
Preconditions	*Der Roboter ist mit dem Computer korrekt verbunden*
Success Guarantees	*Die Einstellungen der Umgebung sind korrekt.*
Trigger	*Der Roboter wird gestartet.*

Main Success Scenario	*1. Der Roboter fährt hoch.* *2. Der Roboter wartet auf Anweisungen*	
Exceptions	1	...
	2	….
	3	….
Version	*1.1*	
Last updated	*19.11.16*	
Issues/Comments		

Category	Entry
Use Case ID	***2***
Title	***Roboter stoppen***
Scope	
Actors	*Roboter, User*
Preconditions	*Der Roboter ist gestartet*
Success Guarantees	*Der Roboter ist korrekt verbunden.*
Trigger	*User stoppt den Roboter.*

	Terminierungsbedingung ist erfüllt.	
Main Success Scenario	1. *Der Roboter unterbricht die jeweilige Aktion.* 2. *Der Roboter fährt herunter.*	
Exceptions	1	....
	2	....
	3	….
Version	*1.1*	
Last updated	*19.11.16*	
Issues/Comments	*Falls der User den Roboter stoppt dann muss das sofort geschehen. Der Roboter beendet jegliche Aktion und fährt herunter. So kann gewährleistet werden, dass man ihn sofort abschalten kann, wenn er sich in Gefahrensituationen begibt.*	

Category	Entry
Use Case ID	***3***
Title	***Hindernis suchen***
Scope	
Actors	*Roboter*
Preconditions	*Der Roboter ist betriebsbereit.* *Der Roboter hat noch kein Hindernis gefunden.* *Der Roboter hat vorherige Aktionen bezüglich Hindernissen abgeschlossen.*
Success Guarantees	*Sämtliche Sensoren des Roberts sind korrekt mit dem Brick-Rechner verbunden, außerdem muss die Batterieladung der Nutzungsdauer entsprechend hoch sein.*

Trigger	*Die Anwendung wird gestartet.*	
Main Success Scenario	1. *Der Roboter scannt die Umgebung mithilfe seines Ultraschallsensors.* a. *Bei erstmaliger Ausführung dieses Use-Cases scannt er die Umgebung in einem n Grad Winkel.* b. *Andernfalls scannt er nur vor sich, geradeaus.* 2. *Der Roboter hat ein Hindernis erkannt und beendet diesen Use-Case.* 3. *Der Roboter hat kein Hindernis erkannt und führt die Terminierungsüberprüfung aus. Falls diese Negativ ist macht er weiter mit Schritt 4.* 4. *Der Roboter fährt n Meter geradeaus bleibt stehen und springt zu Schritt 1*	
Exceptions	1	Der Roboter ist bereits n Meter geradeaus gefahren und hat immer noch kein Hindernis erkannt.
	2	Ein unerkanntes Hindernis ist im Weg
Version	*1.1*	
Last updated	*19-November-2016*	
Issues/Comments	*Wie weit reicht der Ultraschallsensor? Was wenn er beim ersten Scan mehrere Hindernisse erkennt.*	

Category	Entry
Use Case ID	***4***
Title	***Hindernis anfahren***
Scope	
Actors	*Roboter*
Preconditions	*Der Roboter ist betriebsbereit.* *Der Roboter hat ein Hindernis gefunden.* *Der Roboter ist nicht bereits zu nah an einem Hindernis.*
Success Guarantees	*Sämtliche Sensoren des Roberts sind korrekt mit dem Brick-Rechner verbunden, außerdem muss die Batterieladung der Nutzungsdauer entsprechend hoch sein.*
Trigger	*Hindernis gefunden*

Main Success Scenario	1. *Der Roboter richtet sich auf das erkannte Hindernis aus.* 2. *Der Roboter fährt n Meter auf das Hindernis zu bis er die Farbe des Hindernisses erkennen kann.* 3. *Der Roboter bleibt stehen.*	
Exceptions	1	Probleme bei der Farberkennung.
	2	Ein unerkanntes Hindernis ist im Weg.
Version	*1.1*	
Last updated	*19-November-2016*	
Issues/Comments	*Wie weit reicht der Farbsensor? Der Roboter muss sich so ausrichten das das Hindernis auch tatsächlich erreicht wird. Eventuell muss er am Weg noch öfter scannen um festzustellen das er am rechten Weg ist.*	

Category	Entry
Use Case ID	***5***
Title	***Roboter wenden***
Scope	
Actors	*Roboter*
Preconditions	*Der Roboter ist betriebsbereit.* *Der Roboter hat ein Hindernis gefunden.* *Der Roboter ist nahe genug um die Farbe zu erkennen.* *Das Hindernis ist ein Farbblock also rot, blau oder gelb.*
Success Guarantees	*Sämtliche Sensoren des Roberts sind korrekt mit dem Brick-Rechner verbunden, außerdem muss die Batterieladung der Nutzungsdauer entsprechend hoch sein.*
Trigger	*Hindernis gefunden und Hindernis ist ein Farbblock der Farbe rot, gelb oder blau.*

Main Success Scenario	1. *Der Roboter analysiert die erkannte Farbe und ordnet das erkannte Farbspektrum dem „Hue“ zu.* 2. *Je nach erkanntem „Hue“ führt er eine bestimmte Aktion aus.*	
Exceptions	1	....
	2	....
Version	*1.1*	
Last updated	*19-November-2016*	
Issues/Comments	Hier spielt der Farbscanner eine wichtige Rolle. Da je nach erkannter Farbe eine andere Aktion ausgeführt werden soll müssen wir sicherstellen das der Farbscanner die Farben auch Richtig erkennt. Wir müssen daher unseren ausgewählten Hues Rot, Blau und Gelb die korrekten Farbspektren zuweisen damit der Roboter sie auch erkennt. Zu beachten sind auch Helligkeit der Testumgebung und andere Faktoren die den Farbscanner beeinflussen können. Je nachdem wie der Farbscanner Farben erkennt, müssen wir das in der Implementierung beachten. Anmerkung:“Hue“ ist der Name der Farbe z.B.: “Rot“ Rot kann ja in verschiedenen Sättigungen und Helligkeiten auftreten. Der Überbegriff ist der „Hue“.	

Category	Entry
Use Case ID	***6***
Title	***Links wenden***
Scope	
Actors	*Roboter*
Preconditions	*Der Roboter ist betriebsbereit.* *Der Roboter hat ein Hindernis gefunden.* *Der Roboter ist nahe genug um die Farbe zu erkennen.* *Das Hindernis ist ein gelber Farbblock*

Success Guarantees	*Sämtliche Sensoren des Roberts sind korrekt mit dem Brick-Rechner verbunden, außerdem muss die Batterieladung der Nutzungsdauer entsprechend hoch sein.*	
Trigger	*Hindernis gefunden und Hindernis ist ein Farbblock der Farbe Gelb.*	
Main Success Scenario	1. *Der Roboter macht eine n Grad wende nach links bis der gelbe Farbblock nicht mehr im Weg ist.* 2. *Weiter zur Terminierungsüberprüfung* 3. *Falls diese negativ ausfällt wird Use-Case 3 ausgeführt.*	
Exceptions	1	....
	2	....
Version	*1.1*	
Last updated	*19-November-2016*	
Issues/Comments	Es muss definiert werden wie weit sich der Roboter dreht. Falls der Roboter zu nah an dem Farbblock ist, weil die Reichweite des Farbscanners nicht besonders hoch ist oder er aus anderen Gründen zu nah steht muss er zuerst ein Stück zurückfahren.	

Category	Entry
Use Case ID	***7***
Title	***Rechts wenden***
Scope	
Actors	*Roboter*
Preconditions	*Der Roboter ist betriebsbereit.* *Der Roboter hat ein Hindernis gefunden.* *Der Roboter ist nahe genug um die Farbe zu erkennen.* *Das Hindernis ist ein blauer Farbblock*
Success Guarantees	*Sämtliche Sensoren des Roberts sind korrekt mit dem Brick-Rechner verbunden, außerdem muss die Batterieladung der Nutzungsdauer entsprechend hoch sein.*

Trigger	*Hindernis gefunden und Hindernis ist ein Farbblock der Farbe Blau.*	
Main Success Scenario	1. *Der Roboter macht eine n Grad wende nach rechts bis der blaue Farbblock nicht mehr im Weg ist.* 2. *Weiter zur Terminierungsüberprüfung* 3. *Falls diese negativ ausfällt wird Use-Case 3 ausgeführt.*	
Exceptions	1	....
	2	....
Version	*1.1*	
Last updated	*19-November-2016*	
Issues/Comments	Es muss definiert werden wie weit sich der Roboter dreht. Falls der Roboter zu nah an dem Farbblock ist, weil die Reichweite des Farbscanners nicht besonders hoch ist oder er aus anderen Gründen zu nah steht muss er zuerst ein Stück zurückfahren.	

Category	Entry
Use Case ID	***8***
Title	***180 Grad wenden***
Scope	
Actors	*Roboter*
Preconditions	*Der Roboter ist betriebsbereit.* *Der Roboter hat ein Hindernis gefunden.* *Der Roboter ist nahe genug um die Farbe zu erkennen.* *Das Hindernis ist ein roter Farbblock*
Success Guarantees	*Sämtliche Sensoren des Roberts sind korrekt mit dem Brick-Rechner verbunden, außerdem muss die Batterieladung der Nutzungsdauer entsprechend hoch sein.*
Trigger	*Hindernis gefunden und Hindernis ist ein Farbblock der Farbe Rot.*

Main Success Scenario	1. *Der Roboter macht eine 180 Grad Wende.* 2. *Weiter zur Terminierungsüberprüfung* 3. *Falls diese negativ ausfällt wird Use-Case 3 ausgeführt.*	
Exceptions	1	....
	2	....
Version	*1.1*	
Last updated	*19-November-2016*	
Issues/Comments		

Category	Entry
Use Case ID	***9***
Title	***Hindernis umfahren***
Scope	
Actors	*Roboter*
Preconditions	*Der Roboter ist betriebsbereit.* *Der Roboter hat ein Hindernis gefunden.* *Der Roboter ist nahe genug um die Farbe zu erkennen.* *Das Hindernis ist kein Farbblock*
Success Guarantees	*Sämtliche Sensoren des Roberts sind korrekt mit dem Brick-Rechner verbunden, außerdem muss die Batterieladung der Nutzungsdauer entsprechend hoch sein.*
Trigger	*Hindernis gefunden und Hindernis ist kein Farbblock*

Main Success Scenario	1. *Der Roboter fährt n Meter rückwärts.* 2. *Der Roboter dreht sich um n Grad.* 3. *Weiter zur Terminierungsüberprüfung* 4. *Falls diese negativ ausfällt wird Use-Case 3 ausgeführt.*	
Exceptions	1	....
	2	....
Version	*1.1*	
Last updated	*19-November-2016*	
Issues/Comments	Es muss definiert werden wie weit sich der Roboter dreht. Nachdem der Roboter diese Aktion ausgeführt hat muss er sie so oft wiederholen bis er das Hindernis umfahren hat. Dies könnte gewährleistet werden indem der Roboter diesen Use-Case iteriert bis er an dem Hindernis vorbeifahren kann. Wir allerdings haben uns entschieden er folgt dem restlichen Programmverlauf in dem er als nächstes die Terminierungsbedingung überprüft. Falls diese nicht zutrifft macht er weiter bei Use-Case 3, der unweigerlich wieder zu diesem Use-Case zurückführt, es sei denn das Hindernis ist bereits umfahren.	

Category	Entry
Use Case ID	***10***
Title	***Terminierung überprüfen***
Scope	
Actors	*Roboter*
Preconditions	*Der Roboter ist betriebsbereit.* *Der Roboter hat eine Aktion ausgeführt die die Terminierungsüberprüfung aufruft.*
Success Guarantees	*Sämtliche Sensoren des Roberts sind korrekt mit dem Brick-Rechner verbunden, außerdem muss die Batterieladung der Nutzungsdauer entsprechend hoch sein.*
Trigger	*Terminierungsüberprüfung wird aufgerufen*

Main Success Scenario	*1. Der Roboter überprüft die Terminierungsbedingungen.* *a. Terminierungsbedingung ist eingetroffen und er beendet das Programm* *b. Terminierungsbedingung ist nicht eingetroffen und er macht normal weiter.*	
Exceptions	1	....
	2	....
Version	*1.1*	
Last updated	*19-November-2016*	
Issues/Comments	Die Terminierungsbedingungen müssen definiert werden. Essentiell sind 2 Bedingungen: Beende fall bereits n Meter gefahren sind. Beende falls bereits n Zeit vergangen ist. Zusätzlich kann man noch Terminierungsbedingungen nach anderen Logiken einfügen. Beispielweise: Beende nach einer bestimmten Reihenfolge von Farbblöcken	

3. Architecture

3.1. Overview

Die Applikation und ihre Komponenten werden in einem ROS-Package gekapselt um mit dem EV3 Betriebssystem „LejOS“ zusammenzuarbeiten. Ein solches ROS-Package beinhaltet je eine Applikation.

Die Hauptkomponenten einer solchen Applikation können in folgende drei Kategorien zerlegt werden:

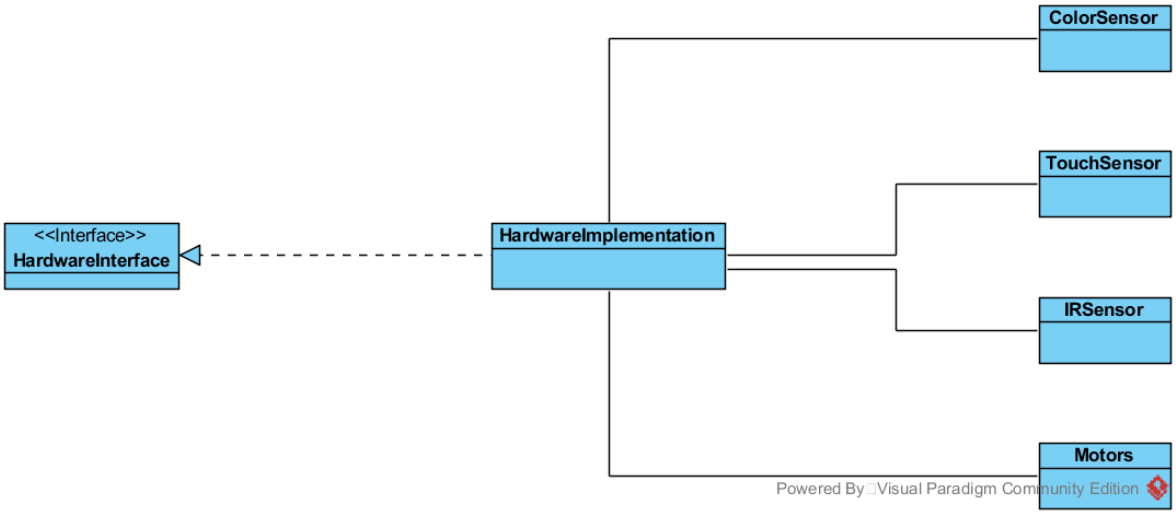
1. Nutzereingaben. Es wird ein Client für den Benutzer, welcher die Applikation aus der Ferne starten/stoppen kann, sowie ein input-handler, welcher die Eingaben verarbeiten kann, bereitgestellt.
2. Hardware-orientierte Komponenten. Aufgrund der vielen Sensoren des Roboters ist es das Ziel vieler Komponenten mit diesen Sensoren zu kommunizieren und es dem Programm zu ermöglichen auf Basis von Echtzeitdaten Entscheidungen zu treffen. Diese Komponente markieren die Schnittstelle zum Roboter

3. Entscheidungsfindung. Die Daten des Roboters und Nutzereingaben alleine reichen nicht aus um den Roboter zu steuern. Daher muss es Komponenten geben, welche diese Daten mit Programmlogik verbinden um das gewünschte Verhalten zu erzeugen.

3.2. Architecture for Component *Hardware Access*

3.2.1.1. Design Considerations

Das HardwareInterface bietet allen anderen Komponenten abstrahierten Zugriff auf die Hardware des Roboters. HardwareImplementation implementiert die Methoden des Interface und greift dafür auf die Klassen „ColorSensor“, „TouchSensor“, „IRSensor“ und „Motors“ zu, wobei jede dieser Klassen Zugriff auf die jeweilige Hardware Komponente bietet



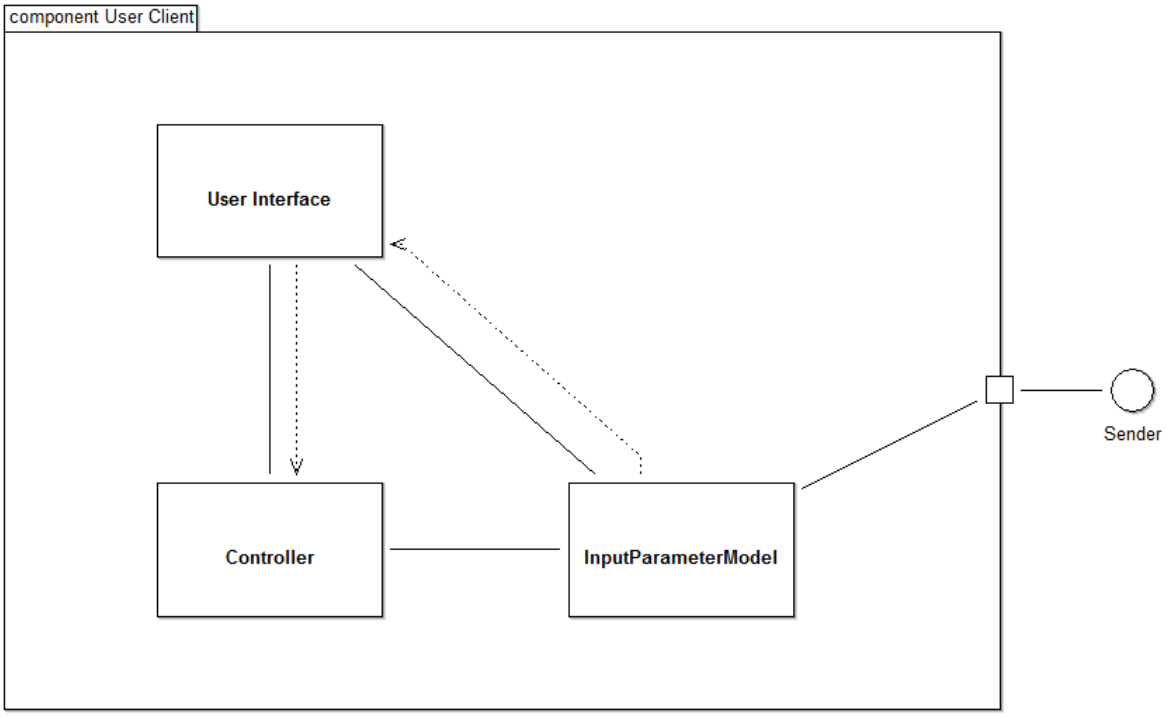
3.2.1.2. Cross References

- *Jeder Use-Case macht Gebrauch von dieser Komponente.*

3.3. Architecture for Component *User Client*

3.3.1.1. Design Considerations

Der Nutzer soll die Möglichkeit haben, den Roboter aus der Ferne zu steuern (Use-Case-spezifisch), zumindest aber ihn jederzeit zu starten bzw. pausieren. Für die Implementierung der User-Client-Komponente wird das „Model-View-Controller"-Anwendungsmuster verwendet.



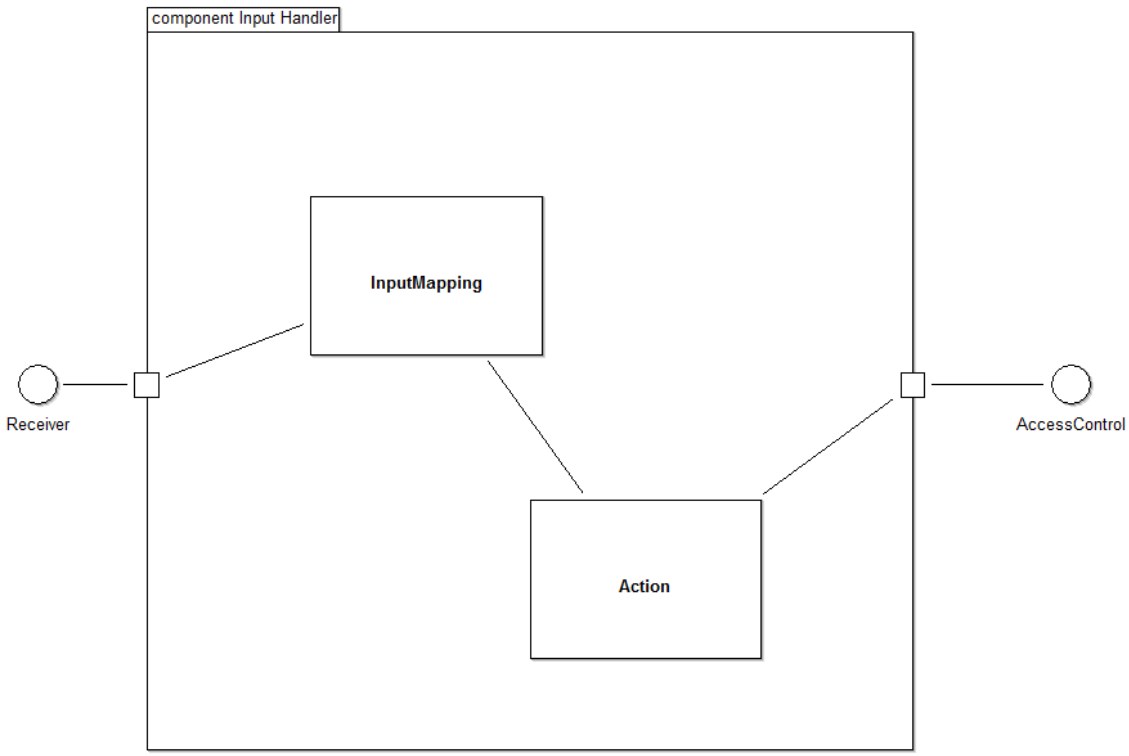
3.3.1.2 Cross References

- *Jeder Use-Case macht Gebrauch von dieser Komponente.*

3.4. Architecture for Component *Input Handler*

3.4.1.1. Design Considerations

Der „Input Handler“ fungiert als Mittler zwischen dem Nutzer und der Roboter-Einheit. Seine Aufgabe besteht darin, die Nutzer-Eingabe als Befehle zu interpretieren und diese dann an die Maschine weiterzuleiten.



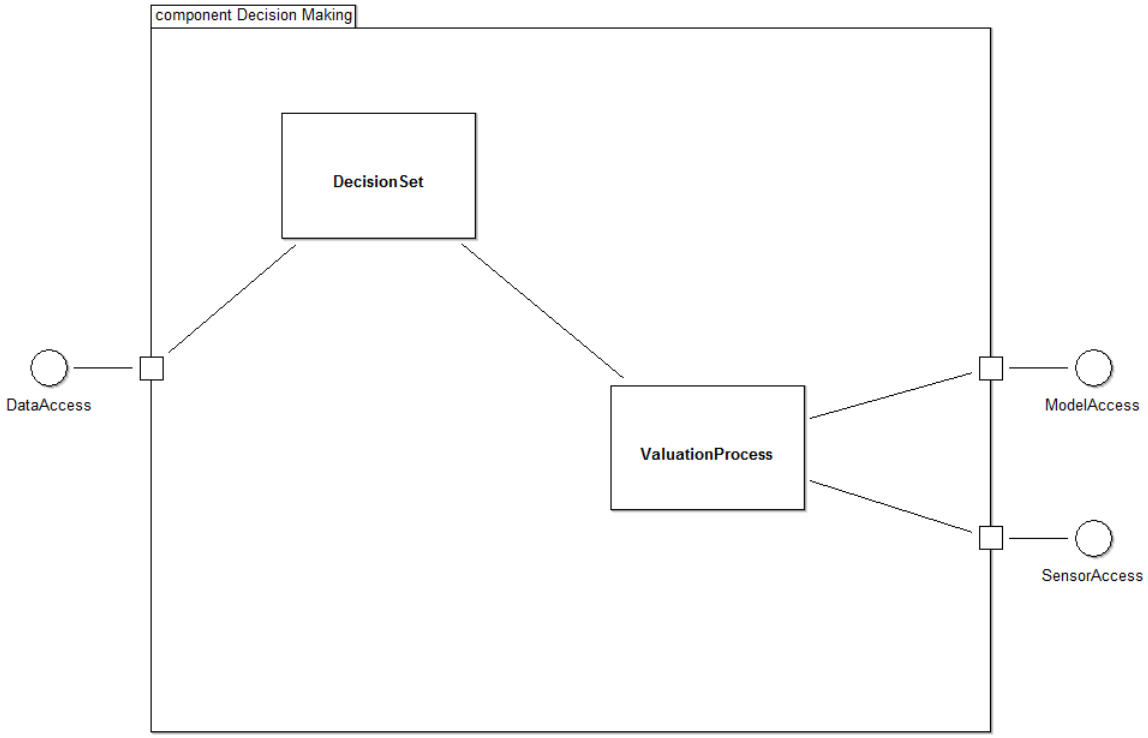
3.4.1.2. Cross References

- *Jeder Use-Case macht Gebrauch von dieser Komponente.*

3.5. Architecture for Component *Decision Making*

3.5.1.1. Design Considerations

Da der Roboter autonom Aufgaben lösen wird, ist eine Menge von Algorithmen notwendig, um die aktuelle Situation beurteilen zu können sowie anhand dieser Bewertungen Entscheidungen treffen zu können. Als „Schaltzentrale“ dienen hierbei die Klassen der Komponente „Decision Making“, welche die jeweils spezifische Anwendungslogik zur Lösung einer Aufgabe beinhalten.



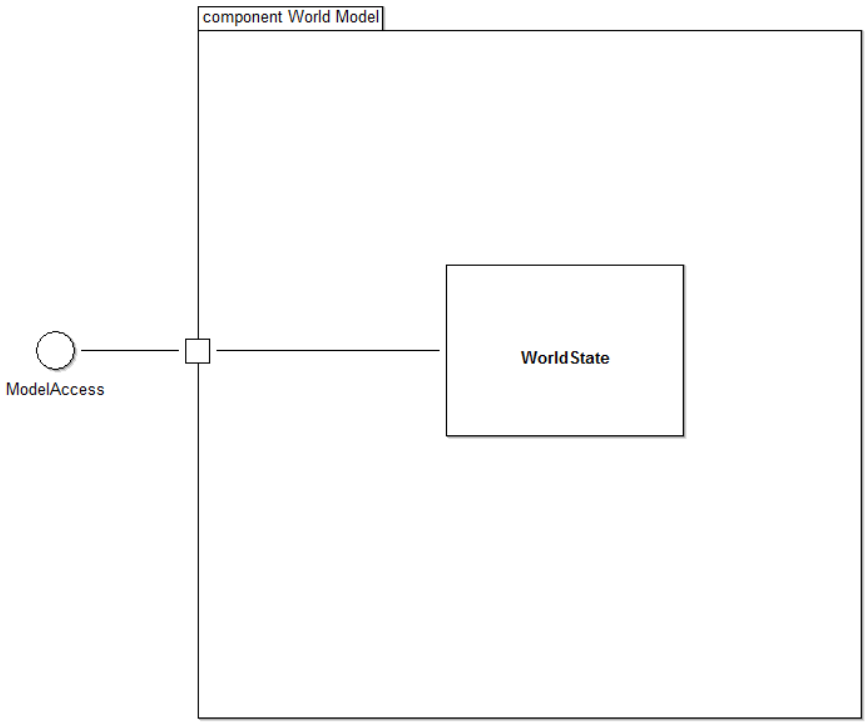
3.5.1.2. Cross References

- *Jeder Use-Case macht Gebrauch von dieser Komponente.*

3.6. Architecture for Component *World Model*

3.6.1.1. Design Considerations

Die Entscheidungen des Roboters beruhen nicht alleine auf der aktuell ausgewerteten Situation, sondern nutzen auch „Erfahrungen“, welche die Maschine seit Beginn des Problem-Lösungs-Prozesses gemacht hat.



3.6.1.2. Cross References

- *Jeder Use-Case macht Gebrauch von dieser Komponente.*

4. Project Management

4.1. Milestones and Schedules

Unsere erste Milestone ist natürlich eben dieses Design and Requirements Dokument in dem wir unsere Ziele und die Anforderungen des Projekts beschreiben. Davor gab es auch schon einige Schritte die dem Projektstart vorausgegangen sind.
Als erstes fanden wir uns als Gruppe zusammen und haben entschieden gemeinsam an einem Projekt zu arbeiten.
Nachdem die Projekte präsentiert wurden haben wir intensiv diskutiert welches Projekt wir nehmen sollen.
Als die Anmeldung zu den Projekten eröffnet wurde haben wir versucht gemeinsam in eins der ausgewählten Projekte zu kommen.
Nach der Anmeldung hatten wir unser erstes Treffen mit dem Projektleiter und einem Experten auf dem Gebiet. Danach trafen wir uns nochmal um mehr über die Details des Projektes zu erfahren. Dann haben wir den Roboter abgeholt und weiter an den Requirements und Design Dokument gearbeitet.
Unsere weiteren Milestones werden sein:

- M1 Roboter zusammenbauen 24.11.16
  Zuerst müssen wir den Roboter korrekt nach dem Prototyp im Labor nachbauen.
- M2 Setting up ROS 28.11.16
  Nachdem der Roboter steht müssen wir das System für ROS aufsetzen und es auf unseren Roboter konfigurieren damit wir mit der Implementierung beginnen können.
- M3 Auswertung 1.12
  Nachdem wir das System aufgesetzt haben müssen wir unser Design und Requirements Dokument überarbeiten aufgrund unserer bisherigen Erkenntnisse.
- M4 ROS package designen 12.12
  Nachdem wir unsere Use-Cases angepasst haben werden wir mit der Implementierung beginnen. Die Implementierungsphase wird ihre eigenen iterativen Milestones haben nach Vorbild iterativer Software Entwicklung.
  - Design
  - Test
  - Implementierung
- M5 Re-Evaluation 16.12.
  Nach der Implementierung müssen wir wieder unser Design and Requirements nach unseren Erkenntnissen anpassen. Außerdem Abgabe von M2 auf Moodle.
- M6 Refine ROS package 16.01.17
  Nun werden wir unserer Implementierung den letzten Feinschliff geben mit den gleichen Iterationen wie in M4
- M7 Project Deadline 23.01.17
  Am Ende werden wir unseren Roboter präsentieren können und dieser wird hoffentlich unsere Anweisungen ohne Murren korrekt ausführen.

4.2. Planned Effort per Person

Da keiner in unserem Team Erfahrungen mit derlei Projekten hat wird die Arbeitsaufteilung voraussichtlich sehr gleichmäßig aufgeteilt werden sein da sich jeder von uns sowieso mal mit dem Thema vertraut machen muss. Auch den Roboter werden wir gemeinsam zusammenbauen damit jeder genau weiß wie er funktioniert und was er kann.
Bei der Software Entwicklung wird sich wohl eine Arbeitsteilung entwickeln da wir ja nur einen Computer brauchen auf dem die Entwicklungsumgebung für ROS läuft. Die Programmierung wird dann auch von allen zu gleichen Teilen stattfinden.
Es ist schwer genaue Angaben zum Zeitaufwand zu treffen da wir selber erst den Umfang des Projekts erahnen können.