Kooperative Steuerung von Modellversuchsfahrzeugen

Entwicklung eines intelligenten Fahrentscheiders

Softwareentwicklungspraktikum
Sommersemester 2008

Pflichtenheft

Auftraggeber
Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund

Prof. Dr.-Ing. Lars Wolf
Mühlenpfordstrasse 23
38106 Braunschweig

Betreuer: Kai Homeier, Carina Flämig

Auftragnehmer:

Name	E - Mail
Wira Kakar	wirak@web.de
Chen Zhiwei	czwnii@hotmail.com
Rayan Merched El Masri	rayan_masri@hotmail.com
Ekrem Enes Duman	dumanenes@hotmail.de
Günter Dusch	g.dusch@Googlemail.com

Braunschweig, 18.04.2008

Versionsübersicht

Version	Datum	Autor	Status	Kommentar
1	17.04.2008	Alle Auftragnehmer	akzeptiert	Erste Version abgeschlossen

Inhaltsverzeichnis

1 Zielbestimmung

In Anbetracht der aktuellen Bedeutung und weltweiten Forschung und Entwicklung in der autonomen Robotertechnologie, die sich noch in den Anfängen befindet, ist es von großer Wichtigkeit für den Innovationsstandort Deutschland auch in diesem mit hohem Potential versehenem Forschungsbereich mitzuhalten und Akzente zu setzen.

Hinzu kommt, dass es seit Menschen gedenken ein Wunschtraum ist autonom arbeitende Maschinen zu erschaffen, die Arbeiten und Leistungen erbringen, die zu gefährlich, unzugänglich, mühsam oder undurchführbar für den Menschen sind. Hier wollen wir einen Beitrag leisten.

Ziel dieses Projektes ist die Erschaffung eines Softwaresystems, das einem sich fortbewegenden Roboter mit diversen Sensoren ermöglicht ein gesuchtes Objekt – in diesem Fall einen Ball - in einem vorher unbekannten Labyrinth zu finden. Die zusammengebaute fertige Hardware wird vom Institut bereitgestellt, sodass es unsere Aufgabe ist diese leere Hülle mit „Leben“ zu füllen, damit die Problematik des Suchens und Findens möglichst effizient gelöst werden kann.

1.1 Musskriterien

- Identifikation von Baken zur Navigation
- Kamerabasierte Identifikation des gesuchten Objekts
- Adäquate Fortbewegung innerhalb des Labyrinths
- Speicherung der bereits abgesuchten Wege und Kreuzungen
- Auffinden des gesuchten Objekts
- Positionsberechnung anhand der aufgestellten Baken

1.2 Wunschkriterien

- Erstellung einer Karte des Labyrinths
- Auffinden des gesuchten Objektes in minimaler Zeit
- Effizienzsteigerung durch den Einsatz mehrerer autonomer Fahrzeuge, die miteinander kommunizieren und so Synergien bilden

1.3 Abgrenzungskriterien

- Der Arbeitsbereich dieses Roboters ist ein Labyrinth, kein anderes Terrain

2 Produkteinsatz

Der Einsatzbereich dieses Systems ist das Modellversuchsfahrzeug des IBR, wobei der Aufgabenbereich dieses Fahrzeugs ein willkürliches Labyrinth ist. Dieses Projekt stellt unter Anderem ein Grundlagensystem in der Forschung der Roboterprogrammierung dar, das für zukünftige Systeme für unterschiedliche Anforderungen die Basis bildet.

2.1 Anwendungsbereiche

Zu den zukünftigen möglichen Anwendungsbereichen gehören:

- Suchen und Finden von Elementen/Objekten auf fremden Planeten, wobei ein autonomes Fahrzeug eine Notwendigkeit darstellt, da eine direkte Steuerung aufgrund der enormen Distanz ungeeignet ist (Aufgabenbereich in der Raumfahrt)
- Erkundungsmissionen in für den Menschen unzugänglichen Gebieten z.B. die Erforschung von einsturzgefährdeten Höhlen oder extrem kleinen unerforschten Hohlräumen in antiken Bauten wie Pyramide (Aufgabenbereich in Forschung)
- Suchen und Finden von Verstopfungen und starken Verschmutzungen in Kanalisationen (wirtschaftlicher Aufgabenbereich)
- Suchen und Finden von Landmienen in für den Menschen zu gefährlichen Gebieten (militärischer Aufgabenbereich)
- Suchen und Finden von verletzten Soldaten auf dem Schlachtfeld (militärischer Aufgabenbereich)

2.2 Zielgruppen

Das Softwaresystem ist primär für Studenten an der technischen Universität Braunschweig als Lernmöglichkeit gedacht, in dem praktisch das erworbene Wissen der ersten vier Semester angewendet werden kann. Dieses System ist sekundär für Programmierer und Entwickler als Ausgangssystem zu verstehen, um nach den jeweiligen Bedürfnissen und Anforderungen einen erfolgreichen Suchen-und-Finden Vorgang zu ermöglichen und so ausbauen zu können, dass es in anderen Anwendungsbereichen Lösungen liefert.

2.3 Betriebsbedingungen

- Die physikalische Umgebung des Systems ist ein Labyrinth in einem beliebigen Raum.
- Die Betriebszeit beläuft sich auf mehrere Minuten bis Stunden je nach dem wie komplex und große das zu bearbeitende Gebiet des Systems ist und die Stromversorgung gewährleistet wird. Abhängig vom zukünftigen Einsatzgebiet soll auch ein Dauerbetrieb, jedoch nur als Erweiterungsmöglichkeit für externe Programmierer und Entwickler, ermöglicht werden.
- Eine dauerhafte Beaufsichtigung des Systems ist nicht zwingend erforderlich, aber durch die Anforderungen von speziellen Einsatzmöglichkeiten möglich.

3 Produktübersicht

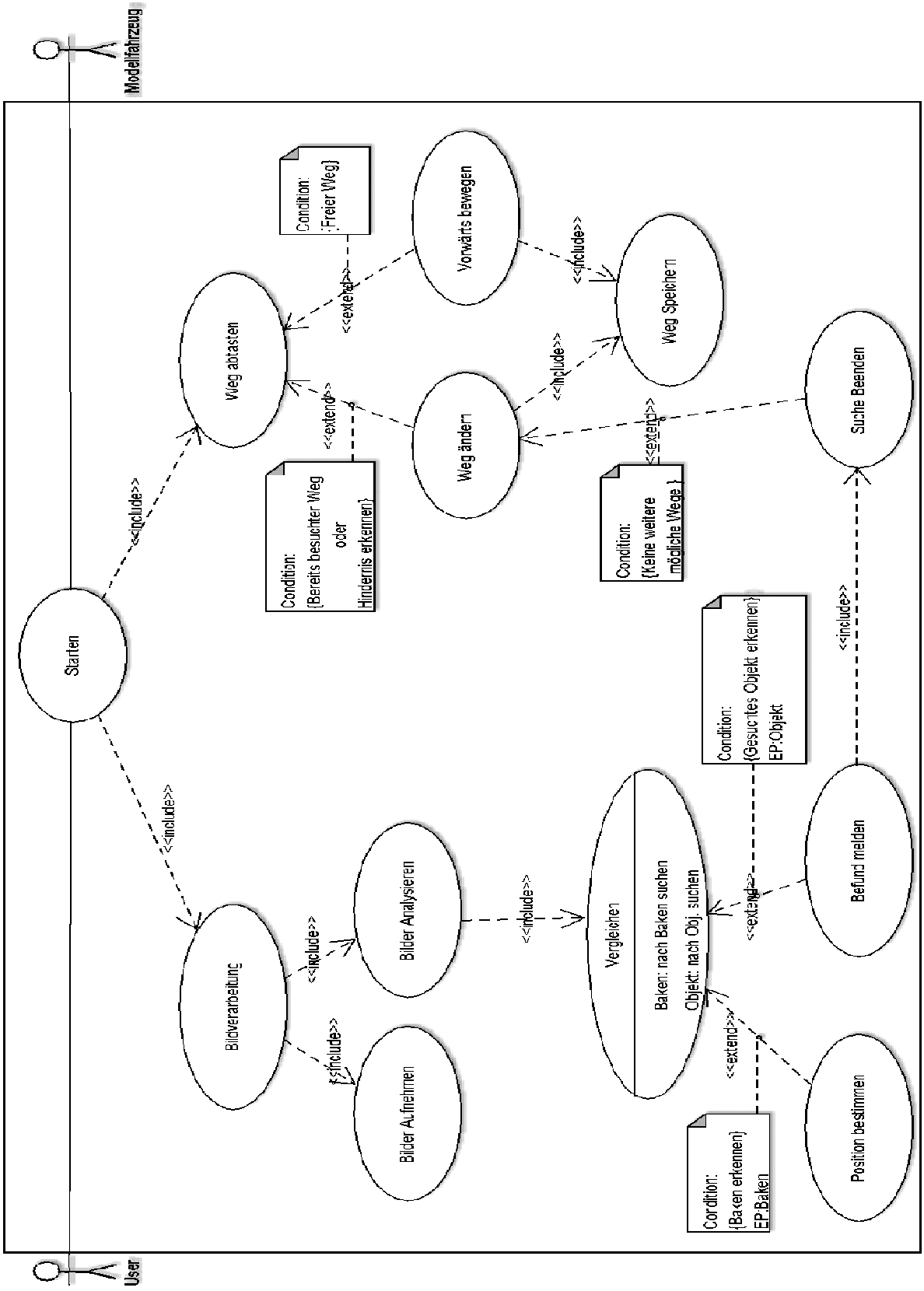

Abbildung 1: Use-Case-Diagramm des autonomen Modellversuchsfahrzeugs

4 Produktfunktionen

/F100/

Geschäftsprozess: Starten

Ziel: Das Gesamtsystem muss eingeschaltet werden.

Vorbedingung: Software, Modellversuchsfahrzeug, und die Kamera müssen eingeschaltet sein, und die Baken aufgestellt sein.

Nachbedingung Erfolg: Die optische Bearbeitung startet.

Nachbedingung Fehlschlag*:* Die optische Bearbeitung startet nicht.

Akteure: User, Modellversuchsfahrzeug

Auslösendes Ereignis: Befehl vom Softwaresystem.

Beschreibung:
1. Start aufrufen
2. Dem Modellversuchsfahrzeug den Startbefehl übergeben

/F200/

Geschäftsprozess: Bildverarbeitung

Ziel: Bilder aufnehmen und Bilder analysieren

Vorbedingung: Das Starten muss erfolgreich sein und die Kamera Aufnahme bereit.

Nachbedingung Erfolg: Bilder aufgenommen.

Nachbedingung Fehlschlag*:* Bilder nicht aufgenommen.

Akteure: Modellversuchsfahrzeug

Auslösendes Ereignis: Das Starten verursacht, dass die Kamera aufnimmt.

Beschreibung: 1. Kamera nimmt die Position des Modellversuchsfahrzeug s auf
2. Die Aufgenommenen Bilder werden dem Softwaresystem übergeben

Erweiterung:

2a Zur Sicherung, speichert das Modellversuchsfahrzeug eigenständig die Bilder

/F300/

Geschäftsprozess: Bilder aufnehmen

Ziel: Bilder aufnehmen und speichern der Bilder

Vorbedingung: Kamera Aufnahmebereit

Nachbedingung Erfolg: Bilder aufgenommen

Nachbedingung Fehlschlag*:* Bilder nicht aufgenommen

Akteure: Modellversuchsfahrzeug

Auslösendes Ereignis: Die optische Bearbeitung gibt den Befehl für die Aufnahme der Bilder an

Beschreibung: 1. Kamera reagiert auf den Befehl
2. Kamera nimmt Bilder auf und übergibt diese dem Softwaresystem zur Speicherung

/F400/

Geschäftsprozess: Bilder analysieren

Ziel: Gesuchtes Objekt finden, Baken erkennen

Vorbedingung: Bilder erfolgreich abgespeichert

Nachbedingung Erfolg: Erkennung von Farbe und Form auf den Bildern

Nachbedingung Fehlschlag*:* Keine oder falsche Erkennung von Farbe und Form auf den Bildern

Akteure: Modellversuchsfahrzeug

Auslösendes Ereignis: Nach dem Aufnehmen der Bilder wird sofort die Funktion „Bilder analysieren“ aufgerufen

Beschreibung: 1. Bilder sind aufgenommen
2. Das Softwaresystem analysiert die Bilder

/F500/

Geschäftsprozess: Vergleichen

Ziel: Das gesuchte Objekt muss gefunden werden

Vorbedingung: Die Bildanalyse wurden auf das Objekt oder die Baken angewendet.

Nachbedingung Erfolg: Gesuchtes Objekt gefunden

Nachbedingung Fehlschlag*:* Gesuchtes Objekt nicht gefunden

Akteure: Modellversuchsfahrzeug

Auslösendes Ereignis: Finden eines Objekts

Beschreibung: Vergleichen der Bilder des gefundenen Objekts mit den Daten des gesuchten Objekts (Form/Farbe)

/F510/

Geschäftsprozess: Befund melden

Ziel: Befundmeldung des gesuchten Objekts (keine Baken)

Vorbedingung: Objekt gefunden mit einem Vergleich

Nachbedingung Erfolg: Suche beenden

Nachbedingung Fehlschlag*:* Es wird weiterhin nach dem Objekt gesucht

Akteure: Modellversuchsfahrzeug

Auslösendes Ereignis: Das Finden eines Objekts

Beschreibung: 1. Objekt gefunden
2. Befund melden durch eine Meldung auf dem Display
3. Suche beenden

/F520/

Geschäftsprozess: Position bestimmen

Ziel: Position des Modellversuchsfahrzeug s bestimmen

Vorbedingung: Baken gefunden

Nachbedingung Erfolg: Position des Modellversuchsfahrzeug s wird bestimmt

Nachbedingung Fehlschlag*:* Position des Modellversuchsfahrzeug s wird nicht bestimmt

Akteure: Modellversuchsfahrzeug

Auslösendes Ereignis: Nach dem Finden der Baken

Beschreibung: 1. Baken gefunden

2. Position des Modellversuchsfahrzeug s wird bestimmt

/F600/

Geschäftsprozess: Weg abtasten

Ziel: Den besuchten Weg, das Hindernis und den freien Weg abtasten

Vorbedingung: Es muss erfolgreich gestartet worden sein

Nachbedingung Erfolg: Eine richtige Entscheidung treffen (einen freien unbesuchten Weg wählen)

Nachbedingung Fehlschlag*:* Einen bereits besuchten Weg wählen oder gegen ein Hindernis fahren

Akteure: Modellversuchsfahrzeug

Auslösendes Ereignis: Das Starten des Systems

Beschreibung: 1. Die Sensoren messen den Abstand zwischen dem Modellversuchsfahrzeug und dem Hindernis
2. Der abgetastete Weg wird mit den gespeicherten Wegen verglichen

/F610/

Geschäftsprozess: Weg ändern

Ziel: Einen freien -und unbesuchten Weg wählen

Vorbedingung: Ein freier und unbesuchter Weg muss existieren

Nachbedingung Erfolg: Richtung ändern und vorwärts fahren

Nachbedingung Fehlschlag*:* Suche wird beendet, weil alle Wege besucht worden sind

Akteure: Modellversuchsfahrzeug

Auslösendes Ereignis: Ein Hindernis oder ein besuchter Weg

Beschreibung: 1. Weg wird abgetastet

2. Es wird geprüft, ob bereits besucht oder Hindernis

3. Wenn kein Hindernis und nichtbesuchter Weg, dann Richtung ändern und vorwärts fahren

/F620/

Geschäftsprozess: Vorwärts bewegen

Ziel: Vorwärts bewegen

Vorbedingung: Es muss ein freier Weg vorhanden sein

Nachbedingung Erfolg: Modellversuchsfahrzeug bewegt sich nach vorne, weil ein freier Weg existiert

Nachbedingung Fehlschlag*:* Modellversuchsfahrzeug bewegt sich nicht nach vorne, weil es ein Hindernis gibt oder das gesuchte Objekt gefunden wurde.

Akteure: Modellversuchsfahrzeug

Auslösendes Ereignis: Das Vorhandensein eines freien Weges führt dazu, dass sich das Modellversuchsfahrzeug vorwärts bewegt

Beschreibung: 1. Ein freier Weg wird abgetastet
2. Modellversuchsfahrzeug bewegt sich vorwärts

/F700/

Geschäftsprozess: Weg speichern

Ziel: Den zurückgelegten Weg speichern

Vorbedingung: Das Modellversuchsfahrzeug muss sich bewegt haben und eine bestimmte Strecke zurückgelegt haben

Nachbedingung Erfolg: Das Modellversuchsfahrzeug hat eine Strecke zurückgelegt und diese Strecke wird gespeichert

Nachbedingung Fehlschlag*:* Das Modellversuchsfahrzeug legt keine Strecke zurück und es gibt keine Strecke, die gespeichert werden kann

Akteure: Modellversuchsfahrzeug , Softwaresystem

Auslösendes Ereignis: Wenn das Modellversuchsfahrzeug sich bewegt

Beschreibung: 1. Modellversuchsfahrzeug fährt los und legt eine Strecke zurück
2. Die zurückgelegte Strecke wird gespeichert

/F800/

Geschäftsprozess: Suche Beenden

Ziel: Die Suche beenden

Vorbedingung: Das Modellversuchsfahrzeug muss das gesuchte Objekt gefunden haben oder alle mögliche Wege besucht haben.

Nachbedingung Erfolg: Nach dem Finden des Objekts wird die Suche beendet

Nachbedingung Fehlschlag*:* Das Objekt wird nicht gefunden und das Modellversuchsfahrzeug fährt noch

Akteure: Modellversuchsfahrzeug , Softwaresystem

Auslösendes Ereignis: Das gefundenes Objekt

Beschreibung:
1. Modellversuchsfahrzeug fährt los und legt eine Strecke zurück
2. Die zurückgelegte Strecke wird gespeichert
3. Modellversuchsfahrzeug findet das gesuchte Objekt
4. Beenden

5 Produktdaten

/**D10**/ Daten der Ortung

. Position des Modellversuchsfahrzeuges
. Karte
. Baken

/**D20**/ Daten der Bildverarbeitung

. Kamerabilder

6 Produktleistungen

/**L10**/ Sensorik

Zwischen einem Event der Software und der Sensoren muss eine Reaktionszeit von max. 100 Millisekunde gewährleistet werden.

/**L20**/ Die Suche erfolgt effizient.

/**L30**/ Das Modellversuchsfahrzeug soll sich von einer beliebigen Startposition aus Orientieren können.

/**L40**/ Das Modellversuchsfahrzeug muss das gesuchte Objekt finden.

/**L50**/ Das Modellversuchsfahrzeug muss die Hindernisse erkennen und darf mit denen nicht kollidieren.

7 Qualitätsanforderungen

	sehr wichtig	wichtig	normal	nicht relevant
Richtigkeit	**X**			
Zuverlässigkeit		**X**		
Robustheit		**X**		
Leistung			**X**	
Benutzerfreundlichkeit				**X**
Überprüfbarkeit	**X**			
Wartbarkeit	**X**			
Reparierbarkeit	**X**			
Entwicklungsfähigkeit	**X**			
Wiederverwendbarkeit	**X**			
Portabilität	**X**			
Verständlichkeit		**X**		
Interoperabilität	**X**			
Produktivität		**X**		
Pünktlichkeit	**X**			
Sichtbarkeit	**X**			

8 Benutzeroberfläche

Es wird eine Benutzeroberfläche und eine Hardware-Schnittstelle definiert.
Da es hier um eine Systemsoftware und nicht um eine Anwendungssoftware geht, werden wir keine Benutzeroberflächen, im Sinne einer grafischen Oberfläche, erstellen.

/**B10**/ USB-Kernel-Schnittstelle die das Modellversuchsfahrzeug mit dem Entwicklungsrechner verbindet um die Übertragung von Java-Quellcode zu ermöglichen.
/**B20**/ Das Display, als Benutzerschnittstelle, hat die Aufgabe einfache Meldungen des Programms anzuzeigen in bestimmten Fällen, wie z.B. das Ergebnis der Suche (sowohl positive als auch negative Suchergebnisse).

9 Nichtfunktionale Anforderungen

Aussagen über nichtfunktionale Anforderungen können in solch frühem Stadium des Projektes schwer gemacht werden. Aus diesem Grund können die wenigen nichtfunktionalen Anforderungen jetzt nur erfolgen.

- . /NF10/Das Produkt ist mit geringem Aufwand weiterentwickelbar und wartbar
- . /NF20/Das Produkt ist benutzerfreundlich
- . /NF30/ Das Produkt ist plattformunabhängig

10 Technische Produktumgebung

10.1 Software

- Betriebssystem Linux
- Java JDK 1.5 oder höher
- Subversion
- TortoiseSVN (SVN Client für Linux und Windows)
- WxWidgets
- Microsoft Visio
- argoUML

10.2 Hardware

- Mini-ITX Express Motherboard
  - CPU: Support VIA Eden 1GHz Processor, Package type : 400
  - Memory: 1 * 240 -pin DIMM Sockets for unbuffered DDR2 533 SDRAM up to 1 GB
  - Chipset: VIA CN700 + 8237RP Chipset.
  - PCI enhanced IDE: 4 * Ultra DMA 133 / 100 / 66 IDE Devices Support.
  - Serial ATA Interface: 2 Serial ATA HDDs with RAID 0.1 Function Support
  - VGA Interface: VIA Unichrome Pro Processor.
  - Video Memory: Share Memory up to 128MB.
  - Audio Interface: VIA VT1617A 6-Channel AC 97' Audio CODEC.
  - LAN Interface: VIA VT6103CL 10 / 100 PCI LAN PHY
  - Extended Interface: 1 * 32-bit PCI Slot.
    1 * PCI 120-Pin Adapter Connector With Expansion Daughter Boards.
  - Internal I/O Port: 3 * High Speed USB Connectors @ 480 Mbit / s for 6 USB 2.0 Ports
    CPU / Chassis Fan Connectors / 1 * 26-pin Parallel Connector
    9-Pin COM2 Connector / 1 * 24 -pin ATX Power Connector .
  - External I / O Port: 2 * High Speed USB Connectors @ 480 Mbit/ s.
    1 * Serial Port / 1 * D-Sub 15-pin VGA Connector.
    1 * PS / 2 Mouse & 1 * PS/2 Keyboard / 1 * RJ45 Connector / 1 * Audio I / O
  - BIOS: Award 4 MB Flash ROM
  - CD / AUX AUDIO IN & SPEAK Out.
  - Power Requirement: standard 24 -Pin ATX Power supply .

- Videosensor: Webcam

- Abstandssensor
- Restliche nichtelektronische Bestandteile des Fahrzeugs

10.3 Orgware

Organisatorische Rahmenbedingungen werden in folgenden festgehalten:

- Bereitstellung der Softwareumgebung auf den Rechnern

10.4 Produktschnittstellen

- Kamera-Prozessor-Schnittstelle: Überträgt die Bilder von der Kamera auf den Prozessor übertragen.

- Sensor-Prozessor-Schnittstelle: Überträgt die gemessenen Werte von den Sensoren auf den Prozessor.

- Motor-Prozessor-Schnittstelle überträgt die Befehle vom Prozessor auf den Motor, um vorwärts zu fahren bzw. stehen zu bleiben.

- Räder-Prozessor-Schnittstelle überträgt die Befehle vom Prozessor auf die Räder, um links oder rechts abzubiegen bzw. gerade zu bleiben.

11 Glossar

- **Richtigkeit** [1]: Das Programm sei richtig, wenn es nach der Spezifikation der Funktionen verhält, die es unterstützen soll.
- **Zuverlässigkeit** [1]: Eine Software sei zuverlässig, wenn der Benutzer sich auf sie verlassen kann. Oder auch die Wahrscheinlichkeit dass die Software über eine bestimmte Zeitspanne wie erwartet funktioniert.
- **Robustheit** [1]: Die Software sei robust, wenn sie sich vernünftig, auch bei unerwarteten Situationen, verhält.
- **Leistung** [1]: Eine Software ist dann effizient, wenn sie Rechenressourcen, wie z.B. Rechenzeit oder Speicher, sparsam verwendet.
- **Benutzerfreundlichkeit** [1]: Eine Software sei benutzerfreundlich, wenn der Benutzer sie leicht bedienbar findet.
- **Überprüfbarkeit** [1]: Eine Software sei überprüfbar, wenn deren Eigenschaften leicht überprüft werden können.
- **Wartbarkeit** [1]: wird verwendet, um auf die Änderungen, die an einem Software-System nach dem ersten Release, zu beziehen.
- **Reparierbarkeit** [1]: Eine Software sei reparierbar, wenn sie die Korrektur von Fehlern innerhalb einer begrenzten Menge an Arbeit zulässt.
- **Entwicklungsfähigkeit** [1]: Eine Software muss in der Lage sein, neue Verwaltungs- und Organisationstechniken unterzubringen.
- **Wiederverwendbarkeit** [1]: Ist das Wiederverwenden von Komponenten an verschiedenen Stellen im Projekt.
- **Portabilität** [1]: Ist die Fähigkeit auf unterschiedliche Hardwareplattformen bzw. Betriebssysteme zu laufen.
- **Interoperabilität**[1]: Beschreibt die Fähigkeit eines Systems mit anderen Systemen zu koexistieren und zu kooperieren.

---

[1] Fundamentals of Software Engineering [1991]. Ghezzi, Jazayeri, Mandrioli.

- **Verständlichkeit**[1]: Eine Software sei verständlich falls sie ein vorhersagbares Verhalten hat.
- **Produktivität**[1]: Misst die Effizienz des Prozesses.
- **Pünktlichkeit**[1] : Die pünktliche Lieferung eines Produkts.
- **Sichtbarkeit**[1]: Eine Software sei sichtbar, falls alle Schritte und aktuellen Status deutlich dokumentiert sind.
- **mini-ITX**[2]**:** Ein Formfaktor für Computer-Mainboards, der kleiner als Mainboards aus der herkömmlichen ATX-Reihe.
- **WxWidgets**[3]**:** Ein auf C++ basierendes Open-Source-Framework zur plattformunabhängigen Entwicklung von Anwendungen mit grafischer Benutzeroberfläche(GUI).
- **JDK**[4]**:** Java Development Kit, abgekürzt JDK, ist die Java Entwicklungswerkzeuge.
- **Visio**[5]**:** Eine weit verbreitete Visualisierungs-Software von Microsoft für Windows. Visio dient dazu, mit Hilfe verschiedener Vorlagen mit passenden Werkzeugen und Symbolen grafische Darstellungen und Diagramme zu erzeugen.

---

[1] Fundamentals of Software Engineering [1991]. Ghezzi, Jazayeri, Mandrioli.

[2] http://de.wikipedia.org/wiki/Mini-ITX

[3] http://de.wikipedia.org/wiki/WxWidgets

[4] http://de.wikipedia.org/wiki/Jdk

[5] http://de.wikipedia.org/wiki/Visio