Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminar - Ausarbeit

WeSe2012/13

Florian Johannßen

Knowledge Sharing for Robots

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Florian Johannßen

Florian.Johannssen@haw-hamburg.de

Thema

Knowledge Sharing for Robots

Stichworte

Knowledge Sharing, Cloud Robotics, Cloud Computing, Nao, RoboEarth, ROS

Kurzzusammenfassung

Diese Ausarbeit zeigt einen Ansatz, wie Roboter mithilfe von Cloud Computing ihre Lernmechanismen verbessern und wie heterogene Roboter über Cloud Services Informationen von Plänen, Objekten und Umgebungen austauschen. Der Fokus dieser Arbeit liegt darin, das verfolgte Ziel der Masterthesis vorzustellen, welches die Idee des Knowledge Sharing for Robots umsetzt, indem mehrere humanoide Nao Roboter über den Cloud Service RoboEarth Informationen austauschen.

Florian Johannßen

Title of the paper

Knowledge Sharing for Robots

Keywords

Knowledge Sharing, Cloud Robotics, Cloud Computing, Nao, RoboEarth, ROS

Abstract

This paper deals with the subject area Knowledge Sharing for Robots. It demonstrates how robots can improve their learning mechanism with Cloud Computing and it shows how heterogeneous robots can exchange information about plans, objects and environments among each other. This work introduces the target of the master thesis, which implements the approach of Knowledge Sharing for Robots via Cloud Services with the aid of the humanoid robot Nao.

Inhaltsverzeichnis

Abbildungsverzeichnis 2
1 Einleitung 3
2 Aufbau der Arbeit 4
3 Masterarbeit 4
3.1 Zielsetzung 4
3.2 Vorarbeit 5
3.2.1 Nao Roboter Plattform 5
3.2.2 Robotic Operation System (ROS) 6
3.2.3 RoboEarth 6
3.2.4 CRAM System 7
3.2.5 Nao in the Cloud 9
3.3 Schnittstellenentwurf 10
3.4 Abgrenzung 11
4 Chancen und Risiken 11
5 Zusammenfassung 12
Literaturverzeichnis 13

Abbildungsverzeichnis

Abbildung 1: Nao 5
Abbildung 2: RoboEarth Architektur 7
Abbildung 3: CRAM System 8
Abbildung 4: Architektur 9
Abbildung 5: Transformation 10

1 Einleitung

Das Internet ist mittlerweile zu einem der wichtigsten Kommunikationsmedien geworden. Es gibt uns die Möglichkeit, Wissen global und mobil zu veröffentlichen und abzurufen. Es hilft uns unbekannte Aufgaben effizient zu lösen und Wissen mit anderen Menschen zu teilen. Wenn man sich im Research-Bereich der Robotik aufhält, so wünscht man sich, dieses Paradigma auf Roboter zu übertragen. Das Problem der Roboter ist es, dass sie in vielerlei Hinsichten limitiert sind. Die Rechenleistung und Hardwareressourcen sind durch Kosten und Platzprobleme begrentzt. Des Weiteren sind Roboter meist auf sich alleine gestellt und nur auf ein bestimmtes Aufgabengebiet programmiert, so dass ihr Verhalten stark durch unflexible Programmierung limitiert ist. Es ist nicht möglich, dass heterogene Roboter Informationen untereinander austauschen und verarbeiten können. Heutzutage sind Roboterhersteller wie *Aldebaran Robotics* und *Willow Garage* in der Lage, WLAN fähige und programmierbare Roboter mit abstrakten Schnittstellen auszuliefern. Die spezifischen Aufgaben, wie z. B. Gesichtserkennung, Wegplanung und Spracherkennung sind bereits hardwarenah und effizient gelöst. Dadruch sind die Voraussetzungen dafür geschaffen Roboter mithilfe des Internets miteinander zu verknüpfen.

An dieser Stelle wird der Begriff *Cloud Robotics* eingeführt, welcher seit kurzer Zeit von immer mehr Firmen und Forschungseinrichtungen wie (Kuffner 2011) und (Quintas, Menezes und Dias 2011) geprägt wird. Diese Idee sieht eine physikalische Trennung zwischen Hardware und Software vor. Die üblichen Hardwarekomponenten eines Roboters, wie z.B. Sensoren, Aktoren, Kameras und Lautsprecher befinden sich weiterhin auf dem Roboter. Was sich vom weitverbreiteten Ansatz, bei dem sich Software und Hardware auf einem Roboter befinden, unterscheidet, ist, dass das Gehirn des Roboters auf entfernte Server ausgelagert ist. Diese Überlegung, komplexe und rechenintensive Operationen auf entfernte Server auszulagern, wurde bereits in den 1990er Jahren vom Japaner Masayuki Inaba in seiner Arbeit (M. Inaba 1993) vorgestellt. Seine Idee war es, Roboter ohne Gehirn auszustatten. Da es in dieser Zeit keine Prozessoren gab wie heute, die leistungsstark und platzsparend zugleich sind, versuchte Inaba, die benötigten Hardwareressourcen durch dieses Vorgehen möglichst gering zu halten.

Beim Cloud-Robotics-Ansatz repräsentiert der Roboter den Client, der Dienste aus der Cloud-Infrastruktur beansprucht und die Server-Seite eine Cloud aus Servern, die gemeinsam die Funktionalität eines verteilten Systems erfüllen. Dieser Ansatz kann wie bei (M. Inaba 1993) dazu verwendet werden, rechenintensive Aufgaben auf leistungsstarke entfernte Server, auszulagern. Zusätzlich ergibt sich dadurch die Möglichkeit, dass sich Roboter mithilfe der Cloud untereinander verständigen, koordinieren und ihre Lernmechanismen durch Knowledge Sharing verbessern.

Die Idee des Knowledge Sharing for Robots umfasst das Problem, wie Roboter untereinander Informationen austauschen, damit sie von den Erfahrungen anderer profitieren können. Da die drahtlose Kommunikation durch sichere und schnellerer Kommunikation attraktiver geworden ist und viele Roboter über IEEE 802.11 standardisierte WLAN Schnittstellen verfügen, ist es zu einem interessanten Research-Bereich geworden.

2 Aufbau der Arbeit

Diese Arbeit beginnt mit einer Einführung, in der die Idee beschrieben wird, die beiden Gebiete Cloud Computing und Robotik, miteinander zu verknüpfen. Im Kapitel *3* wird insbesondere darauf eingegangen, welches Ziel diese Masterthesis verfolgt. Des Weiteren wird erläutert, welche Vorarbeiten geleistet worden sind, welche Werkzeuge und Systeme eingesetzt werden, um einen Wissensaustausch zwischen Robotern zu realisieren. In den Kapiteln *4* und *5* werden Abgrenzungen festgelegt, sowie die Chancen und Risiken der Arbeit beschrieben. Abschließend erfolgt eine kurze Zusammenfassung.

3 Masterarbeit

Dieses Kapitel stellt das konkrete Ziel der Masterarbeit vor. Es wird beschrieben welche Vorarbeiten gleistet worden sind und welche Hilfssysteme und Frameworks eingesetzt werden. Des Weiteren wird die grobe Architektur vorgestellt, auf die der spätere praktische Teil der Arbeit aufbaut.

3.1 Zielsetzung

Diese Arbeit beschäftigt sich damit, wie heterogene Roboter untereinander Informationen über Objekte, Umgebungen und Pläne austauschen können. Wie in (Quintas, Menezes und Dias 2011) konzeptuell beschrieben, können Roboter durch dieses Vorgehen ihre Lernmechanismen erheblich steigern, da sie nicht nur auf sich alleine gestellt sind, sondern von den Erfahrungen anderer Roboter profitieren können.

Sofern die Roboter über WLAN-Schnittstellen mit einer Cloud verbunden sind, können sie ihre erlernten Fähigkeiten, wie die Erkennung eines Objektes oder die Umsetzung eines Plans, in der Cloud veröffentlichen und anderen Robotern zugänglich machen oder Informationen von der Cloud abrufen.

Das konkrete Ziel dieser Arbeit ist es, den noch wenig untersuchten und realsierten Ansatz des Lernens von Robotern, Knowledge Sharing for Robots via Cloud Computing, mithilfe des Nao Roboter und des Cloud Service RoboEarth vorzustellen und umzusetzen. Dabei umfasst der praktische Teil dieser Masterthesis sowohl die Implementierung einer Schnittstelle zwischen dem humanoiden Roboter Nao und dem Cloud Service RoboEarth, als auch die Umsetzung eines Szenarios, bei dem mehrere Nao Roboter Informationen vom Cloud Service herunterladen und ausführen.

3.2 Vorarbeit

Damit das Ziel der Masterarbeit erreicht werden kann, müssen bestimmte Vorarbeiten geleistet werden. Es musste eine Infrastruktur geschaffen werden, die es einem Roboter ermöglicht, mit der Cloud zu kommunizieren. Dieser Abschnitt stellt die Roboter-Plattformen, Bibliotheken, Cloud-Services und Programmiersprachen vor, die im Rahmen dieser Arbeit eingesetzt werden.

3.2.1 Nao Roboter Plattform

Nao wurde vom weltweit führenden Hersteller Aldebaran Robotics für humanoide Roboter (Parent 2011) entwickelt und befindet sich seit 2008 auf dem Markt. Nao ist nach (Aldebaran Robotics 2013) der am häufigsten eingesetzte humanoide Roboter im Bereich Bildung und Forschung und bietet sich auch aufgrund seiner sicheren und performanten Programmierplattform NaoQi an, die Idee des Knowledge Sharing for Robots zu verwirklichen.

Abbildung 1: Nao

NaoQi

Dieses Framwork ermöglicht eine High-Level Programmierung des humanoiden Roboters NAO. NAOqi ist nach (Aldebaran-Robotics 2012) modularisiert. Jedes Modul ist für einen bestimmten Aufgabenbereich zuständig, wie z.B. Spracherkennung, Gesichtserkennung, Wegplanung, Audioverarbeitung oder Text-To-Speech. Jede dieser Low-Level-Aufgaben wurde effizient und hardware-nah im inneren der Plattform implementiert. NaoQi bietet Anwendungsprogrammierern Schnittstellen, um das Verhalten des Nao auf einer hohen Abstraktionsebene zu programmieren.

3.2.2 Robotic Operation System (ROS)

Damit heterogene Roboter miteinander kommunizieren und Informationen austauschen können, wird eine Middleware benötigt, die von der Hardware der Roboter abstrahiert. Für diese Arbeit wird ROS verwendet und mit der Nao-Plattform verknüpft. ROS wurde nach (Morgan Quigley 2009) als ein Open-Source–Meta-Betriebssystem für Roboter entwickelt. Die Motivation hinter ROS war es, die Entwicklung von Roboterappilikationen einfacher und wiederverwendbarer zu gestalten und dafür geeignete abstrakte Schnittstellen anzubieten. Ein weiterer wichtiger Punkt, warum es sich gegenüber anderen Middleware wie Orocos oder Player durchgesetzt hat, ist, dass ROS beliebig durch Module, sogenannte Stacks, erweiterbar ist und von immer mehr Roboterplattformen unterstützt wird.

3.2.3 RoboEarth

Die Forschungsgruppe RoboEarth entwickelte nach (Zweigle 2009) ein Word Wide Web für Roboter. Es stellt einen verteilten Datenspeicher dar, mit dem die Roboter untereinander Informationen austauschen können. Zudem können Roboter vom Verhalten anderer Roboter lernen und ihren Lernmechanismus verbessern. Durch RoboEarth ist es möglich, dass Roboter heterogener Plattformen untereinander Informationen über Objekte, Aufgaben und Umgebungen austauschen können.

Die RoboEarth Architektur besteht, wie in Abbildung *2* ersichtlich, nach (Marco, et al. 2011) hauptsichtlich aus der Execution Engine, der verteilen Datenbank und dem Knowledge Processing Framework KnowRob. Ein Roboter sendet eine Anfrage, wie „Serve a drink“ im XML oder Json Format an die Execution Engine. Die Anfrage wird an die KnowRob Komponente delegiert, welche nach (Tenorth und Beetz 2010) für das Knowledge Processing zuständig ist. Durch KnowRob erfolgt eine semantische Suche in der RoboEarth Datenbank nach dem angeforderten Plan. Zusätzlich prüft sie, ob alle benötigen Informationen für die angeforderte Aufgabe vorhanden sind und lädt den Plan herunter. Solch ein Plan liegt in der RoboEarth Language vor und wird von der KnowRob Komponente zur Execution Engine weitergeleitet.

Die Informationen aus der RoboEarth-Datenbank sind in der abstrakten RoboEarth-Language definiert. Sie ist eine Web Ontology Language, um die Beziehungen zwischen den Objekten, Umgebungen und Plänen semantisch zu beschreiben. In der Execution Engine wird der Plan in das Format der CRAM Plan Language (Rittweiler 2010) übersetzt. Dieses Format wird benötigt, damit die abstrakten Informationen aus RoboEarth in ein ausführbares Format transformiert werden. Jeder teilnehmende Roboter muss das Robotic Operation System von (Quigley, et al. 2010) installiert haben. Die Bibliotheken von ROS enthalten entsprechende CRAM-Schnittstellen, damit die Pläne von RoboEarth auf dem Roboter ausgeführt werden können.

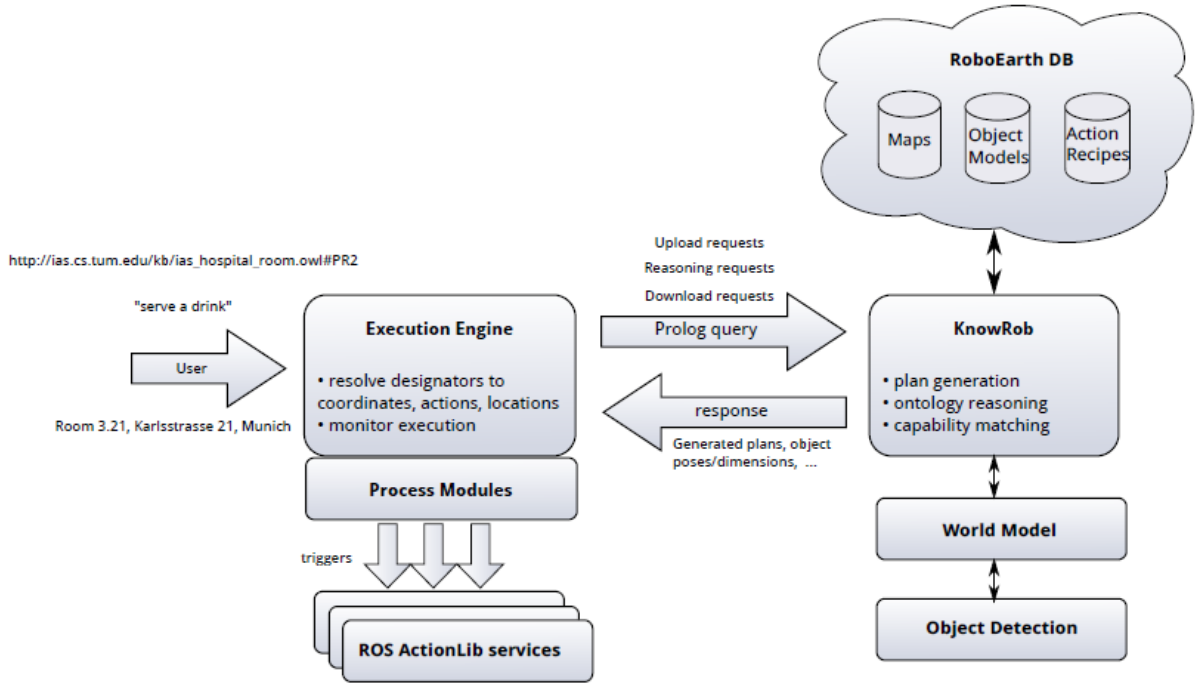

Abbildung 2: RoboEarth Architektur

RoboEarth verwendet die CRAM Plan Language und das Robotic Operation System. Dadurch sind Werkzeuge geschaffen, die bewerkstelligen, dass Roboter unterschiedlicher Hardware die RoboEarth-Plattform nutzen können.

3.2.4 CRAM System

Das Cognitive-Robot-Abstract-Machine-System ist nach (Michael Beetz 2010) ein modularisiertes System, welches die Entwicklung von High-Level Roboterapplikationen ermöglicht. Es besteht, wie in Abbildung *3* ersichtlich, hauptsächlich aus zwei Komponenten. Es bietet mithilfe der Lisp-basierten CRAM Plan Lanuage ein Hilfsmittel, um Roboterpläne abstrakt zu beschreiben. Des Weiteren beinhaltet es das Planungs- und Schlussfolgerungsmodul KnowRob, das für eine gegebene Aufgabe einen Lösungsweg berechnen kann.

KnowRob definiert eine Ontologie, welche speziell für die Serviceroboterdomäne entwickelt worden ist. Zum Beispiel ist es dadurch möglich, dass wenn der Roboter keine Milch entdeckt, er daraus schließt, dass Milch als Lebensmittel im Kühlschrank zu finden ist.

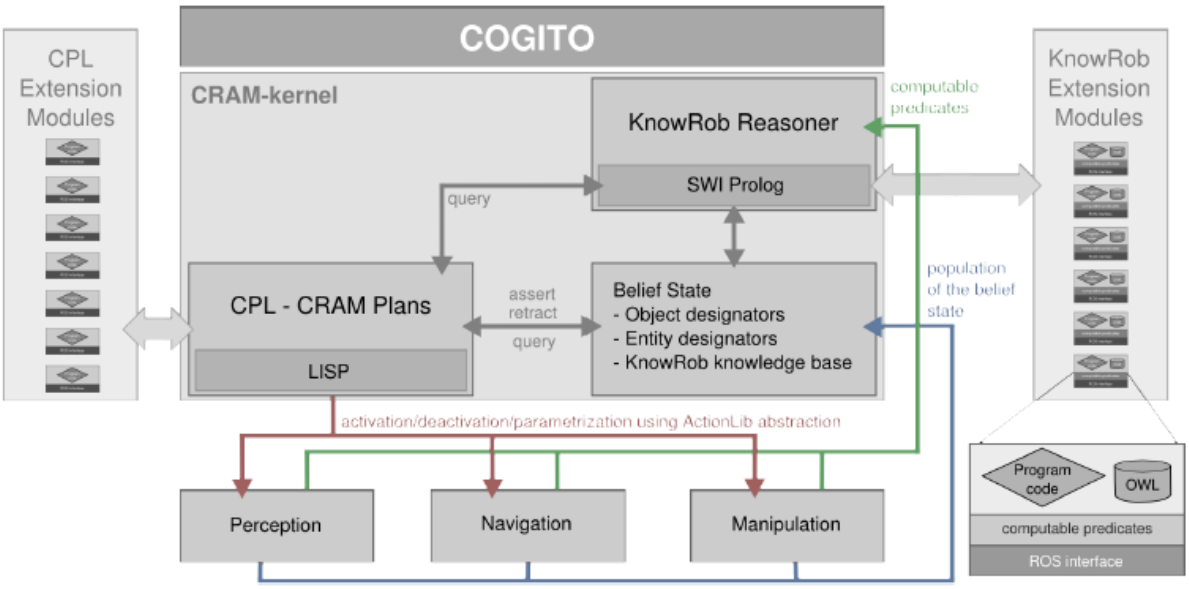

Abbildung 3: CRAM System

KnowRob bietet eine Prolog-basierte Abfrageschnittstelle. Der Nao Roboter kann die RoboEarth-Datenbank durch folgende Zeilen Code nach einen Plan durchsuchen:

```
re_download_action_recipe('Grasp Bottle', 'Nao', Recipe),
re_generate_cpl_plan(Recipe, CplPlan).
```

Bei diesem Beispiel findet eine Anfrag nach den Plan *Grasp-Bottle* statt. Befindet sich ein Plan in der Datenbank, so wird dieser von der RoboEarth Language in die CRAM Plan Language transformiert. Ein CRAM Plan, um eine Flasche zu greifen kann folgendermaßen aussehen:

```
(def-top-level-plan grasp-bottle ()
(with-designators ((fridge (object '((type fridge))))
                   (bottle-loc (location `((inside ,fridge))))
                   (bottle (object `((type bottle) (at ,bottle-loc)))))
  (setf fridge (perceive-object fridge))
  (achieve `(object-in-hand ,bottle :right))))
…
```

Diese Code-Zeilen zeigen das Ergebnis der KnowRob-Anfrage für den Plan *Grasp-Bottle*.

3.2.5 Nao in the Cloud

Die vorgestellten Frameworks, Bibliotheken und Plattformen werden benötigt, damit der Nao Roboter mit der Cloud kommunizieren kann. Die Abbildung *4* zeigt anhand eines Schichtenmodells, wo die einzelnen Komponenten angeordnet sind. Diese Komponenten befinden sich auf einem externen Rechner und bieten dem Roboter eine Schnittstelle zum Internet. Der Nao verbindet sich mit der Cloud über dieses System.

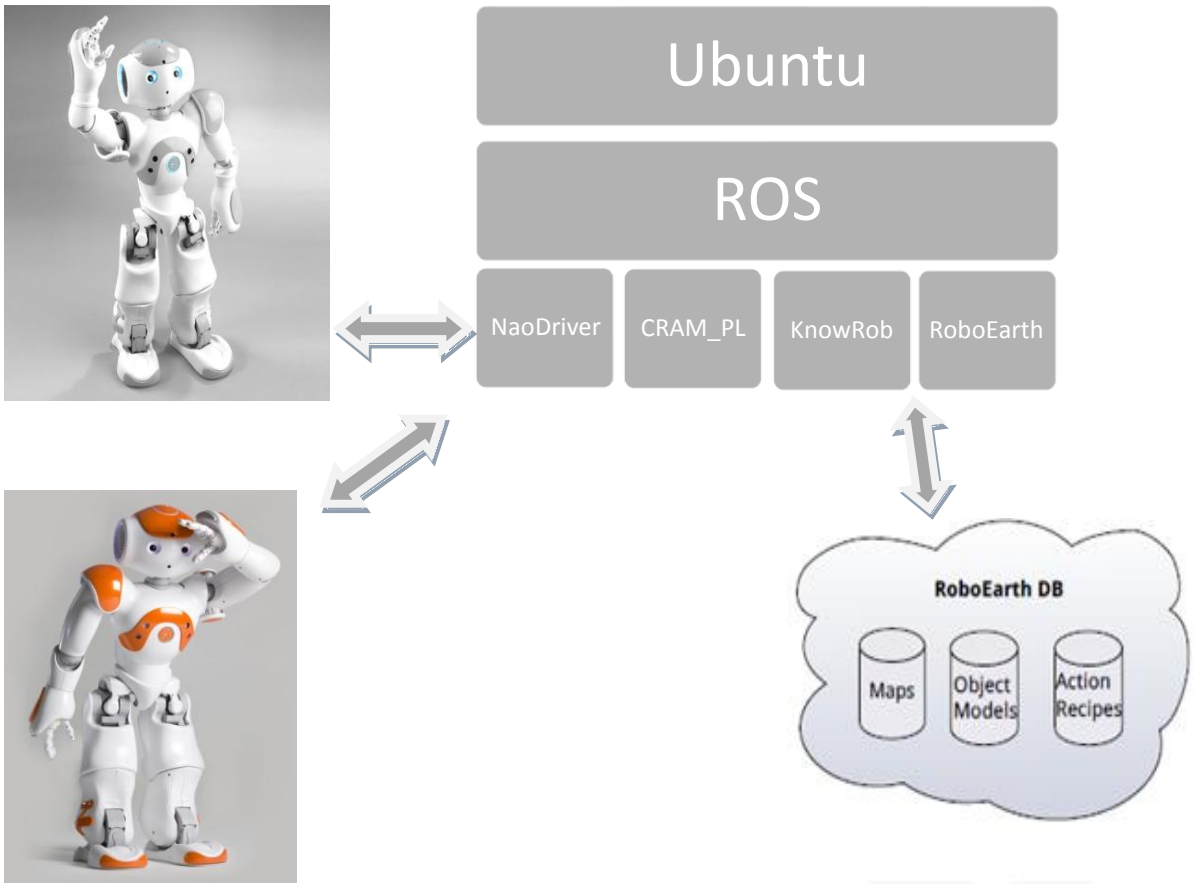

Abbildung 4: Architektur

Diese Architektur besteht zum einen aus dem Open-Source-Betriebssystem Ubuntu und zum anderen aus der bereits vorgestellten Middleware mit den entprechenden Erweiterungs-Stacks. Der NaoDriver repräsentiert den ROS-Stack für den Nao Roboter. Der NaoDriver bildet eine Abstraktionsebene über die Bibliotheken des NaoQi-Frameworks. Im Rahmen dieser Arbeit müssen die Schnittstellen und Interaktionen zwischen dem NaoDriver und den Stacks RoboEarth, KnowRob und CRAM_PL implementiert werden.

3.3 Schnittstellenentwurf

Der Implementierungsaufwand der Masterarbeit beinhaltet größtenteils die Realisierung der Schnittstelle zwischen dem Nao und dem Cloud Service RoboEarth. Hierbei wird zunächst nur das Szenario betrachtet, bei dem der Roboter Informationen aus der RoboEarth Datenbank herunterlädt und ausführt.

Wie in Abbildung *2* zu sehen, bietet RoboEarth Prozessmodule, welche als Schnittstelle zu den ROS-Stacks der Roboter fungieren. Diese Prozessmodule werden ebenfalls in der CRAM Plan Language implementiert und adaptieren sich an den entsprechenden ROS-Bibiliotheken, welche im CRAM_PL Stack der Architektur aus Kapitel *3.2.5*, enthalten sind.

In dieser Arbeit müssen die speziellen Prozessmodule für den Nao Roboter in der Lisp-ähnlichen Sprache CRAM Plan Language entwickelt werden. Abbildung *5* zeigt einen Entwurf, wie die abstrakten CRAM-Befehle in die Nao-spezifischen Anweisungen übersetzt werden, um eine Ausfühung der Pläne auf der Nao Plattform zu erreichen.

```
(def-top-level-plan grasp-bottle ()
(with-designators ((fridge (object '((type fridge))))
                   (bottle-loc (location `((inside ,fridge))))
                   (bottle (object `((type bottle) (at ,bottle-loc)))))
  (setf fridge (perceive-object fridge))
  (achieve `(object-in-hand ,bottle :right))))
…
```

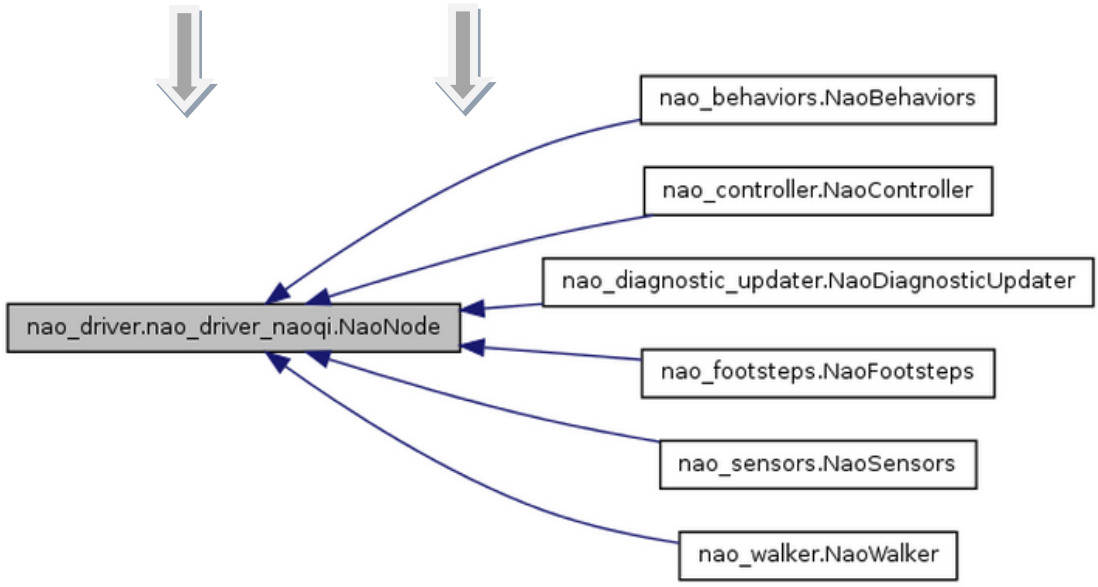

Abbildung 5: Transformation

Da der Nao verschiedene Module hat, wird die Schnittstelle ebenfalls modular entwickelt. Dies bedeutet, dass für die Module Vision, Manipulation und Navigation einzelne CRAM-Schnittstellen implementiert werden, welche die jeweiligen Aufgaben erfüllen.

3.4 Abgrenzung

Das Thema Knowledge Sharing for Robots hat eine sehr große Spannbreite. Es sind verschiedene Bereiche der Informatik involviert, wie Maschinelles Lernen, Wissensmanagement, Ontologien, Wissensverarbeitung, Cloud Computing und Robotik. Damit am Ende der Arbeit konkrete Szenarien durchgeführt werden können und etwas Ausführbares realisiert werden kann, werden einige Punkte vorerst nicht berücksichtigt, die sich aber für eventuelle Weiterentwicklungen dieser Masterthesis anbieten. Der Ansatz des Knowledge Sharing wird vorerst zwischen homogenen Robotern umgesetzt, da sich der Austausch von Plänen mit unterschiedlichen Hardware-Eigenschaften wesentlich komplexer gestaltet, als bei gleichartigen Robotern. Bei den eingesetzten Roboterplattformen wird es sich zunächst nur um die Nao-Plattform handeln.

Der Kern der Arbeit wird sich darauf beziehen, wie die Nao Roboter abstrakte Pläne aus der Cloud herunterladen und ausführen. Das Generieren und Hochladen von Informationen wird aus zeitlichen Gründen vorerst nicht berücksichtigt. Man geht davon aus, dass das Wissen, welches zwischen den Nao Robotern geteilt wird, sich bereits in der Cloud befindet und daher nicht von den Robotern generiert werden muss. Des Weiteren werden Ansätze, wie (Moritz Tenorth 2010), bei denen Instruktionen für Roboter aus Webseiten wie www.wikihow.com generiert werden, nicht berücksichtigt, da sie mit der Analyse und Übersetzung von natürlichen Sprachen in Roboterbefehle eine weiteres Forschungsgebiet eröffnen.

4 Chancen und Risiken

Wie im vorherigen Kapitel zu erkennen, besteht die Architektur aus mehreren Komponenten, die von verschiedenen Firmen und Instituten entwickelt worden sind. Dies führt dazu, dass diese Arbeit starke Abhängigkeiten von der Nao-Plattform, dem Cloud Service RoboEarth und den ROS-Stacks aufweist. Ein weiteres Risiko besteht darin, dass durch die Interaktion der einzelnen Komponenten Probleme bei den Schnittstellen und den Transformationen auftreten könnten. Dadruch, dass die Kommunktion zwischen Roboter und der Cloud über standardisierte Internetprotokolle wie dem Hypertext Transfer Protocol erfolgt, treten wesentlich höhere Latenzzeiten auf als bei einer Echtzeitkommunikation. Besonders durch die auftretenden Verzögerungen müssen gegebenfalls die Echtzeitanforderungen reduziert werden.

Diese Masterthesis behandelt einen neuen Ansatz, die Lernmeachanismen von Robotern zu verbessern. Dadurch ist es möglich, dass Roboter nicht nur auf sich alleine gestellt sind, sondern ihr Wissen mit anderen Robotern teilen können. Das Interessante dieser Arbeit ist es, dass es bisher keine Arbeiten gibt, die diesen Ansatz mit dem meisverkauftesten humoiden Roboter Nao umgesetzt haben. Der bisher entwickelte Appstore for Robots (Inbar 2011) bietet Programmierern die Möglichkeit, Roboterapplikationen zu veröffentlichen und herunterzuladen. Die Roboterporgamme sind hierbei plattformspezifisch und können daher nicht von heterogenen Robotern verwendet werden. Durch die eingesetzte Middlware ROS sind die Voraussetzungen dafür geschaffen, einen Wissensaustausch zwischen verschiedenartigen Robotern zu erreichen.

5 Zusammenfassung

Diese Arbeit gab eine Einführung in das Thema Knowledge Sharing for Robots und zeigte die Vorteile auf, die sich durch die Kombination von Cloud Computing und Robotik ergeben. Der Schwerpunkt dieser Arbeit lag darin, das konkrete Ziel der Masterarbeit vorzustellen. Im Rahmen der Vorarbeiten wurde eine Architektur vorgestellt, welche die Voraussetzungen erfüllt, dass Roboter wie der humanoide Roboter Nao mit der Cloud kommunizieren können. Des Weiteren wurde der Entwurf einer Schnittselle präsentiert, welche dazu dient, den Nao Roboter mit dem Cloud Service RoboEarth zu verknüpfen. Dadurch soll erreicht werden, dass der Nao Roboter aus der Cloud Informationen über Objekte, Umgebungen und Pläne herunterladen und verarbeiten kann. Die weitblickende Intention dieser Arbeit ist es, zu ermöglichen, dass heterogene Roboter über die Cloud abstrakte Pläne, wie z.B Wäsche waschen, kochen oder Tisch decken untereinander austauschen können.

Literaturverzeichnis

Aldebaran-Robotics.2012.http://users.aldebaran-robotics.com/docs/site_en/index_doc.html (Zugriff am 27. 02 2012).

Aldebaran-Robotics. *NAO H25 Humanoid Robot Platform.* 2012.

Inaba, Masayuki. *Remote Brained Robots.* Tokio, 1993.

Inbar, Elad. *robotappstore.* 2011. http://www.robotappstore.com/ (Zugriff am 25. 02 2012).

Marco, Daniel Di, Moritz Tenorth, Kai Häussermann, Oliver Zweigle, und Paul Levi. *RoboEarth Action Recipe Execution.* 2011.

Michael Beetz, Lorenz Mösenlechner, Moritz Tenorth. *CRAM - A Cognitive Robot Abstract Machine for Everday Manipulaiton in Human Enviroments.* 2010.

Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng. *ROS: an open-source Robot Operating System.* 2009.

Moritz Tenorth, Ulrich Klank, Dejan Pangericic, Michael Beetz. „Web-Enabled Robots.“ 2010.

Parent, Bastien. *Aldebaran Robotics, worldleader in humanoid robotics, raises US$13 million in round led by Intel Capital.* 2011.

Perzylo, Alexander, Moritz Tenorth, und Tobias Roehm. *The RoboEarth Language - Description of Requirements.* 2010.

Quigley, Morgan, Brian Gerkey, Ken Conley, Josh Fausty, und Tully Footey. *ROS: an open-source Robot Operating System.* 2010.

Quintas, Menezes, und Dias. *Cloud Robotics: Towards context aware Robotic Network.* 2011.

Rittweiler, Tobias Christian. *CRAM - Design & Implementation of a Reactive Plan Language.* 2010.

Robotics, Aldebaran. *http://www.aldebaran-robotics.com/en/Pressroom/About/NAO.html.* 27. 2 2013.

Tenorth, Moritz, und Michael Beetz. *KnowRob - Knowledge Processing for Autonomous Personal Robots.* München, 2010.

Zweigle, Molengraft, Andrea, Häussermann. *RoboEarth – connecting Robots worldwide.* Eindhoven, 2009.