Rocon - Eine Systemumgebung zur Steuerung und 3D-Visualisierung frei kombinierbarer aktorischer Roboterbausteine mit sensorischer Rückkopplung

Jörg Kaiser[1], Mohammad Ali Livani[1], and Thomas Strzeletz[2]

[1] Universität Ulm, Abteilung Rechnerstrukturen, D-89069 Ulm, Germany
[2] Universität Ulm, Abteilung Datenbanken und Informationssysteme, D-89069 Ulm, Germany

Zusammenfassung Rocon ist eine Umgebung zur Modellierung, Visualisierung und Steuerung von Roboterkomponenten. Ein Roboter wird hierbei als System einzelner Konstruktionselemente angesehen, die in beliebiger Form kombiniert werden können. Dazu müssen die geometrischen und die funktionellen Eigenschaften der Elemente modelliert werden. Dieses Baukastenprinzip mit wiederverwendbaren Komponenten erlaubt eine einfache „virtuelle“ Konstruktion neuer Roboterkomponenten aus bereits bestehenden. Die Benutzeroberfläche von Rocon erlaubt die Manipulation des Roboters, indem man über ein geeignetes Eingabemedium (z.B. die Maus) die virtuelle Darstellung des Roboters manipulieren kann. Aus dem Modell werden dann die Steuersignale für die physischen Komponenten abgeleitet. Zum andern erlaubt die Oberfläche die Beobachtung des Roboters, indem sie über sensorische Rückkopplung die tatsächlichen Positionen der aktorischen Komponenten anzeigt. Rocon wurde in Java implementiert und am Beispiel einer aktiven Niveauregulierung getestet. Als Netzwerkverbindung zwischen Rocon und dem komplexen Aktuator wurde TCP/IP und ein Gateway zum CAN-Bus eingesetzt.

1 Einleitung

Die Modellierung, Simulation und Visualisierung komplexer Roboter, die aus vielen miteinander verbundenen aktorischen Elementen bestehen, wie z.B. Greifwerkzeuge, Roboterarme oder die Beine von Schreitmaschinen sind aus vielen Gründen wünschenswert:

1. Solche Roboter erfordern häufig komplizierte Algorithmen zu ihrer Steuerung. Wünschenswert wäre hier eine geometrische und funktionale Modellierung des Roboters, mit deren Hilfe diese Algorithmen getestet werden können, bevor sie im physischen Roboter angewendet werden.
2. Die manuelle Steuerung dieser Roboter sollte über eine adäquate Schnittstelle erfolgen. Hierbei liefert die Visualisierung über Bildschirm und Steuerung mit der Maus oder einem anderen adäquaten Eingabemedium eine bessere Möglichkeit als numerische Eingaben.

3. Die Beobachtung des tatsächlichen Roboters ist häufig nicht möglich, da kein direkter visueller Kontakt besteht, z.B. bei der Wartung schwer zugänglicher technischer Systeme, der Ressourcenexploration und der Telemedizin. Auch der Beobachtung des Roboters über eine Videokamera sind hier Grenzen gesetzt. Eine visuelle Kontrolle des Roboters kann vorteilhaft über die graphische Oberfläche erfolgen.

Rocon wurde als Werkzeug zur Modellierung, Visualisierung und Steuerung komplexer Roboterkomponenten entwickelt. Dabei wurden mehrere Entwurfsziele verfolgt:

1. Rocon bietet die Möglichkeit, Roboter nach dem Baukastenprinzip zu entwerfen, d.h. aus vorgefertigten, parametrisierbaren geometrischen und funktionalen Komponenten einen Roboter „virtuell“ zusammenzubauen.
2. Das Zusammenspiel der Bausteine wird simuliert und durch eine 3D-Visualisierung sichtbar gemacht, so daß das Verhalten des Roboters als Ganzes beobachtet werden kann. Die Eingabe für die Simulation kann einmal über die Benutzerschnittstelle erfolgen, zum anderen können Steuerungsprogramme in Java erstellt werden, um komplexere Bewegungsmuster zu definieren.
3. Aus der Simulation des Roboters werden Steuersignale für die physischen Komponenten generiert, so daß eine direkte Steuerung des Roboters erfolgen kann, die mit der visuellen Darstellung übereinstimmt.
4. Die aus der Simulation abgeleiteten Steuersignale sollen über TCP/IP an das zu kontrollierende Objekt geschickt werden.
5. Zur Überprüfung des tatsächlichen (Ist-)Zustands des Roboters wird eine sensorische Rückkopplung unterstützt. Dadurch ist es möglich, die tatsächliche Stellung einer Komponente relativ zu anderen Komponenten oder die eines Gelenkes zu ermitteln und in die Visualisierung einzubeziehen. Darüber hinaus können Parameter der Umwelt, die nicht aus dem Modell des Roboters abgeleitet werden können, wie z.B. die Lage im Raum bei nicht stationären Robotern berücksichtigt werden.
6. Bewegungen des Roboters, die als Reaktion auf die Sensorwerte zwingend sind, werden bereits im Simulationsprogramm automatisiert und können nach erfolgreicher Simulation durch die Verbindung mit den physischen Sensoren und Aktuatoren im Einsatz erprobt werden.

Bestehende Entwicklungssysteme (RobotAssist [New], Workspace [Rob], oder EASY-ROB [Ste]) ermöglichen die Konstruktion und Simulation von Robotern, die aus beliebigen aktorischen Bausteinen bestehen. Darüber hinaus ist eine Steuerung der Roboter über ein Datennetzwerk ohne weiteres realisierbar. Diese Systeme berücksichtigen jedoch nicht die sensorische Rückkopplung. Insbesondere werden die unter 5. und 6. angeführten Ziele ohne sensorische Rückkopplung nicht erreicht.

Zur besseren Veranschaulichung werden die Konzepte von Rocon anhand eines Beispielsystems vorgestellt. Bei diesem Beispiel handelt es sich um einen dreibeinigen Roboter, der in der Lage ist, dynamische Lageveränderungen seines

Untergrunds so zu kompensieren, daß eine Ablageplatte auf dem Roboter stets die horizontalen Lage beibehält.

Dieses Papier ist in folgende Teile gegliedert: Kapitel 2 stellt die Architektur des Rocon-Systems vor. Auf das interne Objektmodell und die Schnittstellen des Systems wird im Kapitel 3 näher eingegangen. Im Kapitel 4 wird die Funktionsweise des Rocon-Systems anhand des Anwendungsbeispiels „Dreibeiner“ beschrieben. Eine Zusammenfassung schließt das Papier ab.

2 Architektur von Rocon

Rocon ist ein System, das dem Benutzer die flexible Konstruktion und Steuerung eines Roboters ermöglicht. In Abbildung1 ist die generelle Einordnung des Systems zwischen dem Benutzer und dem echten Roboter dargestellt.

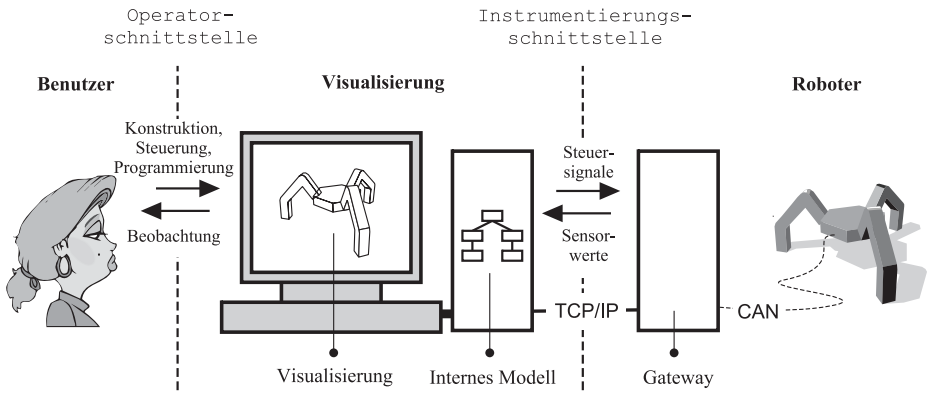

Abbildung1. Einordnung des Systems

Das System bedient sich einer Modellierung der realen Objekte durch computerinterne Datenstrukturen. Der Benutzer kann über die graphische Schnittstelle ein solches Modell erstellen, eine dreidimensionale Visualisierung des Modells betrachten und direkt an dem dargestellten Bild Manipulationen vornehmen, d.h. einzelne Gelenke per Maus auswählen und bewegen. Über eine Programmierschnittstelle kann das Modell durch ein Programm manipuliert werden, wodurch komplexere Bewegungsverhalten erzielt werden können. Rocon leitet aus dem Modell Steuerungsbefehle für die realen Aktuatoren ab. Diese werden über TCP/IP an ein Gateway zum CAN-Bus verschickt. Andererseits ist es möglich, Werte von Sensoren der Realität in das interne Modell zu übernehmen, so daß durch Sensoren gemessene Änderungen unmittelbar auf dem Bildschirm sichtbar werden.

3 Internes Objektmodell

Bei der Betrachtung des Datenmodells sind zwei Aspekte zu unterscheiden: die Visualisierung und die Steuerung. Das Modell muß die Informationen liefern können, die für eine 3D-**Visualisierung** benötigt werden. Beispiele für solche Modelle sind ein Java3D-Szenengraph [DS98], oder eine Szene, die mit VRML beschrieben wird [ANM96]. Für die **Steuerung** eines Roboters ist seine äußere Erscheinung relativ unwichtig. Ob der Arm eines Roboters die Form eines Zylinders oder die eines Quaders hat, ist irrelevant. Hier werden Metainformationen benötigt, die man dem echten Roboter eigentlich nicht unmittelbar ansieht. Diese Metainformationen beschreiben die Position und Richtung der Gelenkachsen und minimale und maximale Auslenkung der Gelenke. Ein Beispiel für eine Modellierung derartiger Informationen sind Denavit-Hartenberg-Koordinaten [DH55], die häufig zur Steuerung von Robotern verwendet werden [Pau81], [Fea84].

Das hier vorgestellte Modell muß diese beiden Aspekte vereinen. Darüber hinaus müssen wiederverwendbare Teile unterstützt und unterschiedliche Sensoren berücksichtigt werden. Im folgenden muß zwischen einem vollständigen Roboter, Roboterbauteilen und Elementarteilen unterschieden werden. Sowohl ganze Roboter, als auch Roboterbauteile setzen sich aus drei verschiedenen Elementarteilen zusammen. Sie werden repräsentiert durch die drei Klassen *Element*, *Joint* und *Anchor*.

Element abstrahiert von den physischen Körpern in der Wirklichkeit. Ein *Element* speichert die Attribute, die für eine dreidimensionale Visualisierung notwendig sind. Dazu gehören Koordinaten der Punkte, die die Oberfläche beschreiben, und Listen von Einzelflächen zwischen diesen Punkten, aus denen sich die Oberfläche des darzustellenden Körpers zusammensetzt. Die Klasse kann zu Grundformen wie Zylinder, Kegel oder Quader spezialisiert werden. Für den Benutzer und auch für die Implementierung ist eine Annäherung an die Realität durch einzelne Quader am einfachsten. Der Nachteil dieser Vereinfachung ist die schlichte Darstellung des Roboters und Ungenauigkeiten bei weiteren Berechnungen wie Kollisionstests.

Ein Objekt vom Typ ***Joint*** speichert die Eigenschaften der beweglichen Verbindung zweier Körper. Es beschreibt die Art des Gelenkes (Translation oder Rotation), die minimal und maximal mögliche Auslenkung der Verbindung und den aktuellen Winkel, also die Stellung der beweglichen Verbindung zu einem bestimmten Zeitpunkt. Dieser aktuelle Winkel kann durch den Benutzer, durch Meldungen von Sensoren oder durch ein kleines Programm verändert werden.

Die genannten Elementarteile werden in einer hierarchischen Struktur organisiert, welche die geometrische Beziehung der Objekte zueinander widerspiegelt. Jeder Knoten besitzt eine geometrische Transformation, bestehend aus Rotation und Translation, die seine relative Stellung zu seinem Vorgänger in der Hierarchie angibt. Die Transformation des Knotens an der Wurzel jeder Hierarchie gibt seine absolute Stellung im virtuellen Universum an. Abbildung2 zeigt einen Greifarm, der vereinfacht dreidimensional dargestellt wird, und die entsprechende Hierarchie im Modell.

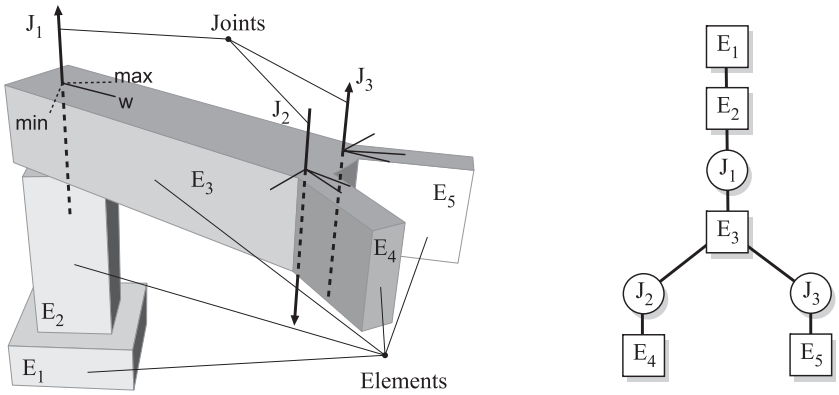

Abbildung2. Ein einfacher Roboter und der zugehörige Graph

Die Form des Greifarms wird durch einzelne Quader vom Typ *Element* angenähert. An den beweglichen Verbindungen sind *Joint*-Objekte eingezeichnet, die logische Informationen beschreiben. Dazu gehören die Richtung und Orientierung der Drehachse, minimale und maximale Auslenkung (min, max) und aktueller Winkel w (stellvertretend für *Joint* J_1 eingezeichnet).

In der Baumstruktur können *Joint*- und *Element*-Instanzen in beliebiger Kombination auftreten. Durch eine direkte Folge mehrerer *Element*-Objekte kann ein Baustein mit einer komplexeren Form modelliert werden. Eine *Element-Element*-Folge im Graphen stellt also eine unbewegliche Verbindung zweier Körper dar. In Abbildung2 wurde z.B. der Rumpf des Greifarmes durch E_1 und E_2 zusammengesetzt.

Befindet sich zwischen zwei *Element*-Objekten genau ein *Joint*-Objekt, so wird ein bewegliches Gelenk mit einem Freiheitsgrad modelliert (zum Beispiel die Kombination $E_2 - J_1 - E_3$ in Abbildung2).

Soll ein Gelenk mit mehr als einem Freiheitsgrad (wie z.B. das Hüftgelenk des Menschen) modelliert werden, wird statt eines einzelnen *Joint*-Objekts eine Folge von mehreren *Joints* eingesetzt. Mit einem einzelnen *Joint*-Objekt kann nur eine Bewegung mit einem Freiheitsgrad entlang einer Achse beschrieben werden.

Es fehlt nun noch ein Mechanismus, der es dem Benutzer erlaubt, einen Roboter aus wiederverwendbaren Bauteilen zusammenzusetzen, so daß der Vorgang der Konstruktion erleichtert wird. Die Bauteile sollen vordefinierte Punkte aufweisen, an denen sie in bestimmten Kombinationen verbunden werden können. Im Modell betrachtet sind wiederverwendbare Bausteine Teilbäume aus der Hierarchie, die zu größeren Bäumen kombiniert werden können. Wird eine *Element-Joint*-Folge als Trennungsstelle definiert, so ist die Gestalt eines Teilbaumes und damit eines Einzelteils eindeutig festgelegt.

Die Verbindungspunkte zwischen den Bauteilen werden durch die Klasse ***Anchor*** repräsentiert. Jedes *Element* kann beliebig viele Ankerpunkte besitzen,

an deren Positionen andere Bauteile anschließen können. Ein *Anchor*-Objekt legt mit seiner Transformation eine Position und Orientierung relativ zu seinem Vorgängerknoten fest. Soll ein Bauteil B angeknüpft werden, so wird diese Transformation an das Objekt an der Wurzel des Teilbaumes B übergeben. Auf diese Weise wird B an die Stelle des Ankers positioniert. Abbildung3 zeigt die Zerlegung des Greifarms in Einzelteile.

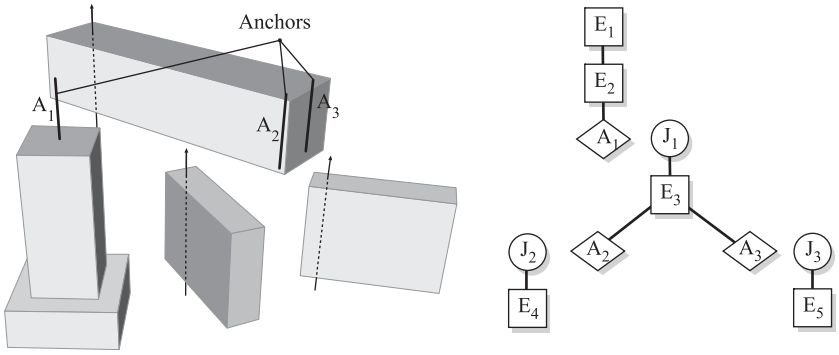

Abbildung3. Die Zerlegung des Greifarms in Einzelteile.

In Abbildung4 ist zu sehen, über welche Schnittstellen auf das interne Modell zugegriffen wird. Diese Schnittstellen werden im folgenden detailliert beschrieben.

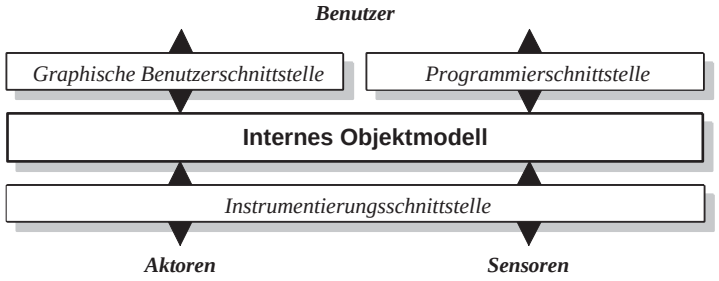

Abbildung4. Die Schnittstellen zum Objektmodell

Die **graphische Benutzerschnittstelle** ist die Verbindung zwischen dem internen Datenmodell und dem Anwender. Sie ermöglicht die Konstruktion und Manipulation eines Roboters mit Hilfe der Tastatur und der Maus. Die dreidimensionale Visualisierung stellt dabei stets das interne Modell dar, wobei Änderungen unmittelbar sichtbar werden. *Joint*-Objekte werden auf ähnliche Art und

Weise wie in Abbildung2 dargestellt. Der Benutzer kann bei der Konstruktion vorhandene Elemente aus einem Bausatz zu Robotern verknüpfen und auch den Bausatz um neue Teile erweitern. Die Verknüpfung von Bauteilen gestaltet sich dabei sehr einfach, so daß hierfür keine umfangreichen Kenntnisse ber dreidimensionale Gestaltung notwendig sind.

Abbildung5 zeigt das Erscheinungsbild der Anwendung nach dem Start. Dort sind im wesentlichen vier verschiedene Ansichten des aktuellen Modells zu sehen, die Manipulationen in der jeweils dargestellten Ebene erlauben. Auf der linken Seite befindet sich eine Informationstafel, über die Attribute der Teile eines Roboters angezeigt und manipuliert werden können.

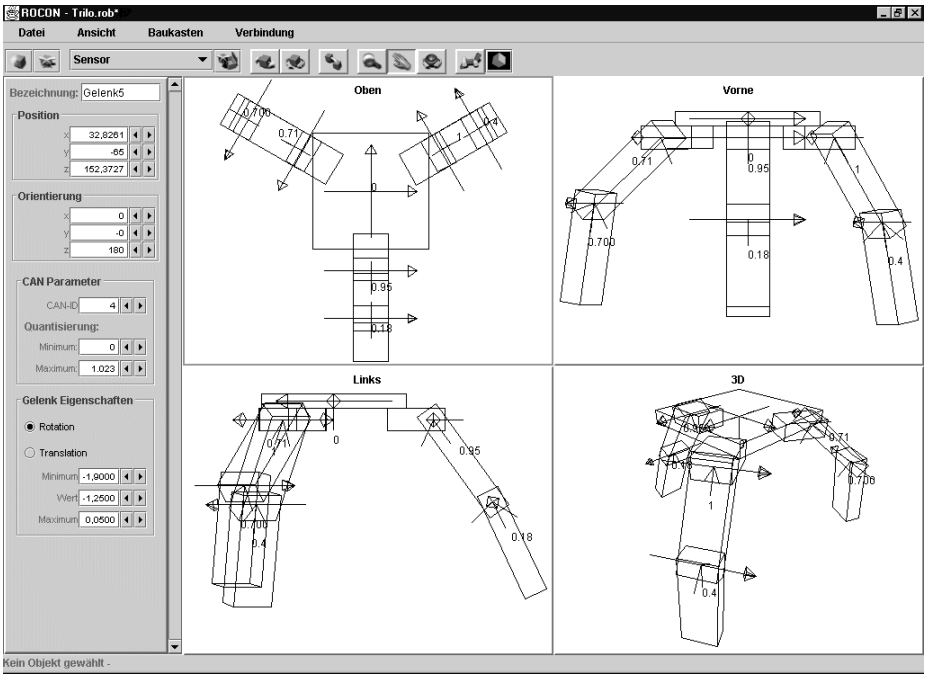

Abbildung5. Die Benutzerschnittstelle

Da die graphische Schnittstelle nur eine beschränkte Möglichkeit zur Manipulation bietet (es kann stets nur ein Gelenk per Maus gleichzeitig bewegt werden), wird als Erweiterung eine **Programmierschnittstelle** angeboten. Rocon bietet die Möglichkeit an, zur Laufzeit eine Java-Klasse dynamisch zu laden und diese über eine festgelegte Schnittstelle aufzurufen. Das Java-Programm kann über eine definierte Schnittstelle an dem aktuellen Modell beliebige Manipulationen vornehmen. Durch die Verwendung von Java muß sich der Programmierer nicht in eine proprietäre (Skript-)Sprache einarbeiten.

Der folgende Programmtext zeigt, wie in dem Beispiel *Joint* J_3 dazu gebracht werden kann, sich simultan mit J_2 zu bewegen:

```
1   ...
2   public class MyControl extends Control {
3
4       class MyListener implements ValueChangedListener {
5           Joint otherJoint;
6           public MyListener(Joint otherJoint) {
7               this.otherJoint = otherJoint;
8           }
9           public void valueChanged(ValueChangedEvent thisJointEvent) {
10              otherJoint.setValue(thisJointEvent.getValue());
11          }
12      }                   // init() überschreibt eine Methode der Klasse Control
13                          // sie wird durch Rocon nach Laden der Klasse gerufen
14      public void init() {// getJoint(x) liefert das Joint-Objekt mit der Bezeichnung x
15          getJoint("J2").addValueChangedListener(new MyListener(getJoint("J3")));
16      }
17  }
```

Wie zu erkennen ist, wird ein ereignisbasiertes Modell der Kommunikation realisiert, das sich sehr gut für Kontrollsysteme eignet. Eine Änderung in Gelenk J_2 bewirkt, daß diese Änderung an alle interessierten Objekte gesendet wird. Programmzeile 15 trägt bei Gelenk J_2 einen neuen Ereignisempfänger (nämlich J_3) ein. Eine Diskussion dieses Modells für Kontrollanwendungen und Lösungen für den CAN-Bus ist in [KM99] zu finden.

Die Programmierschnittstelle erlaubt die Realisierung komplexer Bewegungsalgorithmen durch die Möglichkeit, Beziehungen und Reaktionen zwischen den geometrischen und funktionalen Komponenten des Simulationsmodells zu definieren und durchzusetzen. Beispielsweise könnten an dieser Stelle spezielle Algorithmen der Kollisionsvermeidung von Gliedern oder der inversen Kinematik einbezogen werden.

Die **Instrumentierungsschnittstelle** verbindet bidirektional über das erwähnte CAN-Gateway die *Joint*-Objekte des virtuellen Modells mit verschiedenen Mikrocontrollern in Realität. Jedem *Joint*-Objekt kann über die graphische Oberfläche ein eindeutiger CAN-Identifier zugewiesen werden, der direkt zur Adressierung der CAN-Nachrichten verwendet wird [ROB91]. In Abbildung5 sind zum Beispiel die Attribute des *Joint*-Objekts mit der Bezeichnung „Gelenk5“ zu sehen, dem in der Kategorie „CAN-Parameter“ die ID 4 zugewiesen wurde. Die Manipulation des aktuellen Winkels eines *Joint* bewirkt ein Versenden einer Nachricht mit entsprechender ID und dem neuen Winkel. Dabei wird der Winkel, der im internen Modell als Gleitkommazahl vorliegt, auf ein geeignetes Netzwerkpaket abgebildet. Sollen die Daten zum Beispiel in einem 10-Bit-Paket versendet werden, so muß der Wertebereich eines *Joint* auf den Zahlenbereich von 0 bis 1023 abgebildet werden. Die Grenzen dieses Bereiches können über die Benutzerschnittstelle als Quantisierungsparameter angegeben werden (Abbildung5).

Ein intelligenter Aktuator, dessen CAN-Controller die für ihn bestimmte Nachricht anhand ihrer ID identifiziert, entnimmt der Nachricht die gewünschte Stellung und erzeugt die entsprechenden Steuersignale für den gewünschten

Motor. Umgekehrt werden Nachrichten, die Sensordaten enthalten, über die Instrumentierungsschnittstelle an die entsprechenden Objekte der Simulation weitergeleitet und entsprechend visualisiert. Eine detaillierte Beschreibung der Instrumentierungsschnittstelle und der dazugehörigen Netzwerkarchitektur liegt außerhalb des Rahmens dieses Papiers. Sie wird in [Str99] dargestellt.

4 Anwendungsbeispiel: Dreibeiner

Das in der Einleitung erwähnte Beispiel zur aktiven Niveauregulierung durch einen dreibeinigen Roboter soll nun detailliert beschrieben werden. Die in Abbildung5 dargestellte Oberfläche zeigt diesen Beispielroboter. Damit der Roboter die Plattform bei Neigung der Standfläche horizontal ausrichten kann, besitzt er drei Beine, die jeweils aus zwei servogesteuerten Gelenken bestehen. Ein Bein kann dadurch jeweils eine Ecke der Ablageplatte heben bzw. senken. An der Plattform ist ein Inklinometer installiert, der die Neigung zweier orthogonaler Achsen gegenüber der Erdoberfläche mißt. Ein Steuerungsprogramm soll mit Hilfe dieser Meßwerte die Stellung der Beingelenke berechnen, so daß die Plattform stets horizontal ausgerichtet ist. Abbildung6b) zeigt die Hierarchie der Elementarteile des zugehörigen Modells.

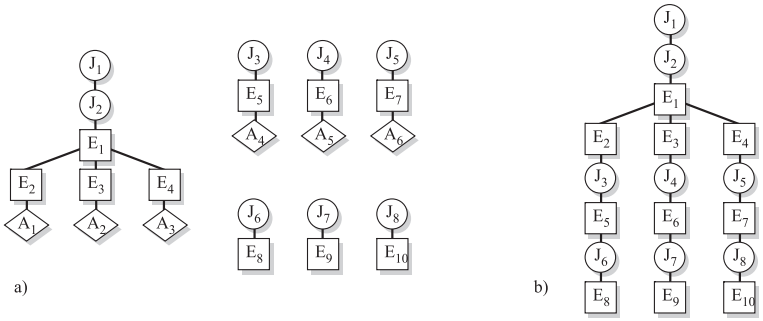

Abbildung6. Modell des Dreibeiners. a) einzelne Bauteile. b) zusammengesetzt.

Die sechs servogesteuerten Gelenke werden durch die sechs *Joint*-Objekte $J_3 \ldots J_8$ repräsentiert. Jedem dieser Objekte wurde eine ID zugeordnet, über die das jeweils korrespondierende Servo angesprochen wird. Die beiden Objekte J_1 und J_2 repräsentieren den Neigungswinkelsensor, wobei die Achsen beider *Joints* horizontal im virtuellen Universum liegen und die Achse von J_2 orthogonal zu der von J_1 ausgerichtet ist. Die aktuelle Stellung von J_1 und von J_2 bestimmen die Neigung der darunterliegenden Hierarchie und somit die des gesamten Roboters. Die IDs von J_1 und J_2 werden so gewählt, daß sie die Nachrichten des echten Inklinometers empfangen. Wird in Realität die Plattform geneigt, so sendet der Inklinometer an die zugehörigen Objekte den neuen Winkel gegenüber

der Erdoberfläche. Diese stellen ihren aktuellen Winkel entsprechend ein, und der virtuelle Roboter wird gegenüber der virtuellen Horizontalen entsprechend der Realität geneigt.

Ein Steuerungsprogramm kann anhand der aktuellen Winkel von J_1 und J_2 die Stellungen der Beingelenke berechnen, die für einen Ausgleich der Plattform notwendig sind. Das Programm manipuliert dabei die Soll-Winkel der *Joints* $J_3 \ldots J_8$ im Modell. Die Kommunikation mit den sechs echten Mikrocontrollern erledigt Rocon.

5 Zusammenfassung

Rocon ist ein Werkzeug, welches universell zur Modellierung, Visualisierung und Steuerung beliebiger Roboter verwendet werden kann. Durch Wiederverwendung bereits konstruierter Bauteile läßt sich sehr leicht ein Modell eines bestimmten Roboters zusammensetzen. Dieses kann über die Visualisierung manipuliert oder über ein Programm gesteuert werden. Rocon organisiert mit Hilfe des Modells die Kommunikation mit den echten Mikrocontrollern. Bei der Steuerung wird von diesem Aspekt abstrahiert, da nicht mehr einzelne Motorsteuerungen zu programmieren sind, sondern das generische virtuelle Modell.

Literatur

[ANM96] Andrea L. Ames, David R. Nadeau, and John L. Moreland. *The VRML sourcebook*. Wiley, New York, NY, USA, 1996.

[DH55] J. Denavit and R. S. Hartenberg. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *J. Appl. Mechanics, June 1955*, 22:215–221, 1955.

[DS98] M. Deering and H. Sowizral. *Java3D Specification, Version 1.1*. Sun Microsystems, 2550 Garcia Avenue, Mountain View, CA 94043, USA, dec 1998.

[Fea84] Roy Featherstone. *Robot dynamics algorithms / by Roy Featherstone*. Boston : Kluwer, c1987, 211 p., 1984.

[KM99] J. Kaiser and M. Mock. Implementing the Real-Time Publisher/Subscriber Model on the Controller Area Network (CAN). In *Proceedings of the 2nd Int. Symp. on Object-oriented Real-time distributed Computing (ISORC99), Saint-Malo, France*, may 1999.

[New] New River Kinematics, 4767 Wurno Road, Pulaski, VA 24301, USA. *RobotAssist*.

[Pau81] R. P. Paul. *Robot Manipulators: Mathematics, Programming and Control*. MIT Series in Artificial Intelligence. The MIT Press, 1981.

[Rob] Robot Simulations Ltd., 21 Amethyst Road, Newcastle-upon-Tyne, NE4 7YL, UK. *Workspace*.

[ROB91] ROBERT BOSCH GmbH. CAN specification version 2.0, sep 1991.

[Ste] Stefan Anton, http://www.easy-rob.de. *EASY-ROB*.

[Str99] Thomas Strzeletz. Entwicklung einer Umgebung zur Steuerung und 3D-Visualisierung hierarchisch kombinierbarer aktorischer Konstruktionselemente mit sensorischer Rückkopplung. Master's thesis, Universität Ulm, 1999.