Mensch-Maschine-Kommunikation in IVUs: Der kommunikative Agent Lokutor

Jan-Torsten Milde, Tobias Ahlers
Fakultät für Linguistik und Literaturwissenschaft, Universität Bielefeld,
EMAIL: milde@coli.uni-bielefeld.de, tahlers@techfak.uni-bielefeld.de

1 Lokutor

Lokutor ist ein natürlichsprachlich steuerbarer anthropomorpher Agenten-Prototyp, der innerhalb einer simulierten 3D Umgebung agiert (s. a. [MA99a], [MA99b]). Seine aktuelle Aufgabe ist die Präsentation eines Autos. Der Agent ist in der Lage, die Funktionalität des Autos zu beschreiben (z.B. wie öffne ich den Tankdeckel, welchen Typ Benzin muß man einfüllen, wie groß ist das Volumen des Kofferraums etc.). Der Benutzer interagiert mit Lokutor durch natürlichsprachliche Kommunikation. Lokutor interpretiert die gegebenen Anweisungen unter Einbeziehung situativer Information und führt die entsprechenden Teilhandlungen aus. Informationen über das Auto werden natürlichsprachlich wiedergegeben. Hier kommt ein Synthesemodul[1] zum Einsatz. Der Inhalt der Beschreibung ist Teil der Verhaltensspezifikation von Lokutor. Diese wird angegeben in einer XML-annotierten Form . Die Wissensbasis des Agenten besteht dabei aus strukturiert annotiertem Domänenwissen (in der Testanwendung sind dies aufbereitete Ausschnitte aus dem Benutzerhandbuch des Wagens) und in der Beschreibung möglicher Verhaltensmuster. Ein Verhaltensmuster kombiniert dabei eine Situationsbeschreibung mit einer Folge möglicher Handlungen. Unter Handlung wird dabei sowohl eine physikalische Handlung, wie auch eine Sprechhandlung verstanden. Das spezifizierte Verhaltensmuster wird umgesetzt in ein Behaviormodul des reaktiven Basissystems.

2 Implementation

Bei der Entwicklung von Lokutor wurde versucht Techniken zu verwenden, die das System möglichst einfach auf einer Vielzahl von Plattformen einsetzbar macht. Um dies zu gewährleisten, wurde das System weitgehend in Java implementiert. Für die grafische Simulation wurde auf die Klassenbibliothek Java3D zurückgegriffen, die Benutzerschnittstelle basiert auf Swing, die grafischen Modelle der Objekte werden im VRML2-Format gespeichert (s.a. [Klo98]), welche durch die verwendete VRML97-Bibliothek in den Java3D Szenegraphen eingebunden werden können. Eine Ausnahme bei der Verwendung von Java als Implementationssprache macht die Kollisionsüberprüfung. Die hierzu von Java3D zur Verfügung gestellte Funktionalität ist zu langsam, um in einer interaktiven Simulation eingesetzt werden zu können. Entsprechend wurde für die

[1] mbrola: http://tcts.fpms.ac.be/synthesis, wxtts: http://web.ukonline.co.uk/julian.smart/

schnelle polygongenaue Kollisionsüberprüfung, welche z.B. bei der Simulation von Sensoren wichtig ist, auf die VCollide-Bibliothek zurückgegriffen. Da diese vollständig in Standard Ansi C/C++ entwickelt wurde, war es möglich, sie ohne allzu großen Aufwand zu portieren. Hierauf aufsetzend wurde eine native Schnittstelle zu Java entwickelt (JCollide), so daß nun die volle Funktionalität und Geschwindigkeit von Vcollide auch in Java3D-Anwendungen genutzt werden kann. Die Grundlage der Objektgeometrie sind VRML-Modelle. Die geometrische Beschreibung von Lokutor entspricht dem H-Anim-Standard[2]. H-Anim erlaubt die Definition flexibel bewegbarer artikulierter Agenten. Das Skellettmodell basiert weitgehend auf dem menschlichen Vorbild (komplette Modellierung aller Wirbel, vollständiges Arm- und Handskelett, reduziertes Fußskelett, reduziertes Kopfmodell) und erlaubt so realitätsnahe Bewegungssimulationen. Durch die Standardisierung der Gelenkbezeichnungen in H-Anim wird es möglich, bereits definierte Bewegungssequenzen unmittelbar auf andere Agenten zu übertragen.

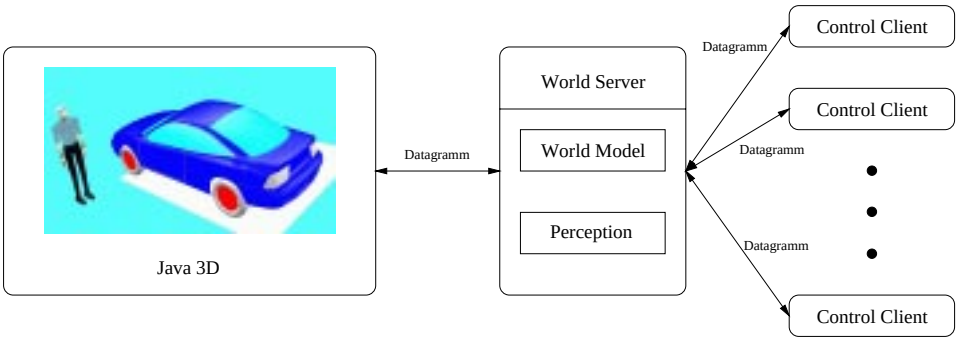

Figure 1: Aufbau der verteilten Simulation.

Die Simulation wurde als verteiltes System realisiert und besteht aus drei separaten Modulen: der Visualisierung, dem Weltserver und (mindestens) einem Kontrollprozeß (Abbildung 1). Zentrales Element der Systemarchitektur ist der Weltserver-Prozeß. Dieser verwaltet das Weltmodell, welches die Position und Ausrichtung des Agenten und aller Objekte der Szene repräsentiert. Virtuelle Objekte werden ebenfalls repräsentiert. Basierend auf diesen Informationen berechnet der Weltserver den aktuellen perzeptuellen Status des Agenten. Lokutor ist mit visuellen und taktilen Sensoren ausgestattet. Das Steuerungssystem kann den sensomotorischen Status des Agenten (Gelenkwinkel der Hände, Arme und Beine) festzustellen. Der Weltserver vermittelt zwischen den Kontrollprozessen und der simulierten Welt. Es ist möglich, eine beliebige Anzahl von Kontrollprozessen und Agenten im Weltserver zu registrieren.

Zum aktuellen Zeitpunkt läuft das System in vollem Umfang unter den Betriebsystemen Win9X/NT, Linux und Solaris, wobei sich die beste Performanz innerhalb der Windows-Umgebung ergibt. Dies liegt vor allen Dingen an der besseren Ausnutzung der Leistung aktueller Grafikkarten.

[2]s.a. http://ece.uwaterloo.ca:80/ h-anim/

3 Steuerung

Aktuelle Architekturprinzipien lassen sich in grob in drei Hauptgruppen aufteilen (siehe [SL99]):

1. rein reaktive Systeme
2. hybride reaktiv/deliberative Systeme, evtl. mit Gedächtnis
3. erweiterte hybride reaktiv/deliberative Systeme mit zusätzlicher emotional/motivationaler Komponente bzw. reflexiven Fähigkeiten

Reaktive Systeme kombinieren eine Anzahl von einfachen interagierenden Basiskomponenten, welche auf externe Situationsänderungen reagieren und so die Handlungsausführung beeinflussen. Hierzu zählen z.B. Ansätze auf der Basis von künstlichen neuronalen Netzen oder behaviororientierte Ansätze (s.a [Ark90], [Bro86], [Bro91], [Doy97], [vDKG94],).

Der Steuerungsansatz mittels künstlicher neuronaler Netze kombiniert die Vorteile einer robusten (numerischen) Verarbeitung sensorischer Daten mit der Fähigkeit, die Funktionalität des Kontrollsystems zu lernen bzw. basierend auf einer Situationsanalyse an die spezielle Fragestellung anzupassen. Ein so definiertes Steuerungssystem ist mithin prinzipiell offen für die Integration beliebiger Sensormodalitäten. Andererseits ist es gerade die bisher eingeschränkte Skalierbarkeit von künstlichen neuronalen Netzen, welche eine einfache Erweiterung i.d.R. verhindert. Beim behaviororientierten Steuerungsansatz liegen die Vorteile gegenüber einer Modellierung mit ANNs in der abstrakteren Dekomposition der Steuerungshierarchie. Die Funktion der sich überlagernden einzelnen Steuerungseinheiten (behaviors) ist klar definiert und somit auch von außen erkennbar und kontrollierbar. Durch die gewählte hierarchische Zerlegung sind Kompetenzebenen des Agenten identifizierbar und durch Hinzufügen weiterer Ebenen erweiterbar. Nachteilig ist jedoch, daß die so programmierten Behaviors nicht lernfähig oder auch nur adaptiv sind. Der Agent wird für seine Umgebung designed; er ist an seine ökologische Nische angepaßt, Handlungsfolgen werden durch die Umwelt ausgelöst, Motivationen sind fest verdrahtet (s.a. [MMRR99]).

Die Kombination einer reaktiven Basisarchitektur mit einer darüberliegenden deliberativen Schicht zu einer hybriden Steuerung stellt zur Zeit die am häufigsten verwendete Architekturkonzeption für autonome Agenten dar. Hierbei reduziert die reaktive Komponente den Aufwand für die Repräsentation der i.d.R. nur gering strukturierten dynamischen Umwelt sowie der expliziten abstrakten Kodierung aller Handlungsmöglichkeiten in dieser Umwelt.

Auch bei Lokutor erfolgt die Steuerung durch einen hybriden Architekturansatz, bestehend aus einer reaktiven Basiskomponente und einem deliberativen höheren System. Abbildung 2 zeigt die (geplanten) Architekturkomponenten und ihr Zusammenspiel. Das Basissystem betsteht aus dem Behaviorsystem und dem emotional/motivationalen System (EMS) erweitert. Die beiden Subsysteme (Behaviorsystem und EMS) sind gleichwertig Teile des Basissystems mit jeweils unterschiedlicher Aufgabenstellung. Im Behaviorsystem wird die situationsgerechte Aktionsselektion durchgeführt. Die Aufgabe des EMS besteht in der situationsgerechten Auswahl des Aktionsmodus. Behaviorsystem und EMS sind jeweils eigenständige dynamische Systeme mit unterschiedlicher Grundarchitektur. Während das Behaviorsystem aus eine Menge hierarchisch geordneter Behaviormodule besteht, ist das EMS als subsymbolisches Netzwerk organisiert.

Das Behaviorsystem besteht aus drei Teilkomponenten: der Schnittstelle, einem Scheduling-Prozeß und der Menge der Behaviormodule. Bei der Implementation des Behaviorsystems wurde versucht, ein generisches Modell eines Behaviormodules zu definieren. Die Beschreibung orientiert sich dabei an notwendigen Bestandteilen eines solchen Moduls. Entsprechend wurde eine abstrakte Klasse in Java definiert, welche ein generisches Behaviormodul beschreibt. In einem Behaviormodul, welches dieses Interface implementiert, müssen die Methoden *initBehavior*, *PreCondition*, *Body*, *PostCondition* und *pause* definiert werden:

```
public interface  behaviormodule extends Runnable {
  public void initBehavior (String n, Scheduler s, int d);
  public int PreCondition (Object params);
  public int Body (Object params);
  public int PostCondition (Object params);
  public void pause (int i);
}
```

Die so definierten Behaviormodule melden sich bei einem Scheduling-Prozeß an. Der Scheduler stellt wiederum eine Basisfunktion für den synchronisierten Zugriff auf die Schnittstelle zur Verfügung. Auch bei der Implementation der Schnittstelle konnte eine bereits in Java vordefinierte Klasse verwendet werden, welche lediglich um einige Methoden erweitert worden ist. Die so definierte Abstrahierung vereinfacht stark die Beschreibung von Behaviormodulen und damit die Entwicklung des Behaviorsystems. Besonders erleichtert wurde die Implementation durch die Möglichkeiten zur parallelen Programmierung in Java. Das Laufzeitsystem der virtuellen Maschine des Java Bytecode Interpreters sorgt für die Verteilung und Synchronisation der Threads, welche jeweils einem Behaviormodul entsprechen.

Das emotional/motivationale System sorgt einerseits für die Erzeugung neuer Kontrollparameter für die Handlungsauswahl und die Handlungsausführung, ist andererseits auch als Monitoringkomponente für das Behaviorsystem konzipiert. Im EMS werden unmittelbar verfügbare sensorische Informationen mit Statusinformation des Behaviorsystems und des höheren Systems kombiniert. Dies erlaubt die Klassifikation von Systemzuständen und Systemzustandsverläufen, die wiederum genutzt werden kann, um die Handlungsauswahl in später auftretenden Situationen zu parameterisieren. Intern ist das EMS als aufgebaut als ein subsymbolisches Netzwerk von Repräsentationseinheiten, die dynamisch miteinander interagieren. Jede diese Einheiten kann interpretiert werden als ein Aspekt einer Motivation oder Emotion.

Die Trennung von Sensor- und Motorsystem im sensomotorischen System liegt begründet in der Notwendigkeit, die Wahrnehmungsleistung und Handlungsausführung weitgehender kontrollieren bzw. modifizieren zu können. Dies soll erreicht werden durch die konzeptuelle Trennung der Sensorvorverarbeitung (im Sensorsystem), der Verknüpfung der verschiedenen Modalitäten (im Behaviorsystem) und schließlich der Handlungskontrolle (im Motorsystem). Die Aufgabe des Sensorsystems ist die Vorverarbeitung der Rohdaten, wie sie von den simulierten Sensoren geliefert werden. Das Sensorsystem liefert die aufbereiteten Sensordaten an das Basissystem (sowohl an das Behaviorsystem als auch an das EMS) und an das höhere System. Je nach Sensortyp sind dies numerische, boolesche oder symbolische Werte. Für den jeweiligen eingesetzten Sensor werden diese Werte im Sensorsystem interpretiert, und zwar parameterisiert durch

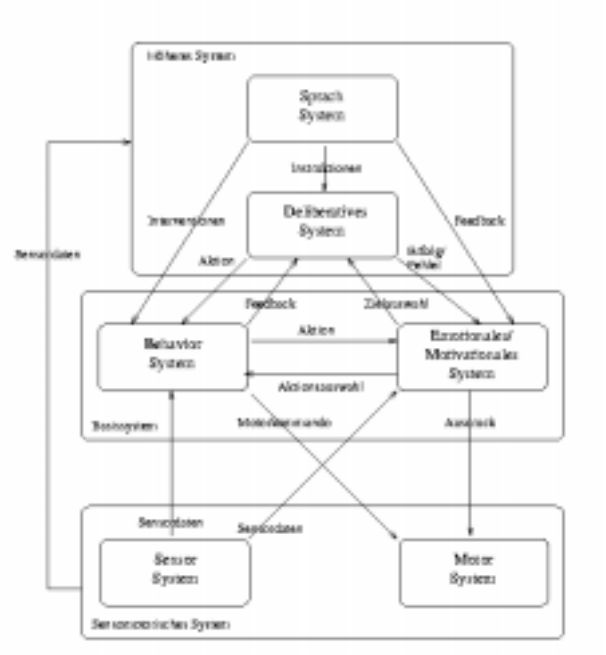

Figure 2: 3-schichtige Steuerungsarchitektur von Loktuor

die Anforderungen des jeweiligen Behaviormoduls. Das Motorsystem stellt ein Menge von parameterisierbaren Basishandlungen zur Verfügung. Sowohl das Behaviorsystem als auch das EMS erhalten parallelen Zugriff auf das Motorsystem. So können Basishandlungen initiert und modifiziert (z.B. in der Geschwindigkeit, der Lautstärke, der Genauigkeit etc.) werden. Im Motorsystem erfolgt auch die Aktivierung der Äußerungsproduktion auf Interventionsniveau.

Das Behaviorsystem von Lokutor wird deklarativ definiert. Hierzu wurde eine XML-Dokumenttypdefinition (DTD) entwickelt, die Grundlage ist für die strukturierte Verhaltensbeschreibung. Folgend eine vereinfachte Form der XML-DTD:

```
<!ELEMENT agent (geom,scene,knowb)>

<!ELEMENT geom (#PCDATA)>
<!ELEMENT scene (#PCDATA)>

<!ELEMENT knowb (entry+)>
<!ELEMENT entry (name,␣desc)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT desc (sit,␣action)>

<!ELEMENT sit (cond*)>
<!ELEMENT cond (att,val)>
<!ELEMENT att (#PCDATA)>
<!ELEMENT val (#PCDATA)>
```

```
<!ELEMENT action (move*,␣say*)>
<!ELEMENT move (basic*)>
<!ELEMENT basic (att,val)>

<!ELEMENT say (utt*)>
<!ELEMENT utt (#PCDATA)>
```

Die Beschreibung des Agenten (agent) besteht aus der Agentengeometrie (geom, Verweis auf das externe H-ANIM Modell), dem geometrischen Szenenmodell (scene, ebenfalls ein Verweis auf eine externe Datei) und der Wissensbasis (knowb). Die Wissensbasis besteht aus Einträgen, welche eine Situationsbeschreibung (sit, eine Folge von Bedingungen) mit einer Aktionsfolge (act) in Beziehung setzt. Ausführbare Aktionen bestehen aus physikalischen Handlungen (move) und aus möglichen Äußerungen (say). Ein anhand der Strukturbeschreibung annotiertes Dokument kann nun durch einen XML-Parser (IBM xml4j2.0.11, ein in Java geschriebener freier XML-Parser[3]) eingelesen und strukturell validiert werden. Alle XML-Dokumente sind Baumstrukturen. Der verwendete XML-Parser gibt diese Struktur als DOM-Tree zurück[4] und erlaubt so die Traversierung und damit die inhaltsgesteuerte Auswertung der Verhaltensbeschreibung.

4 Sprachverarbeitung

Handlungsanweisungen lassen sich aus der funktionalen Perspektive danach unterscheiden, in welcher Form bzw. zu welchem Zweck sie das handelnde System beeinflussen. Es lassen sich drei Gruppen von Anweisungen identifizieren (s.a [PSM97]):

- Anweisungen, die Aktionen initiieren
- Anweisungen, die in die Aktionsausführung eingreifen
- Anweisungen, die die Wiederaufnahme einer Aktion auslösen

Bei der Verarbeitung von sprachlichen Eingaben werden zwei Typen von Anweisungen unterschieden: Interventionen (initiieren Basisaktionen oder greifen in laufende Handlungen ein) und Instruktionen (werden durch die deliberative Komponente in Basisaktionen zerlegt, vgl. [MPS97], [GM98]).

Interventionen können unterschiedliche Funktionen haben: Sie können die aktuelle Aktion abbrechen, unterbrechen, verlängern oder modifizieren. Die Verarbeitung von Interventionen wird durch den FSTN-Interpreter geleistet. Die zu verwendende Grammatik wird in einer Datei im ASCII-Format abgelegt, von dem Interpeter geladen und die Eingabekette abgearbeitet. Das Ergebnis der Verarbeitung ist eine Reihe von Attribut-Wert Paaren, die einer Belegung in der Schnittstelle des reaktiven Steuerungsystems entspricht. Die Behaviormodule des reaktiven Steuerungssystems haben Zugriff auf diese Schnittstelle und können so aktiviert werden. Die Interpretation von Interventionen erfolgt direkt über das Basissystem. Die Behaviors des Basissystems haben Zugriff auf den aktuellen Sensorstatus des Agenten und entscheiden so, ob eine für sie relevante Situation vorliegt. Ist dies der Fall, so versuchen sie ihre Aktion zu initiieren. Dies gelingt jedoch nur, falls der Schedulingprozeß das Behavior aktiviert.

[3] http://www.alphaworks.ibm.com/
[4] http://www.w3.org/TR/WD-DOM/

Das deliberative System zerlegt komplexere Instruktionen in Folgen von Basisaktionen und gibt diese an das reaktive System weiter. So besteht etwas das Öffnen des Tankdeckels aus den drei Teilaktionen: *gehe* zum Tankdeckel, *bewege* Hand zum Tankdeckel, *öffne* den Tankdeckel. Die hybride Architektur ermöglicht eine natürlichsprachliche Steuerung in unterschiedlicher Granularität. Eine komplexe Aktionssequenz kann ausgelöst werden durch eine einzige Anweisung oder durch eine Folge von Anweisungen, die die Handlungsausführung in kleinen Schritten steuern.

Teilhandlungen können während ihrer Ausführung durch sprachliche Anweisungen modifiziert werden. Handelt es sich bei der Modifikation um eine Korrektur, wird bereits gegebene Informationen überschrieben. Greift der Benutzer unterstützend ein, so wird zusätzliche Information an das System gegeben.

Die natürlichsprachlichen Antworten des Agenten werden als Sprachhandlungen aufgefaßt und sind in der Verhaltenbeschreibung kodiert. Ihre Auswahl erfolgt also analog zur Bestimmung der nächsten Aktion und basiert wie diese auf der aktuell wahrgenommenen Situation. Der Inhalt der Sprachhandlung ist für die jeweilige Situation vorgegeben.

Um die Kommunikationssituation mit dem System möglichst natürlich zu gestalten, sind Schnittstellen zu einer kommerziellen Spracherkennungsoftware (IBM VoiceType) und zu einer TextToPhonem/Synthese-Sotware implementiert worden. Die interaktive Steuerung von Lokutor kann so vollständig natürlichsprachlich durchgeführt werden.

References

[Ark90] R.C. Arkin. Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation. In P. Maes, editor, *Designing Autonomous Agents : Theory and Practice from Biology to Engineering and Back*, volume 6,1 of *Robotics and autonomous systems*, pages 105–122, 1990.

[Bro86] R.A. Brooks. A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation*, volume 2, 1, pages 14–23, 1986.

[Bro91] R.A. Brooks. Challenges for Complete Creature Architectures. In J.-A. Meyer and S.W. Wilson, editors, *From animals to animats*, First International Conference on Simulation of adaptive Behavior, pages 434–443, 1991.

[Doy97] Kenji Doya. Efficient nonlinear control with actor-tutor architecture. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 1012. The MIT Press, 1997.

[GM98] Karl Ulrich Goecke and Jan-Torsten Milde. Natural language generation in a behavior-oriented robot control architecture. In *submitted to publication*, 1998.

[Klo98] Jörg et. al. Kloss. *VRML97 - Der neue Standard für interaktive 3D-Welten im WWW*. Addison-Wesley, 1998.

[MA99a] J.-T. Milde and Tobias Ahlers. The communicative agent lokutor. In *Posterbeitrag zu KogWis 1999, Bielefeld*, 1999.

[MA99b] J.-T. Milde and Tobias Ahlers. Lokutor: an articulated presentation agent. In *Workshopbeitrag zur KI-99, Bonn*, 1999.

[MMRR99] D. Metzing, J.-T. Milde, R. Rae, and H. Ritter. Kommunizierende Agenten: Gestische und natürlichsprachliche Interaktion. *Sonderheft Kognitionswissenschaft*, 1999.

[MPS97] Jan-Torsten Milde, Kornelia Peters, and Simone Strippgen. Situated communication with robots. In *Proceedings of the first international Workshop on Human-Computer Conversation, Bellagio, Italy*, 1997.

[PSM97] Kornelia Peters, Simone Strippgen, and Jan-Torsten Milde. Language processing in action-centered communication. In *Tagungsband der DGFS, Sektion Computerlinguistik*, Heidelberg, 1997.

[SL99] A. Sloman and B. Logan. Building Cognitively Rich Agents. *Communications of the ACM*, 42(3), 1999.

[vDKG94] J. W. M. van Dam, B. J. A. Kröse, and F. C. A. Groen. CNN: a neural architecture that learns multiple transformations of spatial representations. In M. Marinaro and P. G. Morasso, editors, *Artificial Neural Networks*, pages 1420–1423. Springer-Verlag, May 1994.