Reale Gesten zur Steuerung ausführbarer Modelle in der Evaluation bewegungsbasierter Interaktionen

Birgit Bomsdorf, Sebastian Hesse, Rainer Blum

Fachbereich Angewandte Informatik
Hochschule Fulda
Marquardstraße 35,
36039 Fulda
Birgit.Bomsdorf@informatik.hs-fulda.de
Sebastian.Hesse@informatik.hs-fulda.de
Rainer.Blum@informatik.hs-fulda.de

Abstract: In der modellbasierten Entwicklung bewegungsbasierter Interaktionen wird die frühzeitige Evaluation der Gesten, d.h. der Posen und der Körperbewegungen, von den aktuellen Werkzeugen noch nicht ausreichend unterstützt. Vor diesem Hintergrund werden im vorliegenden Beitrag erste Ergebnisse in der Entwicklung eines Werkzeugs für bewegungsbasierte Interaktionen (WeBewIn) vorgestellt. Auf Basis ausführbarer Modelle können unter Verwendung realer Gesten komplexe Gestenabläufe im Kontext verschiedener Interaktionssequenzen überprüft werden. Insgesamt erlaubt WeBewIn ein Rapid Prototyping von Gesteninteraktionen in einer kombinierten benutzer- und technik-basierten Vorgehensweise. Hierbei erstreckt sich eine frühzeitige Benutzerpartizipation nicht nur auf die Test-, sondern auch auf die Gestenfindungsphase, was über eine Gestenspezifikation mittels Vormachen (By-Demonstration) realisiert ist.

1 Einleitung

Interaktive Systeme werden zunehmend über Gesten bzw. mittels Körperbewegungen gesteuert. Auch wenn bewegungsbasierte Interaktionen nicht generell von Vorteil sind, finden sich doch spezielle Nutzungskontexte und Applikationen, in denen sie zur Usability, User Experience (UX) oder zur Barrierefreiheit beitragen können. Gerade aus diesem Blickwinkel heraus ist eine Benutzerpartizipation in einem iterativen Designprozess von Vorteil, um die Interaktionen und die mit ihnen auszuführenden Gesten gemeinsam mit dem Nutzer herauszufinden und zu überprüfen.

Die modellbasierte Erstellung von Benutzungsschnittstellen verfolgt ebenfalls eine benutzerzentrierte Vorgehensweise. Ausgehend von der Spezifikation der mit dem System durchzuführenden Aufgaben und den dabei involvierten Objekten werden die darauf basierenden Dialoge und die zugehörige Präsentation entwickelt [MPV11]. Die Aus-

führbarkeit der dabei entstehenden Modelle erlaubt eine frühzeitige Überprüfung erster Designentscheidungen. Entsprechende Werkzeuge (wie [BBS99], [MPS02], [RFD04]) bieten im Kern Schaltflächen, über deren Aktivierung simuliert wird, dass der Benutzer eine Aktion durchführt. Dabei dient eine Modellanimation der Visualisierung der Systemreaktionen. Über das Setzen von Bedingungen können die Abläufe entsprechend unterschiedlicher Nutzungssituationen durchgespielt werden.

Derartige Ansätze zum Testen bieten zwar eine frühzeitige Überprüfung der geplanten Interaktionen, richten sich jedoch in erster Linie an die Entwickler und unterstützen weniger die direkte Einbindung repräsentativer Endbenutzer. Sollen interaktive Systeme mittels Körperbewegungen gesteuert werden, sind die auszuführenden Gesten und deren Abfolgen von entscheidender Bedeutung für die Usability der Applikation. In diesem Beitrag werden erste Ergebnisse eines Entwicklungswerkzeugs für bewegungsbasierte Interaktionen (WeBewIn) vorgestellt, das nicht nur die Einbindung von Gesten direkt in den Modellierungsprozess sondern auch die Steuerung der entstehenden ausführbaren Modelle durch eine tatsächliche Ausführung von Raumgesten (reale Körpergesten) ermöglicht. Mit der Zielsetzung ein Rapid Prototyping mittels Low-Fidelity Prototypen zu unterstützen, kann zusätzlich ein grafischer Prototyp an ein ausführbares Modell angebunden und über Gesten gesteuert werden. Insgesamt unterstützt das Werkzeug eine frühzeitige Benutzerpartizipation in der Gestenermittlung (derzeit diskreter Körpergesten) und im modellbasierten Testen, indem reale Gesten sowohl zur Spezifikation der Bewegungen als auch zur Überprüfung der Modelle bzw. Interaktionen eingesetzt werden können.

Im folgenden Kapitel 2 wird zunächst auf Arbeiten mit Bezug zu WeBewIn eingegangen. Anschließend wird mit Kapitel 3 ein Überblick über die derzeitige Werkzeugumgebung gegeben, um darauf aufbauend in Kapitel 4 den Einsatz von WeBewIn zum Testen von Gesten im Kontext der Dialogmodellierung aufzuzeigen. In Kapitel 5 wird der präsentierte Ansatz kurz reflektiert, Einsatzmöglichkeiten und die aktuellen Entwicklungen des Werkzeuges aufgezeigt.

2 Vergleichbare Arbeiten

Das in [LL12] vorgestellte Werkzeug ermöglicht eine Gestenspezifikation mittels Demonstration, um den Implementierungsaufwand für Multitouch-Gesten zu reduzieren. Dies wird erreicht, indem der Entwickler die Geste auf einem berührungssensitiven Endgerät vormacht (eng. by demontration), diese durch das Werkzeug aufgezeichnet und anschließend ein zugehöriges Programmcodefragment generiert wird. Nach der Integration dieses Fragments in eine Anwendung ist diese in der Lage, die Geste zu erkennen. Um die Erkennungsrate einer Geste zu erhöhen, können mehrere Aufnahmen einer Touch-Geste miteinander kombiniert werden. Das durch die Autoren präsentierte Werkzeug und WeBewIn zeichnen demonstrierte Gesten auf, die bereits im Werkzeug getestet und anschließend in einer Anwendung verwendet werden können. Beide reduzieren den Entwicklungsaufwand und unterstützen die frühzeitige Benutzerpartizipation bei der Ermittlung von Gestensätzen. Der Unterschied liegt in der Art der verarbeiteten Gesten.

Das Werkzeug aus [LL12] verarbeitet Touch-Gesten, wohingegen im WeBewIn-Projekt mit Raumgesten gearbeitet wird.

Zurzeit existieren zwei weitere Werkzeuge, die das By-Demonstration Konzept für Raumgesten unterstützen. Beim Kinetic Space Tool[1] wird eine Geste nur einmal vorgemacht und kann anschließend mit unterschiedlichen Personen verwendet werden. Das Werkzeug lässt sich über ein vorgegebenes Kommunikationsprotokoll auch mit anderen Applikationen, z.B. einem Dialogeditor, verbinden um die Gesten dort einzusetzen. Eine engere Integration in einem einzigen Werkzeug, z.B. um zwischen Gestenspezifikation und Dialogmodellierung wechseln zu können, ist allerdings nicht möglich. Zudem erwies sich die Erkennungsrate der Software für unsere Ansprüche als mangelhaft, die Bedienbarkeit als zu komplex. Das Omek GAT[2] unterstützt ebenfalls das By-Demonstration Konzept und über die Omek Beckon Middleware die Nutzung der so spezifizierten Gesten in beliebigen Anwendungen. Jedoch erfordert die Festlegung einer Geste ein mehrmaliges Vormachen, möglichst durch unterschiedliche Personen, um das System zu trainieren. Es werden ca. 30 Trainingsdatensätze empfohlen. Für das von uns intendierte schnelle Prototyping von Gesten-Interaktionen ist diese Eigenschaft jedoch hinderlich.

Feuerstack et al. [FdP11] setzen sich in ihrer Arbeit mit der Modellierung und der Untersuchung von multimodalen Anwendungen auseinander. Diese werden mittels York Interaktoren [DH93] spezifiziert, welche wiederum auf ausführbaren Zustandsdiagrammen basieren. Bei der Modellierung unterscheiden die Autoren zwischen abstrakten und konkreten Interaktoren. Erstere werden zur Beschreibung modalitätsunabhängiger Interaktion genutzt. Konkrete Interaktoren hingegen beschreiben spezifisch die einzelnen Modalitäten und Medien. Die Aufteilung in eine abstrakte und eine konkrete Ebene folgt der bereits in [Ga03] präsentierten Differenzierung. WeBewIn ermöglicht eine Modellierung abstrakter und konkreter Interaktionen, unterstützt jedoch derzeit keine explizite Trennung dieser Ebenen. Gemeinsam ist dem hier und dem von Feuerstack et al. [FdP11] präsentierten Ansätzen, dass die erstellten Dialoge mittels Gesten als Eingabemodalität getestet werden können und somit die Eingabemodalitäten nicht über Schaltflächen simuliert werden müssen.

Die in [FdP11] angeführten Beispiele lassen darauf schließen, dass jede Modalität und damit jede einzelne Geste zunächst programmtechnisch umgesetzt werden muss, bevor diese als konkreter Interaktor genutzt werden kann. Die Gesten im WeBewIn-Ansatz werden hingegen lediglich vorgemacht und können anschließend sofort im Kontext eines Dialogmodells genutzt werden. Beide Ansätze verfolgen das Ziel, Dialoge durch ausführbare Modelle untersuchen und testen zu können, wobei jeweils als Notation auf Zustandsdiagramme zurückgegriffen wird.

[1] Kinetic Space, Training and Recognizing 3D Gestures, https://code.google.com/p/kineticspace, letzter Zugriff: 28.06.13

[2] Gesture Authoring Tool, http://www.omekinteractive.com/products/beckon-usability-framework, letzter Zugriff: 28.06.13

Dumas et al. [DSL11] präsentieren einen grafischen UIDL (User Interface Description Language) Editor für die Modellierung multimodaler Interaktionen. Auf Basis ihrer Modellierungssprache SMUIML (Synchronized Multimodal User Interaction Modelling Language) werden die Modelle multimodaler Dialoge erstellt, welche auch hier als Zustandsautomat repräsentiert werden. Durch die Verwendung eines eigens entwickelten Frameworks der Autoren ist es möglich, verschiedene Eingabemodalitäten, wie z.B. Gesten- oder Spracherkennung, sowie verschiedene Ausgabemodalitäten an das Modell anzubinden. Auch das Ausführen des Modells wird durch das Framework ermöglicht. Insgesamt liegt in diesem Ansatz der Fokus auf der Beschreibungssprache und der Dialogmodellierung, weniger auf der Untersuchung einzelner Gesten und Gestenabfolgen. Im Gegensatz dazu zielt WeBewIn auf die werkzeuggestützte Untersuchung bewegungsbasierter Interaktion in frühen Entwicklungsphasen ab.

Zur Beschreibung und Untersuchung von Interaktionen mittels Datenhandschuh und Zeigegerät in virtuellen 3D-Welten haben Boeck et al. [Bo07] eine grafische Notation namens NiMMiT (Notation for Multimodal Interaction Techniques) entwickelt. Wie [LL12] zielt auch diese Arbeit darauf ab, den Entwicklungsaufwand für Benutzungsschnittstellen zu reduzieren. Mittels NiMMiT und den dazugehörigen Werkzeugen soll der Entwickler in der Lage sein, auf Modellebene verschiedene Lösungen für 3D-Welten zu erstellen. Die Notation basiert auf Zustandsdiagrammen und ist ereignis-, zustands- und datengesteuert, wobei das entstehende Modell durch einen Interpreter ausgeführt werden kann. Hier werden, im Gegensatz zu WeBewIn, das derzeit nur diskrete Interaktion berücksichtigt, auch kontinuierliche objektbezogene Gesten einbezogen. Zudem wird die Entwicklung von 3D-Welten unterstützt; inwiefern sich das Werkzeug für 2D-Anwendungen einsetzen lässt, ist nicht ersichtlich. Beiden Ansätzen ist jedoch gemein, dass sie ein frühzeitiges, schnelles Testen von Interaktionen ermöglichen.

3 Einführung in das WeBewIn-Werkzeug

WeBewIn, ein Entwicklungswerkzeug für bewegungsbasierte Interaktionen, besteht zurzeit im Kern aus dem Gesten-Editor, dem Dialog-Editor und dem Modell-Interpreter sowie den darin genutzten Modulen Gesten-Verwaltung, Gesten-Erkenner und Gesten-Player.

Gesten-Editor: Der Gesten-Editor ermöglicht das Aufzeichnen und Speichern von Gesten. Eine aufgezeichnete Geste kann modifiziert werden, zum einen indem per Selektion die Körperregionen bestimmt werden, die für die Ausführung und das Erkennen der Geste relevant sind. Zum anderen kann die aufgezeichnete Sequenz der Geste durch die Entfernung einzelner Frames zugeschnitten werden. Um eine Geste beschreiben und speichern zu können, werden die vom Microsoft Kinect SDK gelieferten Gelenkpositionen (Skelettdaten) aufgezeichnet und in einer Datei abgelegt. Hierzu werden diese Daten in sogenannten Frames organisiert. Ein Frame beinhaltet pro Gelenk dessen Raumkoordinaten zu einem bestimmten Zeitpunkt, wobei eine Sequenz von Frames eine Geste bildet.

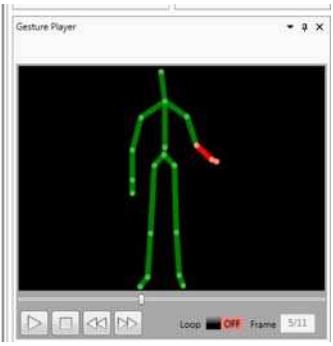
Dialog-Editor: Mittels dieser Anwendung können Dialogmodelle auf Basis der Notation von Zustandsdiagrammen erstellt werden. Eine Transition zwischen zwei Zuständen kann mit Ein- und Ausgabemodalitäten sowie mit Bedingungen versehen werden. Der Fokus bei den Eingabemodalitäten liegt derzeit auf Körpergesten, die durch den Gesten-Editor aufgezeichnet und im Dialog-Editor an die Transitionen gebunden werden.

Modell-Interpreter: Der Modell-Interpreter ist im Dialog-Editor integriert und für die Ausführung der erstellten Modelle zuständig. Über die einzelnen Zustände kann im Modell auf verschiedene Arten traversiert werden, worauf in Kapitel 4 genauer eingegangen wird. Des Weiteren ermöglicht der Modell-Interpreter das Anbinden und Steuern erster Mockups bis hin zu prototypischen grafischen Benutzungsschnittstellen.

Gesten-Verwaltung: Mit dieser Komponente, auch als Gesten-Katalog bezeichnet, werden die vorhandenen Gesten strukturiert verwaltet und in Gestensätzen organisiert. Hierzu bietet der Katalog eine Übersicht aller vorhandenen Gesten und die Möglichkeit, sich alle Elemente eines Gestensatzes auf einen Blick in animierter Form anzusehen.

Gesten-Erkenner: Der Erkenner basiert auf dem DTW (Dynamic Time Warping)-Verfahren (ähnlich [Wö12]) und ist zurzeit in der Lage diskrete Gesten zu erkennen. Er ist so konzipiert, dass er im Implementierungsprozess einer Anwendung als Bibliothek eingebunden werden kann, wie es auch beim Modell-Interpreter der Fall ist. Somit lassen sich aufgezeichnete Gesten auch in anderen Anwendungen nutzen. Für die Aufzeichnung der Gesten selbst wird dabei in WeBewIn derzeit der Microsoft Kinect-Sensor verwendet.

Gesten-Player: Bei verschiedenen Schritten in der Entwicklung werden die WeBewIn-Nutzer von einem Gesten-Player (Abb. 1a) mit den üblichen Funktionalitäten eines Videoplayers unterstützt. Dieser animiert zur Information über den Bewegungsablauf die Geste. Dabei werden Körperregionen, die für eine Geste relevant sind, farblich hervorgehoben. Dies ermöglicht es Entwicklern und Ergonomen, sich bereits in frühen Projektphasen über geplante Bewegungen auszutauschen und sie in Kombination mit anderen Bewegungen im Kontext kompletter Interaktionssequenzen zu beurteilen. Der Player wird auch für die Darstellung der Echtzeitdaten des Sensors benutzt, jedoch ist dann die Interaktionsleiste unter dem Videobild ausgeblendet und die farbliche Hervorhebung von Körperregionen entfällt.



(a) Gesten-Player

(b) Mockup des Beispiels „Bewertung eigener Fähigkeiten“

Abbildung 1: Gesten-Player und beispielhaftes Mockup

Zur Erläuterungen, wie WeBewIn mittels ausführbarer Modell zur Evaluation bewegungsbasierter Interaktion eingesetzt werden kann, wird in diesem Beitrag ein einfaches Beispiel verwendet. Dieses ist einer sich derzeit in der Entwicklung befindenden Anwendung zur Barrierefreiheit für Personen mit kognitiven Einschränkungen entnommen. Der hier exemplarisch gewählte Ausschnitt bezieht sich auf die Bewertung eigener Fähigkeiten. Wie anhand des Mockups in (Abb. 1b) dargestellt, werden pro Fähigkeit jeweils links die aktuelle Bewertung („wie gut kann ich das“) und rechts die noch zur Bewertung verfügbaren Sterne angezeigt. Je mehr Sterne die Anwender links anhäufen, desto stärker ist die abgebildete Fähigkeit ausgeprägt.

3.1 Beispiel zur Dialogmodellierung

Abbildung 2 enthält einen Ausschnitt des derzeit implementierten WeBewIn-Dialogeditors, der auf Zustandsdiagrammen basiert. Das im Bereich (a) gezeigte, exemplarische Dialogmodell „Bewertung eigener Fähigkeiten“ ist hier zur Erläuterung vereinfacht dargestellt. Es besteht lediglich aus den Dialogzuständen *rating not possible* und *rating* sowie sechs Transitionen. Eine Transition wird, wie bereits erwähnt, jeweils mit den für einen Dialogablauf relevanten Informationen versehen. Dies sind in der aktuellen Editorversion Benutzeraktionen (Action), die den Zustandsübergang anstoßen, z.B. *activate*, *deactivate* oder auch *next*, Bedingungen (im Editor als Constraint bezeichnet), die für einen Übergang gelten müssen, z.B. *Sterne noch verfügbar*, sowie Rückmeldungen (Feedback) als Folge eines Übergangs, wie etwa das Hervorheben der zur Bewertung ausgewählten Fähigkeit. In Abbildung 2 sind die Bedingungen jedoch zur weiteren Vereinfachung ausgeblendet, lediglich die Aktionen und Rückmeldungen sind „aufgeklappt“. Mit *previous* und *next* kann die vorherige bzw. nächste Fähigkeit ausgewählt, mit *increase* und *decrease* die Bewertung um einen Stern erhöht bzw. verringert werden. Das Symbol der Hand kennzeichnet die Aktion als Geste (analog zu den Rückmeldungen (Abb. 2b) sind hier verschiedene Modalitäten möglich).

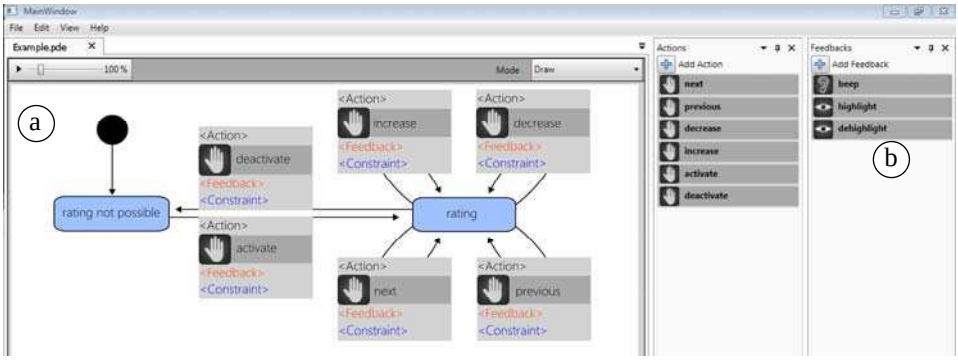


Abbildung 2: Dialogmodell für das vereinfachte Beispiel „Bewertung eigener Fähigkeiten“

Das Beispiel stellt ein abstraktes Dialogmodell dar, jedoch hätten auch konkrete Gestennamen, z.B. „Wischgeste“ verwendet werden können. Konkrete Namen vermitteln zumeist eine Vorstellung von der auszuführenden Bewegung. Dies mag in vielen, jedoch nicht in allen Fällen zutreffen. Zudem kann es sein, dass eine Gestenbezeichnung, wie z.B. hier „Wischgeste“, bei verschiedenen Personen unterschiedliche Vorstellungen über deren Ablauf hervorruft. Zuweilen müssen, so auch in unserer Beispielanwendung, zusätzlich auch körperliche Einschränkungen berücksichtigt und das Dialogmodell unabhängig von konkreten Gesten entwickelt werden.

Mittels des Gesten-Editors können nun, sofern noch nicht erfolgt, verschiedene Gesten spezifiziert und anschließend an das Dialogmodell gebunden werden.

3.2 Erfassen und Zuordnen von Gesten

Die erstmalige Erfassung einer Geste erfolgt über den Gesten-Editor durch Vormachen der Bewegung (eng. „by demonstration“) [BBHH13], [LL12]. Unter Nutzung des Gesten-Katalogs können Entwickler verschiedene Gestensätze erstellen und verwalten.

Der Editor bietet zusätzlich erste Funktionalitäten zur Modifikation von Gesten. Für jede demonstrierte Bewegung sind die für die Geste relevanten Gelenkpunkte zu definieren. So sollen z.B. in Abbildung 3 für die im Player gezeigte *increase*-Geste (eine von rechts nach links ausgeführte Wischbewegung mit der rechten Hand) die Kopf-, Bein- und Fußpositionen bzw. -bewegungen nicht berücksichtigt werden. Daher sind nur die übrigen, wenigen Gelenkpunkte des Unterarms ausgewählt (Abb. 2a). Zusätzlich (Abb. 2b) sind Basisfunktionalitäten zur Segmentierung einer Geste in Unter-Gesten vorhanden.

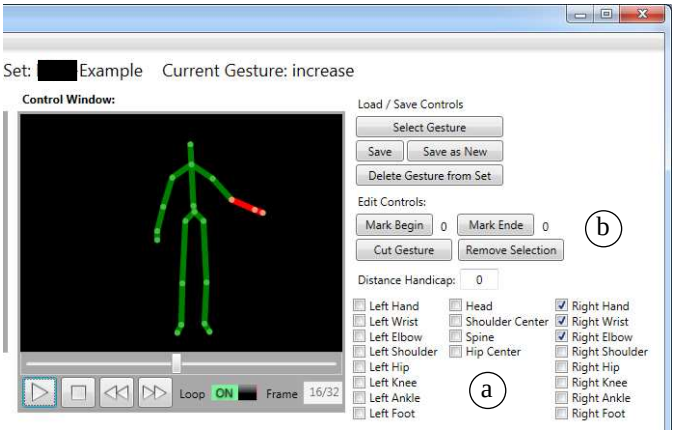


Abbildung 3: Ausschnitt Gesten-Editor

Ein erstellter Gestensatz kann nun im Dialog-Editor genutzt werden, indem dieser zunächst dem Dialogmodell zugeordnet wird. Zur Festlegung einer der Aktionen als Körpergeste wird diese aus dem Gestensatz ausgewählt und der Transition zugeordnet. Hierbei wird der Entwickler abermals von dem Gesten-Player unterstützt, der zur Information über den Bewegungsablauf die Geste animiert. Dies funktioniert sowohl aus dem Gestensatz heraus als auch für bereits den Transitionen zugeordnete Gesten.

Soll eine noch nicht erfasste Geste verwendet werden, kann deren Bezeichner festgelegt und einer Transition zugeordnet werden. Hiermit ist das Dialogmodell bereits interaktiv überprüfbar, auch wenn einzelne konkrete Posen oder Bewegungen noch offen sind. Insgesamt kann damit die Aufnahme bzw. Spezifikation einer Geste vor, während oder nach der Dialogmodellierung erfolgen.

4 Gestenevaluation auf Basis ausführbarer Dialogmodelle

WeBewIn ermöglicht die Evaluation bewegungsbasierter Interaktionen auf Basis ausführbarer Modelle und bietet hierzu verschiedene Möglichkeiten: Das Testen des Dialogmodells mit Gesten, die entweder von einer Person direkt ausgeführt oder als Aufzeichnung abgespielt werden, sowie die Evaluierung in Kombination mit ersten prototypischen Präsentationsentwürfen. Zusätzlich können einzelne Gesten als solche evaluiert werden und des Weiteren die ausführbaren Modelle ohne Gesten traditionell per Aktivierung von Schaltflächen gesteuert werden.

Evaluation einzelner Gesten: Sobald eine Geste mittels des Gesten-Editors erfasst wurde, kann diese bereits evaluiert werden, was unabhängig von den Modellen erfolgt. Zum einen kann unter Verwendung des Gesten-Erkenners die Geste von der vormachenden Person nun wiederholt ausgeführt werden, um sie etwa bezüglich der Anstrengung

und Eindeutigkeit zu bewerten. Zum anderen kann der Bewegungsablauf, unter Nutzung des Gesten-Players, an sich und im Vergleich zu anderen Gesten analysiert werden.

Testen des Dialogmodells ohne Gesten: Wie in Absatz 3.1 beschrieben kann das Dialogmodell bereits interaktiv, d.h. auf Basis ausführbarer Modelle getestet werden, bevor die einzelnen Gesten spezifiziert bzw. den Transitionen zugeordnet sind. Hierzu wird der Modell-Interpreter gestartet und die Gesten-Aktionen per Aktivierung von Schaltflächen simuliert. Das Modell kann somit untersucht werden, ohne dass zuvor Überlegungen bzgl. benötigter und geeigneter Gesten gemacht werden müssen.

4.1 Testen des Dialogmodells mit Gesten

Die nachträgliche Verknüpfung mit Gesten erfolgt durch Laden eines Gestensatzes in den Editor. Dabei ist in der aktuellen Version des Editors auf Namensgleichheit zwischen den vergebenen Aktionsbezeichnern und den Gestennamen zu achten. Andernfalls müssen die Aktionen, die bereits mit Transitionen verbunden sind, manuell ersetzt werden. Sobald eine Transition mit einer Geste versehen ist, kann sie bei der Ausführung des Modells zum Schalten der Transition genutzt werden. Um ein Modell mit einer weiteren Zusammenstellung von Gesten zu testen, ist es lediglich nötig, im Dialog-Editor den bereits zugeordneten Gestensatz durch einen anderen zu ersetzten. Auch hier ist auf Namensgleichheit zu achten, damit die neuen Gesten mit den bisherigen Aktionsbezeichnern verknüpft werden und so den Platz der bisherigen Gesten einnehmen können.

Der Modell-Interpreter erlaubt neben der oben erwähnten Simulation von Gesten-Aktionen per Aktivierung von Schaltflächen eine Evaluation durch das Erkennen aufgezeichneter Gesten, wobei zusätzlich erste Präsentationsentwürfe einbezogen werden können. Allen Vorgehensweisen ist das Durchwandern des Modells gemeinsam, indem entweder einzelne Aktionen oder gezielt spezifische Transitionen ausgewählt und vom Interpreter verarbeitet werden. Gelangt das Modell dabei in einen Folgezustand, spielt der Gesten-Player die zugehörige Geste ab.

Abspielen aufgezeichneter Gesten: Um die mit dem Modell spezifizierten Interaktionen besser beurteilen zu können, können die an den Transitionen stehenden Gesten über einen Eintrag im Kontextmenü im Player abgespielt werden (Abb. 4a). Während der Ausführung eines Modells wird jeweils die Geste abgespielt, die zu der aktuell gewählten Transition gehört. Die Transitionen können bei der Ausführung auch geschaltet werden, ohne dass die Gesten durch den Nutzer vorgemacht werden müssen. In diesem Fall kann das Erkennen einer Geste simuliert werden, indem die entsprechende Transition oder der gewünschte nächstmögliche Zustand über einen Mausklick aktiviert und die Transition damit geschaltet wird (Abb. 4b). Des Weiteren kann der Interpreter über ein Interaktions-Steuerelement bedient werden (Abb. 4c).

In dem Steuerelement werden die nächsten möglichen Transitionen mit den angebundenen Gesten angezeigt. Mit einem Mausklick auf eine der aufgelisteten Transitionen wird diese ausgeführt und die angebundene Geste wird im Gesten-Player abgespielt. Die

geplante Interaktion kann getestet werden, ohne immer wieder die Gesten vormachen zu müssen.

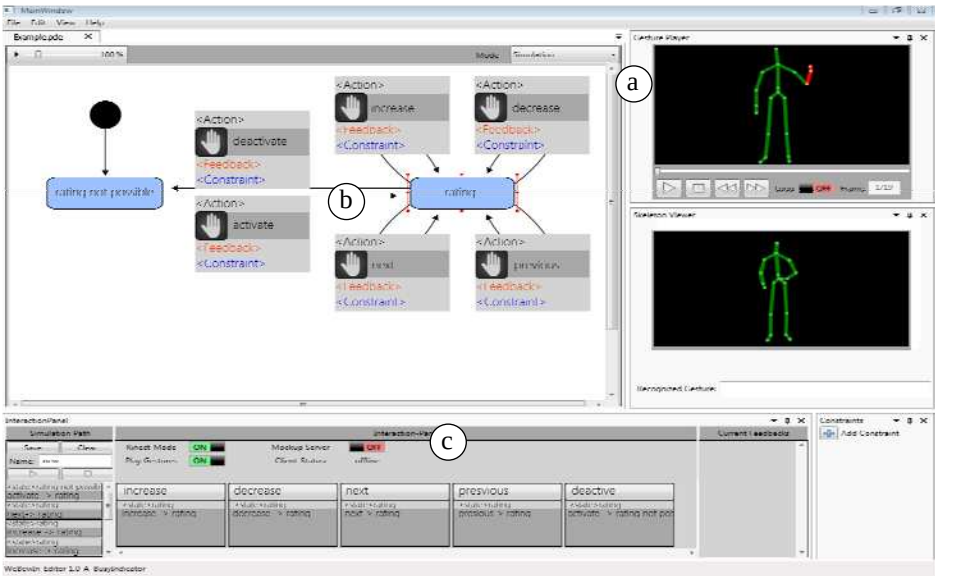


Abbildung 4: Modell-Interpreter im Dialog-Editor

Dem WeBewIn-Nutzer wird somit ein iteratives Modifizieren und Testen des Modells ermöglicht, ohne dass dabei eine Person erforderlich ist, die tatsächlich Gesten vor dem Sensor ausführt.

Ausführung realer Gesten: Wie in Abschnitt 3 bereits erwähnt, wird der Gesten-Erkenner nicht nur im Gesten-Editor sondern auch im Modell-Interpreter verwendet, wodurch reale Gesten bei der Ausführung eines Modells eingesetzt werden können. Hierbei stehen alle Funktionalitäten aus dem Abschnitt „Abspielen realer Gesten“ zur Verfügung.

Der WeBewIn-Nutzer kann beim Durchwandern des Modells die entstehenden Gestenabfolgen auf Probleme und mögliche körperliche Anstrengung hin untersuchen. Dabei kann ein erster realistischer Eindruck der modellierten Interaktion entstehen.

4.2 Anbindung von Präsentationsentwürfen

Die Untersuchung von Gesten und Gestenfolgen wird durch die Ausführung der erstellten Modelle ermöglicht. Das Testen einer Gestenfolge anhand eines Modells wirkt jedoch recht abstrakt. Prototypische Oberflächen, z.B. Mockups, werden häufig eingesetzt, um sich ein besseres Bild von der Interaktion mit einer Anwendung machen zu können. Daher kann der Einsatz eines grafischen Prototyps, der durch ein ausführbares Modell gesteuert wird, hilfreich sein, um die Interaktion zu konkretisieren (vgl. [BS96]).

In Abbildung 5 ist zu sehen, wie eine Testperson den Interpreter durch die Ausführung von Gesten steuert. Der obige Mockup zur „Bewertung eigener Fähigkeiten“ ist hier an das ebenfalls oben eingeführte Dialogmodell gebunden. Führt die Person in einem aktuellen Zustand eine erlaubte Geste aus, führt dies zu Veränderungen in der prototypischen Benutzungsschnittstelle.

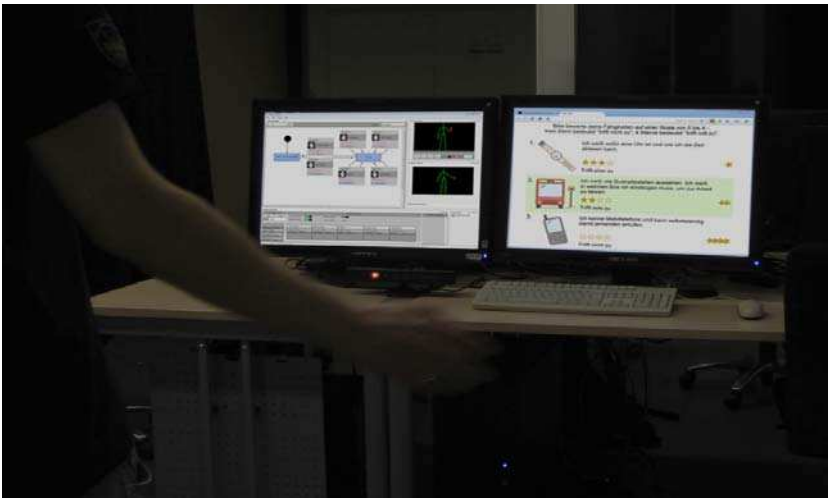


Abbildung 5: Steuerung eines Mockups (rechts) durch den Modell-Interpreter (links)

Um eine prototypische Oberfläche mit einem Modell zu verbinden, wurde in den Editor ein Websocket-Server integriert. Als Client kommen HTML5 Prototypen zum Einsatz, die mittels JavaScript eine Verbindung zum Server aufbauen. Damit ein Client durch das ausführbare Modell gesteuert werden kann, müssen die Aktionen des Modells auf die Interaktionselemente des Clients abgebildet werden.

Zurzeit befindet sich die Anbindung der Modell-Aktionen an die Client-Interaktionselemente noch in der Entwicklung und ist in der obigen Beispielanwendung nur prototypisch auf Programmierebene umgesetzt. In Zukunft sollen der Verbindungsprozess und die Abbildung der Aktionen über einen Dialog erfolgen, indem Aktionen des Modells auf die Interaktionselemente abgebildet werden können. Anschließend lässt sich der grafische Prototyp durch die Ausführung des Modells steuern.

Wenn eine Anwendung als Eingabemodalität Gesten nutzen soll, muss auch die Nutzeroberfläche auf diese Art der Interaktion ausgelegt werden. Herkömmliche interaktive Klick-Prototypen können dies nur bedingt, da sie meist auf andere, weit verbreitete Eingabemodalitäten (Maus, Tastatur) ausgelegt sind. Falls eine Gesteninteraktion zur Evaluation einer Nutzeroberfläche eingesetzt wird, muss sie meist aufwändig implementiert werden. Die Ausführung von Modellen mit der in diesem Abschnitt beschriebenen Technik besitzt alle Funktionalitäten eines Klick-Prototyps, mit dem Vorteil, dass die gewollte Eingabemodalität eingesetzt werden kann. Somit ist es möglich Gesten-

Interaktion und Nutzeroberfläche gleichzeitig zu entwerfen und aufeinander abzustimmen.

4.3 Szenarien

In allen der oben beschriebenen Varianten zum Testen können die verschiedenen, im Test durchlaufenen Interaktionssequenzen gespeichert (Abb. 6a) und zur Analyse wiederholt bzw. automatisch abgespielt werden (Abb. 6b). Beim Test des Modells mittels des Interpreters werden die durchlaufenen Zustände in einer Liste in Form einer Historie protokolliert (Abb. 6c). Jedes Listenelement beschreibt eine im Test ausgeführte Transition in Form des Ausgangszustands (hellgrau unterlegt), der ausgeführten Aktion und des Folgezustands (dunkelgrau). Die Historie in Abbildung 6 stammt aus einem Testdurchlauf des in Abschnitt 3.1 vorgestellten Dialogmodells. Aus dem ersten Element der Historie geht hervor, dass der Durchlauf in dem Zustand *rating not possible* startete und durch die Ausführung der Aktion *activate* der Folgezustand *rating* angenommen wurde. Daher war *rating* der Ausgangzustand des nächsten Elements bzw. der als nächstes ausgeführten Transition. Mit der Aktion *next* wurde der Zustand *rating* beibehalten usw. Die entstandene Historie kann über der Schaltfläche *Save* gespeichert und für weitere Analysezwecke wieder in den Interpreter geladen und abgespielt werden.

Es ist zudem möglich einzelne Gesten zu ersetzen und die Szenarien erneut – nun mit den alternativen Gesten – ablaufen zu lassen. Ebenso können die Testvarianten „Abspielen von Gesten“ und „Ausführung von Gesten“ sowie ein Triggern von Transitionen mit der Maus miteinander kombiniert verwendet werden. Eine solche Kombination kann beispielsweise eingesetzt werden, wenn die meisten Gesten für die geplante Benutzungsschnittstelle bereits geklärt sind und nur einige wenige noch untersucht werden müssen. Die Entwickler haben in dieser Situation die Möglichkeit, die Ausführung von Gesten zu simulieren, bis sie an eine Stelle im Modell kommen, an der die Gesteninteraktion noch verfeinert werden soll. Dies kann sinnvoll sein, um die körperliche Anstrengung während der Entwicklung zu verringern, da ein Benutzer nicht immer wieder dieselben Gesten vormachen muss.

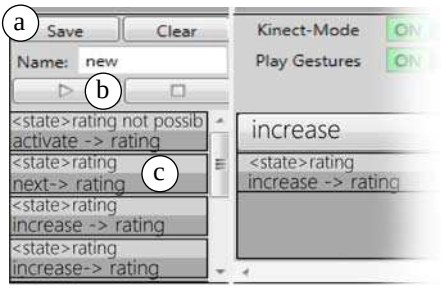


Abbildung 6: Aufgezeichnetes Szenario im Interaktions-Steuerelement

Die Transitionen können mit Bedingungen versehen werden (s. Abschnitt 3), die vor oder auch während des Testens auf *erfüllt* oder *nicht erfüllt* gesetzt werden können.

Hierüber können verschiedene Situationen „eingestellt“ und im Test berücksichtigt werden, entweder beim Durchspielen eines neuen oder beim erneuten Durchspielen eines aufgezeichneten Szenarios.

5 Diskussion

Gesten werden in WeBewIn derzeit immer als Ganzes betrachtet, d.h. jede Geste führt mit ihrem Erkennen zu einem Ereignis (diskrete Geste), das im Dialogmodell zu einem Zustandsübergang und damit im Mockup zu einer Veränderung bzw. einem Feedback führen kann. Feedback ist daher grundsätzlich erst nach vollständiger Durchführung einer Geste möglich, nicht bereits während der Ausführung. Jedoch können im Dialog-Editor kleinere Einheiten einer Geste spezifiziert und im Dialog-Editor jeder Teilgeste ein Feedback zugeordnet werden, um so simultane Rückmeldungen zu simulieren. Eine systematische Segmentierung bzw. Komposition von Gesten im Dialog-Editor und die Zuordnung eines simultanen Feedbacks zu den Teilgesten soll in den weiteren Arbeiten zu WeBewIn realisiert werden.

Da mit dem Entwicklungswerkzeug bisher nur diskrete Gesteninteraktion modelliert und untersucht werden kann, werden Szenarien, bei denen die Handposition als (Maus-) Zeigerersatz genutzt wird bzw. bei denen kontinuierliche Gesten zum Einsatz kommen, noch nicht unterstützt. In welcher Form kontinuierliche Interaktion zukünftig in WeBewIn eingehen wird, ist noch nicht endgültig geklärt. Erste Überlegungen gehen in die Richtung von [FdP11], wo die kontinuierliche Interaktion im Modell behandelt wird.

Der hier präsentierte Ansatz zielt auf eine frühe Benutzerpartizipation in der Entwicklung bewegungsbasierter Interaktionen ab. In eigenen Arbeiten zur benutzerzentrierten Gestenspezifikation wurde die Wizard-of-Oz Technik eingesetzt, um passende Gesten für die Steuerung einer virtuellen Anprobe zu finden [RBB13]. Jeder Proband dachte sich Gesten für einzelne Interaktionen aus und konnte diese dann anschließend zur Steuerung der Anwendung nutzen. Nachteilig bei dieser Vorgehensweise ist jedoch, dass ggf. einzelne ermittelte Gesten aufgrund technischer Einschränkungen der Gestenerkennung nicht oder nur ähnlich implementierbar sind. Dieser Nachteil entfällt bei den technik-basierten Ansätzen [NSMG04], in denen zunächst Gesten entsprechend der technischen Machbarkeit „gefunden“, realisiert und dann z.B. mittels Benutzertests evaluiert werden. Die technik-basierten Vorgehensweisen besitzen generell den Nachteil, dass in den Lösungen die Benutzer, ihre Eigenschaften und Bedürfnisse nur unzureichend berücksichtigt werden. WeBewIn unterstützt den Brückenschlag zwischen beiden Richtungen zu einem benutzer-technik-zentrierten Ansatz. Es ermöglicht die Spezifikation von Gesten, indem diese von zukünftigen Benutzern lediglich vorgemacht werden, wobei eine sofortige Überprüfung der Interaktionen auf Basis ausführbarer Dialogmodelle möglich ist. Der Wizard kann so in manchen Testsituationen durch das Werkzeug ersetzt werden. Vor allem hinsichtlich einer Gestenfindungsphase bzw. Gestenevaluationsphase ist es von Vorteil, dass Gesten „By-Demonstration“ erstellt und, auf Grund der Organisation in Gestensätzen, einfach ausgetauscht werden können.

Das Werkzeug besteht zurzeit noch aus zwei getrennten Anwendungen, die im weiteren Projektverlauf integriert werden. Gesten können dann direkt, d.h. ohne Einsatz eines separaten Gesten-Editors, aus dem Dialog-Editor heraus aufgenommen, einem Gestensatz zugeordnet und direkt an eine Transition angebunden werden.

Literaturverzeichnis

[BBS99] Biere, M.; Bomsdorf, B.; Szwillus, G.: The Visual Task Model Builder: Proceedings of the third international conference on Computer-aided design of user interfaces. Kluwer Academic Publishers, Norwell, MA, USA, 1999; S. 245-256.

[BS96] Bomsdorf, B.; Szwillus, G.: Early prototyping based on executable task models: Conference Companion on Human Factors in Computing Systems. ACM, New York, NY, USA, 1996; S. 254-255.

[BBHH13]Bomsdorf, B.; Blum, R.; Hesse, S.; Heinz, P.: WeBewIn: Rapid Prototyping bewegungsbasierter Interaktionen: Mensch & Computer 2013. Interaktive Vielfalt, Oldenbourg Verlag, 2013, zur Veröffentlichung.

[RBB13] Rupprecht, D.; Blum, R.; Bomsdorf B.: Toward A Gesture Set For A Virtual Try-On: IADIS International Conference: Interfaces and Human Computer Interaction 2013, 2013, zur Veröffentlichung.

[Ga03] Gaëlle Calvary et al.: A unifying reference framework for multi-target user interfaces. In INTERACTING WITH COMPUTERS, 2003, 15; S. 289-308.

[LL12] Lü, H.; Li, Y.: Gesture coder: a tool for programming multi-touch gestures by demonstration: Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems. ACM, New York, NY, USA, 2012; S. 2875-2884.

[Bo07] Boeck, J. De; Vanacken, D.; Raymaekers, C.; Coninx, K.: High-Level Modeling of Multimodal Interaction Techniques Using NiMMiT. In: Journal of Virtual Reality and Broadcasting, 2007, 4.

[DH93] Duke, D.; Harrison, M.: Abstract Interaction Objects. In: Computer Graphics Forum, 1993, 12; S. 25-36.

[DSL11] Dumas, B.; Signer, B.; Lalanne, D. Hrsg.: A Graphical UIDL Editor for Multimodal Interaction Design Based on SMUIML, Lisbon, Portugal, 2011.

[FdP11] Feuerstack, S.; dos Santos Anjo, M.; Pizzolato, E. B.: Modellierung und Ausführung von multimodalen Anwendungen auf Basis von Zustandsdiagrammen. In: i-com, 2011, 10; S. 40-47.

[MPV11] Meixner, G.; Paternò, F.; Vanderdonckt, J.: Past, Present, and Future of Model-Based User Interface Development. In: i-com, 2011, 10(3), Oldenbourg Verlag, S. 2-11.

[MPS02] Mori, G.; Paternò, F.; Santoro, C.: CTTE: support for developing and analyzing task models for interactive system design. In: IEEE Trans. Softw. Eng. 28, 8 (August 2002), S. 797-813.

[NSMG04] Nielsen, M.; Störring, M.; Moeslund, T. B.; Granum, E.: A Procedure For Developing Intuitive And Ergonomic Gesture Interfaces For Man-Machine Interaction. In (Camurri, A.; Volpe, G., Hrsg.): Gesture-Based Communication in Human-Computer Interaction - 5th International Gesture Workshop. Heidelberg: Springer, 2004, S. 409-420.

[RFD04] Reichart, D.; Forbrig, P.; Dittmar, A.: Task Models as Basis for Requirements Engineering and Software Execution. In: *Proceedings* of the 3rd annual conference on Task models and diagrams TAMODIA'04, 2004, New York, USA, ACM Press, S. 33-42.

[Wö12] Wölfel, M.: Kinetic Space. User Manual v1.2. Online verfügbar unter http://kineticspace.googlecode.com, 2012, zuletzt geprüft am 12.05.2013.