S. Boll, S. Maaß & R. Malaka (Hrsg.): Mensch & Computer 2013
München: Oldenbourg Verlag, 2013, S. 251– 260

WeBewIn: Rapid Prototyping bewegungsbasierter Interaktionen

Birgit Bomsdorf, Rainer Blum, Sebastian Hesse, Patrik Heinz

Angewandte Informatik, Hochschule Fulda

Zusammenfassung

In diesem Beitrag werden erste Ergebnisse in der Entwicklung eines Werkzeugs für bewegungsbasierte Interaktionen (WeBewIn) vorgestellt. Es erlaubt die Spezifikation von im Raum ausgeführten Körpergesten (Posen und Bewegungsabläufe) mittels Vormachen (By-Demonstration). Die so erfassten Gesten können, mit oder ohne Nachbearbeitung, sofort an ein Dialogmodell gebunden und evaluiert werden. Auf Basis ausführbarer Modelle können komplexe Gestenabläufe im Kontext verschiedener Interaktionssequenzen überprüft werden. WeBewIn ermöglicht so ein Rapid Prototyping von Gesteninteraktionen in einer kombinierten benutzer- und technik-basierten Vorgehensweise.

1 Einleitung

Interaktive Systeme werden zunehmend über Gesten bzw. mittels Körperbewegungen gesteuert, deren Vorteil in den natürlicheren Interaktionsmöglichkeiten gesehen wird. Zwar sind bewegungsbasierte Interaktionen nicht generell von Vorteil, doch gibt es spezielle Nutzungskontexte und Applikationen, in denen sie die Usability und UX zu steigern vermögen. In der Entwicklung entsprechender Gestensätze können Nielsen et al. (2004) folgend zwei grundsätzliche Vorgehensweisen unterschieden werden. Zum einen sind dies die technik-basierten Ansätze, in denen zunächst Gesten entsprechend der technischen Machbarkeit „gefunden“, realisiert und dann z.B. mittels Benutzertests evaluiert werden. Dem stehen die benutzerbasierten Ansätze gegenüber, in denen die Gesten direkt unter Einbezug potentieller Benutzer und unabhängig von einer Technologie zur Erkennung spezifiziert werden. Methodisch kommen bei letzteren beispielsweise Wizard of Oz-Studien oder Videoaufzeichnungen zum Einsatz (Nielsen et al. 2004, Höysniemi et al. 2004). Nachteilig ist jedoch, dass ggf. einzelne ermittelte Gesten aufgrund technischer Einschränkungen der Gestenerkennung nicht oder nur ähnlich implementierbar sind. Demgegenüber gehen die technik-basierten Ansätze mit den für sie generellen Nachteilen einher, dass in den Lösungen die Benutzer, ihre Eigenschaften und Bedürfnisse nur unzureichend berücksichtigt sind.

Die Sichtweisen beider Ansätze sind notwendig. Die Entwicklung benutzerzentrierter Gestensätze, die bereits jetzt mit aktuell verfügbaren Technologien einsetzbar sein sollen, muss auch deren Einschränkungen berücksichtigen. Zur Kombination beider, der benutzer- und der technik-basierten Vorgehensweisen realisieren wir derzeit ein Entwicklungswerkzeug für

bewegungsbasierte Interaktionen (WeBewIn). 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 derzeitige Fokus liegt dabei auf einer Testunterstützung sehr früher Entwicklungsschritte. Im folgenden Kapitel 2 werden zunächst verwandte Arbeiten mit engem Bezug zum eigenen Ansatz vorgestellt. Anschließend (Kapitel 3) werden anhand eines Beispiels wesentliche Eigenschaften von WeBewIn präsentiert. Nach einem kurzen Einblick in die technische Umsetzung gehen wir auf die aktuelle und zukünftige Werkzeugentwicklung ein (Kapitel 4).

2 Stand der Technik

Im Zentrum der modellbasierten Entwicklung von Benutzungsschnittstellen steht die Spezifikation der mit dem System durchzuführenden Aufgaben, der darauf basierenden Dialoge und der zugehörigen Präsentation (Meixner et al. 2011). Die Ausführbarkeit der dabei entstehenden Modelle erlaubt eine frühzeitige Überprüfung erster Designentscheidungen. Entsprechende Werkzeuge (z.B. (Biere et al. 2002), (Mori et al. 2002), (Reichart et al. 2004)) 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. Diesen Ansatz zum frühzeitigen Testen verfolgen wir ausgehend von unseren früheren Arbeiten (Biere et al. 2002) ebenfalls in WeBewIn, wobei der derzeitige Fokus auf einer werkzeugtechnischen Unterstützung einer frühzeitigen Gestenermittlung und -evaluation auf der Ebene der Dialogmodellierung liegt.

Auf Dialogebene werden vielfach, so auch in WeBewIn und in (Feuerstack et al. 2011), Zustands-Transitions-Diagramme eingesetzt, wie sie auch in UML verwendet werden. Im Vordergrund stehen dabei die Dialogzustände und -übergänge sowie die auslösenden Ereignisse und die Situationen, unter denen ein Übergang (Transition) erfolgen darf. Zur Spezifikation multimodaler Dialoge werden mit den Ereignissen zusätzlich die jeweiligen Modalitäten (Sprache, Geste etc.) verknüpft. Zur Überprüfung gestenbasierter Dialoge gehen das Werkzeug von Feuerstack et al. (2011) und WeBewIn einen Schritt weiter als bisherige Arbeiten, indem die Gesten nicht per Aktivierung von Schaltflächen simuliert, sondern real durchgeführt werden können. In (Feuerstack et al. 2011) erfolgt die Spezifikation der Gesten (nur Posen, keine Bewegungsabläufe) und der Dialoge separat. Anschließend wird in einem expliziten Modellierungsschritt die Gestenerkennung an das ausführbare Dialogmodell gebunden. In WeBewIn verfolgen wir hingegen eine technisch integrierte Lösung, die auch während einer Modellsimulation die Spezifikation von Gesten und deren Nutzung im nächsten Schritt des aktuellen Testlaufs erlaubt.

Zur Spezifikation von Posen und Bewegungsabläufen existieren verschiedene formale Gestenbeschreibungssprachen. Ein sehr früher Ansatz hierzu ist die Labanotation (Hutchinson 1977), die zur Aufzeichnung und Analyse von Tanz-Choreographien entworfen wurde, aber auch Arbeiten in der MCI beeinflusst hat, z.B. (Loke et al. 2005) und (Gockel et al 2012). Durch die explizite Berücksichtigung qualitativer Aspekte, wie Geschwindigkeit und Beschleunigung, definieren die formalen Beschreibungen eine Geste zumeist präziser als dies mit einer Demonstration der Geste erfolgen kann. Zunehmend XML verwendend, er-

leichtern die Formalisierungen den werkzeugübergreifenden Austausch der Spezifikationen und eine Integration in den modellbasierten Entwurf. Ebenso wird die animierte Visualisierung der Bewegungen vereinfacht (Wilke et al. 2005), die im Bereich der MCI der ergonomischen Evaluation dienen kann. Bisherige Werkzeuge unterstützen dies jedoch nicht im Kontext der auszuführenden Dialoge. Auch in (Feuerstack et al. 2011) müssten die Benutzer für eine (dann spätere) Analyse per Vido aufgezeichnet werden. WeBewIn setzt hingegen auf das By-Demonstration Konzept, wodurch die Gesten als Aufzeichnung sofort für Evaluationen verfügbar sind.

Nach unserem Kenntnisstand existieren zurzeit zwei Werkzeuge, die das By-Demonstration Konzept zur Spezifikation von 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.

3 Werkzeug für bewegungsbasierte Interaktionen

Zur Beschreibung wesentlicher Eigenschaften des WeBewIn-Werkzeugs dient uns im Folgenden ein sehr vereinfachtes Beispiel, das einem aktuellen Projekt zur Barrierefreiheit entnommen ist: Ein Dialog zur Bewertung eigener Fähigkeiten. Wie in Abbildung 1 (a) gezeigt, werden pro Fähigkeit jeweils links die aktuelle Bewertung („wie gut kann ich das“) und rechts die noch zur Bewertung verfügbaren Sterne angezeigt.

[1] Kinetic Space, Training and Recognizing 3D Gestures, https://code.google.com/p/kineticspace, Zugriff: 28.6.13

[2] Gesture Authoring Tool, http://www.omekinteractive.com/products/beckon-usability-framework, Zugriff: 28.6.13

3.1 Dialogmodellierung

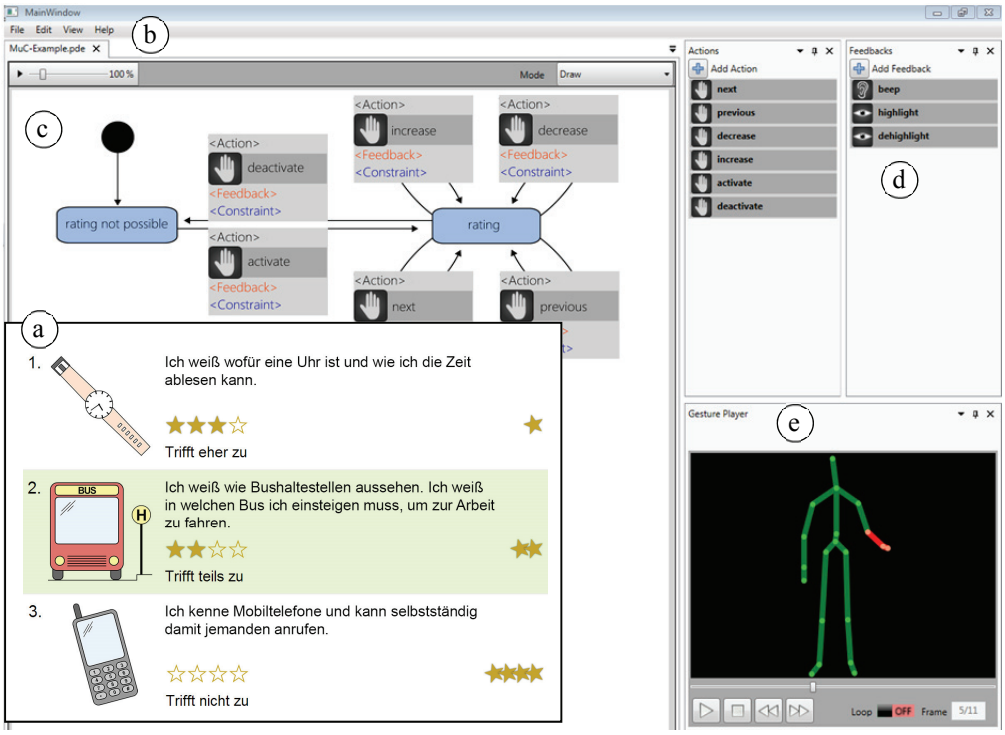

Abbildung 1: Dialog-Modell für das gewählte Anwendungs-Beispiel

Abbildung 1 (b) zeigt einen Ausschnitt des derzeitigen WeBewIn-Dialogeditors, der eine Notation für Zustands-Transitions-Diagramme verwendet (Abb.1 c). Das Dialogmodell besteht hier lediglich aus den beiden Dialogzuständen *Bewerten nicht möglich* und *Bewerten* sowie sechs Transitionen. Eine Transition wird jeweils mit den für einen Dialogablauf relevanten Informationen versehen. Dies sind in der aktuellen Editorversion Benutzeraktionen (Action), die den Zustandsübergang bewirken, z.B. *activate*, *deactivate* oder auch *next*, Bedingungen (im Editor als Constraints 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 1 (c) sind die Bedingungen und Rückmeldungen jedoch zur weiteren Vereinfachung ausgeblendet, lediglich die Aktionen 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.1 d) sind hier verschiedene Modalitäten möglich).

Zur Festlegung einer Benutzeraktion als Körpergeste kann diese aus einem bereits spezifizierten Gestensatz ausgewählt und der Transition zugeordnet werden. Hierbei wird der Entwickler von einem Gesten-Player (Abb.1 e) mit den üblichen Funktionalitäten eines Videoplayers unterstützt, der zur Information über den Bewegungsablauf die Geste animiert. Dies funktioniert sowohl aus dem Gestensatz heraus als auch für bereits den Transitionen zuge-

ordneten Gesten. Die Körperregionen, die für eine Geste relevant sind, werden dabei 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.

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. Die Spezifikation einer Geste kann damit vor, während oder nach der Dialogmodellierung erfolgen.

3.2 Gestenspezifikation

Gesten werden mit dem WeBewIn-Werkzeugs im Gesteneditor (Abbildung 2) mittels Vormachen, dem sog. By-Demonstration, spezifiziert. Hierdurch ist keine explizite Gesten-Spezifikation durch den Entwickler nötig, vielmehr werden die vom Microsoft Kinect SDK gelieferten Gelenkpositionen (Skelettdaten) aufgezeichnet und in einer Datei abgelegt.

Abbildung 2: Gesten-Editor

Alle Bewegungen der Person, die Gesten demonstriert, werden live in einem eigenen Bereich (Abb.2 a) dargestellt. Diese Anzeige dient der Kontrolle, ob die Person sich im Bildbereich des Aufnahmesensors befindet. Die Aufzeichnung einer Geste erfolgt für eine einstellbare Anzahl von Frames, wobei die Microsoft Kinect in der Regel 30 Frames pro Sekunde aufnimmt. Ein verzögerter Start der Aufzeichnung ermöglicht die Benutzung des Gesteneditors auch durch eine einzelne Person, indem dieser ihr Zeit gibt, sich zunächst korrekt vor dem Sensor zu positionieren.

Ist die Aufzeichnung einer Geste beendet, wird sie gespeichert, z.B. unter einem vorher im Dialogmodell festgelegten Namen; in unserem Beispiel etwa unter dem Namen *increase*. Anschließend, da sie nun als Gestenvorlage dem Erkenner bekannt ist, kann die neue Geste

sofort getestet werden. Hierzu wird sie erneut ausgeführt. Der Editor informiert in einem separaten Bereich laufend über erkannte Gesten (Abb.2 d). Zur zuletzt erkannten Geste, im Beispiel ist dies die *increase*-Geste, zeigt er hier den Gestennamen, den Zeitpunkt der Erkennung und den Distanzwert an. Je kleiner der Distanzwert, umso ähnlicher war die durchgeführte Bewegung der erkannten Geste. Varianzen gibt es hier immer, da selbst derselbe Benutzer eine bestimmte Geste nur in ähnlicher Weise, aber nicht bewegungsidentisch ausführen kann.

Zudem kann eine gerade vom Benutzer demonstrierte und aufgezeichnete Geste im Gesten-Player (Abb.2 b) abgespielt werden. Sofern dies nicht bereits vor der Aufnahme erfolgte, sind vor der Speicherung noch die für die Geste relevanten Gelenkpunkte zu definieren. So sollen z.B. 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 der Unterarme ausgewählt. Ohne die Spezifikation der relevanten Gelenkpunkte würde die Gestenerkennung in ihrer Analyse immer den gesamten Körper einbeziehen. Hätte die Person bei der Aufnahme einer Wischgeste zufällig die Beine weit auseinander gehabt, müssten für die Erkennung dieser Wischgeste die Beine immer ähnlich weit auseinander positioniert werden. Als weitere Editiermöglichkeit kann die Aufnahme Frame-genau zugeschnitten werden (Abb.2 c), um irrelevante Sequenzen zu entfernen.

Mit dem Editor können Entwickler Gestensätze erstellen und im Dialogeditor nutzen (vgl. Absatz 1), um einzelne Gesten oder komplette Gestensätze einem Modell hinzuzufügen oder auszutauschen und zu testen. Ein zusätzlicher Gestenkatalog ermöglicht es, die Menge der vorhandenen Gesten strukturiert zu verwalten. Hierzu bietet er eine Übersicht aller vorhandenen Gesten und die Möglichkeit, sich alle Elemente eines Gestensatzes auf einen Blick in animierter Form anzusehen. Einzelne Gesten können jederzeit wie oben beschrieben bearbeitet und getestet werden.

3.3 Interaktives Testen

WeBewIn ermöglicht ein Rapid Prototyping auf Basis ausführbarer Modelle und bietet in seinem Simulator hierzu verschiedene Möglichkeiten. Wie in Absatz 1 beschrieben kann das Dialogmodell bereits interaktiv getestet werden, bevor die einzelnen Gesten spezifiziert sind. Die nachträgliche Verknüpfung mit Gesten erfolgt durch Laden eines Gestensatzes in den Editor. Dabei ist in der aktuellen Version auf Namensgleichheit zwischen den für die Aktionen vergebenen Bezeichnern und den Gestennamen zu achten.

Sobald die Gesten mit dem Dialogmodell verknüpft sind, können sie im Test direkt mitberücksichtigt und damit auch als Gestenabfolge evaluiert werden. Abbildung 3 zeigt diesen Fall für das gewählte Anwendungsbeispiel.

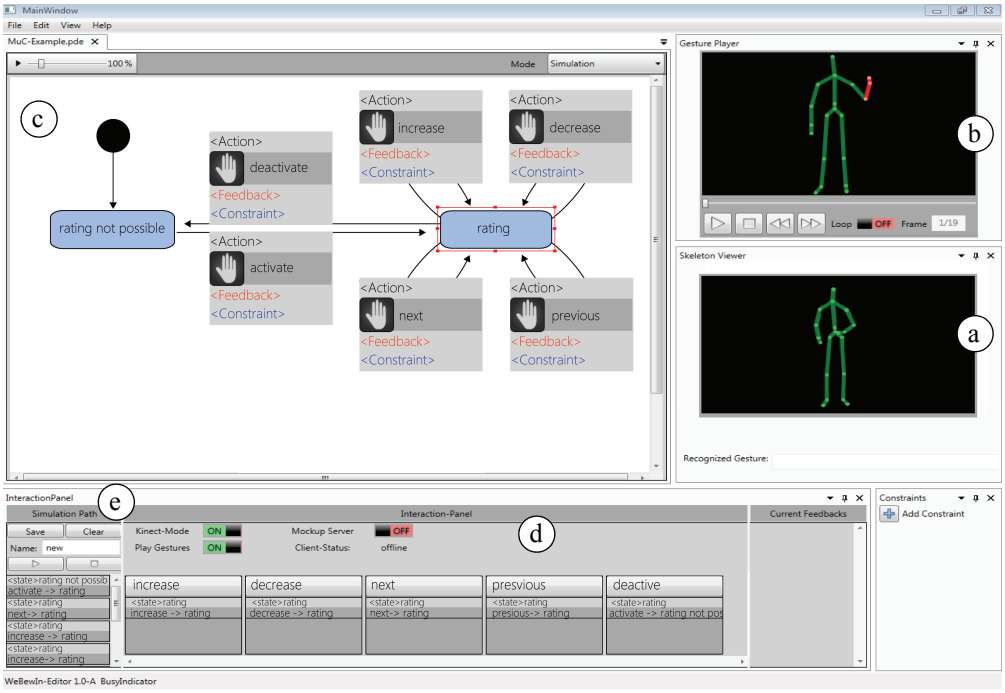

Abbildung 3: Interaktives Testen eines Dialog-Modells mit Gesten-Interaktionen

Um das Dialogmodell interaktiv zu durchlaufen, führt ein möglicher Benutzer die für die gewünschte Transition vorgesehene Aktion bzw. Geste aus. Dadurch wird der jeweilige Zustandsübergang ausgelöst und die Simulation wechselt in den nächsten Zustand. Die am Test beteiligten Personen können auch hier die aktuellen Bewegungen über die Live-Wiedergabe nachvollziehen (Abb.3 a). Bei Bedarf können sie sich die mit einer Aktion verknüpfte Geste im Gesten-Player vorspielen lassen (Abb. 3 b), z.B. um sie sich wieder in Erinnerung zu rufen. Zudem besteht die Option, durch einen Wechsel in den Gesteneditor einzelne Gesten zu modifizieren oder neu aufzunehmen.

Eine weitere Möglichkeit, das Dialog-Modell interaktiv zu testen, funktioniert über das schrittweise Auswählen aufeinanderfolgender Zustände im Diagramm mit der Maus und damit ohne Verwendung der Gestenerkennung.

Die in einem aktuellen Zustand (*rating* in Abb.3 c) jeweils erlaubten Aktionen, inkl. Angabe der Folgezustände, werden zusätzlich in einem eigenen Bereich aufgeführt (Abb.3 d). Damit sind die für einen gültigen, nächsten Schritt wichtigen Informationen gefiltert auf einen Blick verfügbar. Eine hier aufgeführte Aktion kann alternativ zur tatsächlichen Ausführung der Geste auch per Mausklick ausgelöst werden. Dies stellt eine weitere Möglichkeit der interaktiven Simulation des Dialog-Modells dar und bietet ein intensives Testen des Dialogmodells ohne ein ggf. ermüdendes Wiederholen einzelner Bewegungen.

Unabhängig von der verwendeten Simulationsmethode wird der aktuelle Zustand jeweils im Diagramm markiert und mögliche nächste sowie bereits durchlaufene Transitionspfade farblich hervorgehoben. Jeder im Rahmen eines Simulationsdurchlaufs traversierte Zustand und jede Bedingungsänderung werden in einer Historie aufgelistet (Abb.3 e). Sie kann zum

Zweck einer genaueren Analyse gespeichert und wieder abgespielt werden. Dabei werden nacheinander auch die mit den Aktionen verknüpften Gesten im Gesten-Player abgespielt.

3.4 Technische Umsetzung der Gestenerkennung

Die Erkennung der Gesten ist mit einem selbst entwickelten Gestenerkenner umgesetzt. Er nutzt das Microsoft Kinect for Windows SDK und verarbeitet die von diesem bereitgestellten Positionsdaten zu verschiedenen Skelettpunkten weiter. Dabei kommt, wie auch beim oben genannten Kinetic Space Werkzeug, der Dynamic Time Warping (DTW) Algorithmus[3] zum Einsatz, der zeitliche Variationen beim Vergleich zweier Gestensequenzen eliminiert. Räumliche Varianzen werden, vereinfacht ausgedrückt, aufsummiert und über Grenzwerte bewertet. Durch die starken Abweichungen in der Ausführung von Gesten, selbst bei derselben Person, ist beides entscheidend. Zunächst wird aber jeweils eine einfache Form des Gesture Spottings angewendet: Jedes pro Frame aufgenommene Skelett wird mit dem letzten Skelett aller aktuell in den Gestenerkenner geladenen Gestenvorlagen abgeglichen und analysiert, ob es sich dabei um den letzten Frame einer dieser Gestenvorlagen handeln könnte. Das im positiven Fall anschließend angewendete, verhältnismäßig rechenintensive DTW-Verfahren liefert dann wiederum Werte, die noch weiterer Interpretationen bedürfen. Anhand derer wird schließlich entschieden, ob eine bekannte Geste aufgetreten ist.

Für einzelne Dialoge werden in Abhängigkeit von den aktuellen Zuständen zusammengestellte Gestensätze im Gestenerkenner aktiviert bzw. geladen, da nicht zu jedem Zeitpunkt bzw. in jedem Dialogschritt alle Gesten erkannt werden müssen. Zudem verringert es die Wahrscheinlichkeit einer Fehlkennung, wenn sich nur die relevanten Gesten im aktuellen Katalog des Gestenerkenners befinden.

4 Diskussion und Ausblick

Auf dem Weg zu Entwicklungswerkzeugen für bewegungsbasierte Interaktionen ist derzeit noch eine Vielzahl von Fragestellungen zu klären. Es ist zudem offen, welche Konzepte (theoretisches Wissen, Methoden) Designern und Entwicklern an die Hand gegeben werden müssen, wie diese in einem Werkzeug als praktikabel nutzbare Funktionalitäten angeboten werden können und wie sie mit Ansätzen des etablierten Usability- und Software-Engineerings integriert werden können. Erschwerend kommt hinzu, dass aufgrund der Neuartigkeit der Thematik die Zielgruppe der Entwickler noch keine konvergierenden Anforderungen an ein sie unterstützendes Werkzeug formulieren kann. Wie auch bei früheren Innovationen werden sich diese erst mit der Zeit bilden können, was durch die frühzeitige Entwicklung erster dedizierter, unterstützender Werkzeuge – wie die in (Feuerstack 2004) und in diesem Beitrag beschriebenen – signifikant vorangetrieben werden kann.

So wurden im Laufe der bisherigen WeBewIn-Entwicklung bereits Rückmeldungen von Anwendungspartnern eingeholt. Im Wesentlichen bestätigte sich dabei der hohe Nutzen des Werkzeugs, auch wenn es sich derzeit erst im Entwicklungsstadium eines Prototyps befindet. Der Gesten-Editor und der Dialogeditor mit Simulator liegen zurzeit als zwei separate An-

[3] http://www.inf.fu-berlin.de/lehre/WS98/SprachSem/culjat/node4.html, Zugriff: 28.06.13

wendungen vor. Dies resultiert lediglich daraus, dass beide zunächst getrennt zu realisieren waren. Einzelne Funktionalitäten werden dank des modularen Aufbaus bereits gegenseitig genutzt, wie etwa der Gesten-Player. Die angestrebte Integration der beiden Benutzungsschnittstellen wird die Entwicklungsabläufe noch weiter vereinfachen und flexibilisieren, z.B. die angestrebte Möglichkeit zwischen Dialogmodellierung, Gestenspezifikation und Simulation auch spontan zu wechseln.

Das umgesetzte By-Demonstration Konzept erlaubt die Spezifikation von Gesten durch Vormachen. Der Editor bietet erste Möglichkeiten zu deren Nachbearbeitung. Unsere Anwendungspartner und wir selbst sehen hier die Notwendigkeit, Gestenvorlagen präziser zu erfassen bzw. nachbearbeiten zu können. Beispielsweise sind qualitative Aspekte, wie Geschwindigkeit und Beschleunigung, neben der technischen Realisierung auch aus ergonomischer Sicht sehr relevant.

Die realisierten Simulationsalternativen unterstützen verschiedenartige Vorgehensweisen des Testens, insbesondere mit Vormachen der Gesten, etwa zur Überprüfung der Ermüdung, und ohne Ausführung der Gesten, zur Überprüfung der Dialogabläufe, ohne dass es für eine Person anstrengend wird. Jedes durchspielte Szenario kann gespeichert, modifiziert und wieder abgespielt werden. Hierbei unterstützt der Player die Analyse alternativer Gestenfolgen für einen Dialog. Insgesamt wünschen sich die Anwendungspartner die Integration mit einer Usability Test Suite, so dass umfassend werkzeugunterstützte Usability Tests unterschiedlicher Ausprägung und mit reichhaltiger Datensammlung und -auswertung möglich werden. Mehrmals wurde dabei auch die sich bereits in der Entwicklung befindende Einbindung erster Entwürfe der Präsentation, etwa durch Mockups genannt. Damit werden Veränderungen der Präsentation in Abhängigkeit von Benutzeraktionen und den resultierenden Zustandsänderungen testbar, s. auch (Biere et al. 2002). Für das in Kapitel 3 eingeführte Beispiel liegt bereits ein Demonstrator vor, der mit Gesten gesteuert werden kann.

Die in WeBewIn implementierte Komponente zur Gestenerkennung kann in spätere Endanwendungen eingebunden werden. Dies stellt sicher, dass die in einer benutzerzentrierten Vorgehensweise entwickelten Gestensätze auch tatsächlich in der Zielanwendung einsetzbar sind. Diese und WeBewIn unterliegen dann denselben Möglichkeiten und Einschränkungen der verwendeten Technologien, wodurch eine kombinierte technik- und benutzer-basierte Vorgehensweise in der Entwicklung von Gestensätzen unterstützt wird. Jedoch werden in WeBewIn derzeit nur diskrete Gesten betrachtet. Im Kontext von Low-Fidelity Prototypen ist dies vielfach ausreichend, da für Testzwecke kontinuierliche Gesten gut mittels diskreter abgebildet werden können. Die zukünftigen Arbeiten an WeBewIn sollen diese Einschränkungen beseitigen und auch kontinuierliche Gesten mit direktem Objektbezug einschließen.

Literaturverzeichnis

Baron, M.; Lucquiaud, V.; Autard, D.; Scapin, DL. (2006). K-MADe: un environement pour le noyau du modèle de description de l'activité. Proceedings of the 18th French-speaking conference on Human-Computer Interaction.

Biere, M.; Bomsdorf, B.; Szwillus, G. (1999). The Visual Task Model Builder. In: Vanderdonckt, Jean; Puerta, Angel R. (Hg.): Computer-Aided Design of User Interfaces II, Proceedings of the Third International Conference of Computer-Aided Design of User Interfaces, Kluwer, S. 245-256.

Feuerstack, S., Anjo, MDS., & Pizzolato, EB (2011). *Model-based design and generation of a gesture-based user interface navigation control*. In Proceedings of the 10th Brazilian Symposium on on

Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction (IHC+CLIHC '11). Brazilian Computer Society, S. 227-231.

Höysniemi, J., Hämäläinen, P. & Turkk, L. (2004). *Wizard of Oz prototyping of computer vision based action games for children*. In Proceedings of the 2004 conference on Interaction design and children: building a community (IDC '04). New York, USA, S. 27-34.

Hutchinson, A. (1977). *Labanotation or Kinetography Laban: The System of Analyzing and Recording Movement*, 3. Ausgabe, Theatre Arts Books, New York.

Gockel, B., Staab, T., Bomsdorf, B. (2012). *Benutzerzentrierte Beschreibung bewegungsbasierter Interaktionen*. In Mensch & Computer 2012, Oldenbourg Verlag, S. 363-364.

Loke, L., Larssen, A.T. & Robertson, T. (2005). Labanotation for design of movement-based interaction. In Pisan, Y. (Hrsg.): *Proceedings of the second Australasian conference on Interactive entertainment.* Creativity & Cognition Studios Press, Sydney, Australia, Australia, 113-120.

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

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

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

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

Wilke, L., Calvert, T., Ryman, R., Fox, I. (2005). *From dance notation to human animation: The LabanDancer project*, In Journal of Comput. Animat. Virtual Worlds 16, 3-4, John Wiley & Sons, S. 201-211.

Kontaktinformationen

Birgit Bomsdorf, Fachbereich Angewandte Informatik, Hochschule Fulda, Marquardstraße 35, 36039 Fulda, bomsdorf@hs-fulda.de.