Prototyping von intuitiven und interaktiven Benutzerschnittstellen

Schnelles und einfaches Design von Anwendungen mit virtuellen Agenten

P. Kulms, H. v. Welbergen, S. Kopp
Universität Bielefeld, AG Kognitive Systeme und soziale Interaktion, Technische Fakultät
Exzellenzcluster Cognitive Interaction Technology (CITEC)
Inspiration 1, 33619 Bielefeld
pkulms@techfak.uni-bielefeld.de, hvanwelbergen@techfak.uni-bielefeld.de,
skopp@techfak.uni-bielefeld.de

Kurzzusammenfassung
Virtuelle Agenten ermöglichen intuitive Kommunikation. Durch Sprache, Gestik und Mimik entsteht in der Interaktion ein sozialer Charakter, der zum Beispiel in Assistenzsystemen von Vorteil ist [1]. Wir präsentieren ein Tool für das Kombinieren unseres virtuellen Agenten Billie mit normalen Benutzerschnittstellen-Elementen, so dass entstehende Interfaces über Natürlichkeit, Robustheit und Exaktheit verfügen. Unsere Lösung bietet Designern eine Zustandsgraph-basierte Modellierung der Interaktionslogik und nutzt Qt für das Benutzerschnittstellen-Design. Damit versuchen wir den Programmieraufwand für das Design zu reduzieren. Eine Nutzerstudie mit acht Teilnehmern zeigt, dass es trotz geringer Erfahrung mit den Komponenten möglich ist, einen beispielhaften Prototypen zu vervollständigen.

Abstract
Prototyping of intuitive and interactive user interfaces: Easily designing virtual agent applications – Through the use of speech, gestures, and facial expressions the interaction with a virtual agent user interface is intuitive and social which is advantageous for virtual assistive systems [1]. Our tool enables the combination of the virtual agent Billie with regular user interface elements, such that resulting interfaces feel natural, robust, and accurate. Our solution provides interaction modeling based on statecharts and uses Qt for interface design, thus reducing the programming effort. A user study with eight participants shows the tool’s feasibility, even with only little experience with the components.

1 Einleitung

Virtuelle Agenten sind interaktive Assistenten, die in vielfältigen Szenarien Anwendung finden. Sie werden beispielsweise als Tutoren in Lehr- und Lernumgebungen, Museumsführer, Therapeuten, Figuren in Computerspielen, oder als soziale Begleiter eingesetzt. Je nach Anwendungsgebiet ergeben sich distinkte Anforderungen an den Input des Agenten, zum Beispiel wenn es seine Aufgabe ist, den aktuellen emotionalen Zustand seines Gegenübers zu erkennen und darauf aufbauend fundierte Schlüsse zu ziehen. Die zugrunde liegende Idee ist, dass die Kombination aus natürlichsprachlicher Kommunikation und einer computeranimierten Figur eine Interaktion ermöglicht, die idealerweise intuitiver ist als klassische WIMP-Interfaces (Windows Icons Menus Pointer). Es entstehen ähnliche emotionale und kognitive Reaktionen wie in Gegenwart eines Menschen [2]. Aus diesem Potenzial sowie den vielfältigen Szenarien ergeben sich für Entwickler (Softwareentwickler, Computergrafiker) und Designer (Interface-Designer, Usability-Spezialisten, Medienwissenschaftler, Psychologen, Pädagogen) Anforderungen, die bisher kaum unterstützt und vereinheitlicht werden. Es mangelt an Möglichkeiten, die Interaktion selbst zu modellieren. Bei strukturlosem Entwerfen der Anwendungen besteht die Gefahr, dass die einzelnen Zustände unzureichend aufeinander abgestimmt werden. Obgleich zahlreiche Wege existieren, Zustände und Übergänge separat formal darzustellen, ist es hilfreich, diesen elementaren Aspekt als notwendiges Grundgerüst der Anwendung zu integrieren. Der Agent ist in vielen Szenarien Teil eines größeren Interfaces mit weiteren Benutzerschnittstellen-Elementen. In der Regel fehlt es jedoch an einer Architektur, um den Agenten innerhalb einer klassischen Benutzerschnittstelle darzustellen. Designer mit weniger Programmiererfahrung stehen hier vor der komplexen Aufgabe, zunächst die Schnittstelle der Agenten-Software zu bearbeiten, anstatt sich der eigentlichen Aufgabe widmen zu können. Wir präsentieren ein Multimodal Interaction Prototyping Tool (fortan MIPT), ein integriertes Werkzeug zur Lösung solcher Probleme. Sowohl in der Forschung als auch in der Praxis ist Mensch-Computer Interaktion seit jeher ein stark interdisziplinäres Feld. Wir erwarten daher zusätzlich, dass unser Ansatz die Kollaboration von Spezialisten mit verschiedenen Fähigkeiten vereinfachen könnte, weil Designer befähigt werden, stärker in den Bereich virtuelle Agenten vorzustoßen. Als Beispiel dient ein einfacher Prototyp für einen intelligenten Kaffeeautomaten.

2 Problemstellung

Prototyping ist ein wesentlicher Schritt bei der Gestaltung komplexer interaktiver Systeme. Je stärker der Fokus auf der Interaktion anstelle der Benutzeroberfläche liegt, desto größer ist der Mehrwert von lauffähigen anwendungsnahen Prototypen. Ist das Prototyping zu komplex, können sich Designer nicht selbstständig an dem Prozess beteiligen. Für gängige Benutzerschnittstellen gibt es daher Lösungen. Freie und mächtige Programme wie Qt [http://www.qt-project.org] bieten die Möglichkeit, lauffähige Entwürfe relativ einfach und schnell zu erstellen. Ein integrierter Benutzerschnittstellen-Designer unterstützt hier intuitives Entwerfen von Benutzeroberflächen. Das Verhalten einer Benutzeroberfläche zu explorieren ist jedoch anspruchsvoller. Ein Survey zeigte, dass 86% der befragten Designer eher

Schwierigkeiten mit dem Verhalten einer Oberfläche als mit der grafischen Umsetzung haben [3]. Derartige Probleme können sich im Umgang mit Agenten leicht vervielfachen, da die Technologie per se komplex ist und für die geübte Nutzung Programmierkenntnisse erfordert. In Kapitel 2.1 wird daher neben einer Beispielanwendung aus unserer Arbeitsgruppe der exemplarische Aufbau eines Systems kurz umrissen, damit die Funktionsweise unseres Tools verständlich wird (Kapitel 3). Die zentrale Problemstellung wird in Kapitel 2.2 zusammengefasst. Kapitel 4 beschreibt eine Nutzerstudie und weitere Erprobung von MIPT, Kapitel 5 diskutiert das Potenzial und bietet einen Ausblick.

2.1 Virtuelle Agenten

Virtuelle Agenten stellen eine besonders interaktive Form von Benutzeroberflächen dar. Sie ermöglichen eine multimodale Interaktion und können zudem mit künstlichen Formen von Motivation, Zielen, oder Wünschen ausgestattet sein, um Kooperation mit Menschen zu unterstützen [4]. Die Arbeit an virtuellen Agenten ist keine Aufgabe für einzelne Softwareentwickler; vielmehr vereinen die interdisziplinären Teams Wissen aus den Bereichen Informatik und Geisteswissenschaften, um den verschiedenen Perspektiven auf ein derartiges System gerecht zu werden. In der Arbeitsgruppe Kognitive Systeme und soziale Interaktion wird u. a. der virtuelle Agent Billie (Abb. 2.2) verwendet. In einem bereits evaluiertem Szenario dient Billie als Assistent, der ältere Menschen und solche mit kognitiven Einschränkungen dabei unterstützt, Termine zu verwalten [1]. Das System nutzt Sprachsynthese, Blickverhalten, Kopfbewegungen und Gesten zur intuitiven Kommunikation. Neben Billie wird ein Kalender eingeblendet. Die dort angezeigten Termine werden basierend auf dem Sprachinput des Nutzers aktualisiert. Zusätzlich verfolgt ein Eye-Tracker, ob Billie die Aufmerksamkeit des Nutzers hat und adressiert wird. In einer Studie mit der nicht-autonomen Variante des Systems konnte gezeigt werden, dass beide Nutzergruppen dazu bereit und fähig waren, mit dem Agenten zu kommunizieren [1]. Obwohl ältere Nutzer zunächst zögerlich eingestellt waren, konnte für diese Gruppe interessanterweise ein aus der interpersonalen Interaktion bekanntes Phänomen beobachtet werden: das spontane Erzählen von Anekdoten aus dem eigenen Leben. Damit erfüllt Billie nicht nur erfolgreich einen explizit assistierenden Zweck für Menschen mit Gedächtnisproblemen, sondern hat theoretisch das Potenzial eines sozialen Begleiters für eine Zielgruppe, die besonders auf soziale Kontakte angewiesen ist. Dieser Aspekt muss in Langzeitstudien bestätigt werden.

```
<bml xmlns="http://www.bml-initiative.org/bml/bml-1.0" character="Billie" id="bml1">
    <gaze id="gaze1" start="1" end="10" influence="NECK" target="camera"/>
</bml>
```

Abb. 2.1: Einfaches BML-Beispiel für das Blickverhalten des Agenten. Der Agent blickt für 9 Sekunden direkt in die Kamera und benutzt die Augen und den Hals.

Billies Verhalten ist flexibel. Die Manipulation erfolgt durch eine auf XML basierende Skriptsprache (BML: Behavior Markup Language; Abb. 2.1). BML ermöglicht Designern die Synchronisation von verbalem und nonverbalem kommunikativen Verhalten [5]. Umgesetzt

wird das Verhalten von einem Realizer [6]. Der Realizer plant und führt das in BML spezifizierte Verhalten zur Laufzeit aus. Die Manipulation des Agenten soll der Dynamik menschlicher Kommunikation Rechnung tragen und eine Interaktion mit natürlichen Abläufen ähneln, in der wechselseitige Dialoge entstehen, bis hin zu gegenseitiger abgefederter Unterbrechung. Neue BML-Erweiterungen ermöglichen zusätzlich das synchronisierte Abspielen von Audio- und Videodateien. Es obliegt dem Designer den Agenten so zu gestalten, dass sein Verhalten innerhalb des Kontexts stimmig ist und die gewünschten Effekte erzielt werden. Eine Vielzahl von Faktoren müssen berücksichtigt werden, zum Beispiel die äußere Erscheinung (Geschlecht, Alter, kultureller Hintergrund), Expressivität, Erwartungshaltung der Nutzer, Aufgabenstellung. Integrierte Design-Systeme für virtuelle Agenten sind mittlerweile ein Fortschritt bei dem Unterfangen, den Zugang zu dieser Technologie zu vereinfachen, doch sie umfassen nur einen (oder mehrere) Agenten, keine weiteren Interface-Elemente [7].

Abb. 2.2: Der virtuelle Agent Billie der AG Kognitive Systeme und soziale Interaktion

2.2 Resultierende Anforderungen

Die Manipulation des Agenten ist augenscheinlich nicht gleichzusetzen mit der erweiterten Kontrolle über die gesamte Benutzeroberfläche, es gibt also ein Steuerungsproblem. Hinzu kommt, dass in der Regel keine fest definierte Schnittstelle für externe Elemente der Benutzerschnittstelle existiert (Schaltflächen, Textfelder, etc.), d. h. es gibt ebenfalls ein Gestaltungsproblem. Da eine solche Schnittstelle jedoch benötigt wird, sobald eine funktionale Umgebung gestaltet werden soll, folgt daraus, dass nur mit dem System vertraute Entwickler in der Lage sind, Prototypen zu generieren. Innerhalb von Arbeitsgruppen, die Anwendungen für virtuelle Agenten entwickeln, kann demnach niemals eine echte Rollenverteilung stattfinden. Ein weiteres Problem ist das Fehlen von Design-Werkzeugen, die schnelles Generieren funktionsfähiger Benutzerschnittstellen unterstützen. Selbst wenn das Gestaltungsproblem gelöst werden kann, bleiben zentrale Fragen offen und können erneut nur von Entwicklern beantwortet werden: Wie ist der Eintritt in verschiedene Interaktionszustände definiert? Wie sind Ereignisse mit dem Agenten gekoppelt? Wie kann aus der laufenden Arbeit heraus ein lauffähiger Prototyp generiert werden? Wie viel Vorwissen benötigen Designer?3 Prototyping mit MIPT
MIPT besteht aus drei Hauptkomponenten:

1. der Realizer (für den Agenten Billie)
2. ein grafischer Editor für Zustandsgraphen (für das Interaktionsdesign, umgesetzt mit StateChart XML)
3. QtDesigner (für das Design der Benutzerschnittstelle).

Damit beinhaltet ein Prototyp zwei verschiedene Dateitypen: eine Zustandsgraph-Datei sowie eine oder mehrere .ui-Dateien aus Qt. Für einen lauffähigen Prototypen muss der Realizer im Hintergrund gestartet und die Dateien korrekt miteinander gekoppelt werden.

3.1 Interaktionsdesign mit StateChart XML (SCXML)

Die Interaktion mit der Benutzerschnittstelle wird mit einem Zustandsgraphen modelliert [8]. Zustandsgraphen bieten nützliche Erweiterungen im Vergleich zu Zustandsmaschinen, zum Beispiel Modellierung von Hierarchien, Nebenläufigkeit, und sind dadurch kompakter und ausdrucksstärker. Für das Design der Zustandsgraphen nutzen wir eine standardisierte, XML-basierte Sprache: W3C SCXML [http://www.w3.org/TR/scxml]. Verschiedene Werkzeuge unterstützen das (grafische) Design und die Ausführung von SCXML. Wir nutzen scxmlgui [https://code.google.com/p/scxmlgui/] und Apache SCXML [http://commons.apache.org/proper/commons-scxml/]. In unseren Zustandsgraphen können *Ereignisse* (lösen Zustandsübergänge aus) an Nutzer-Aktionen oder Sensor-Input gekoppelt werden, zum Beispiel das Betätigen einer Schaltfläche, die Erfassung des Nutzers in der Umgebung via Kamera, oder eine Swipe-Geste auf einem Touchscreen. Zustandsgraph *Aktionen* werden an den Entry oder Exit eines neuen Zustands gekoppelt, oder direkt an ein *Ereignis*. *Aktionen* senden BML-Kommandos an den Realizer, damit Billie zum Beispiel spricht oder eine Geste zeigt, und um Teile der Benutzerschnittstelle zu ändern (Abb. 3.1). Zusätzlich zum reaktiven

Verhalten, das basierend auf *Ereignisse* und *Aktionen* modelliert wird, kann das SCXML Datenmodell als Speicherstruktur genutzt werden, um Verhalten mit mehr Intention zu erzeugen.
Sensoren und Benutzerschnittstellen-Elemente (z. B. Textfelder) können ihren Inhalt in den Speicher schreiben, der aktualisiert wird. Bedingte Transitionen können erfolgen, indem der Speicherinhalt als *Wächter* für ein *Ereignis* funktioniert. So würde eine Transition zum Zustand ‘TooMuchCoffee’ im Prototyp der Kaffeemaschinen (Abb. 3.2) nur dann stattfinden, wenn die vom Nutzer konsumierte Kaffeemenge größer ist als 9 Einheiten. Die bisher georderten Einheiten wurden in früheren Interaktionen im Speicher abgelegt. Wir können ebenfalls mit BML auf den Speicher zugreifen. Wenn der Nutzer zum Beispiel seinen Namen in ein Textfeld eingibt, können wir den Eintrag im Speicher auslesen und Billie begrüßt den Nutzer mit „Hallo Herwin” (Abb. 3.2).

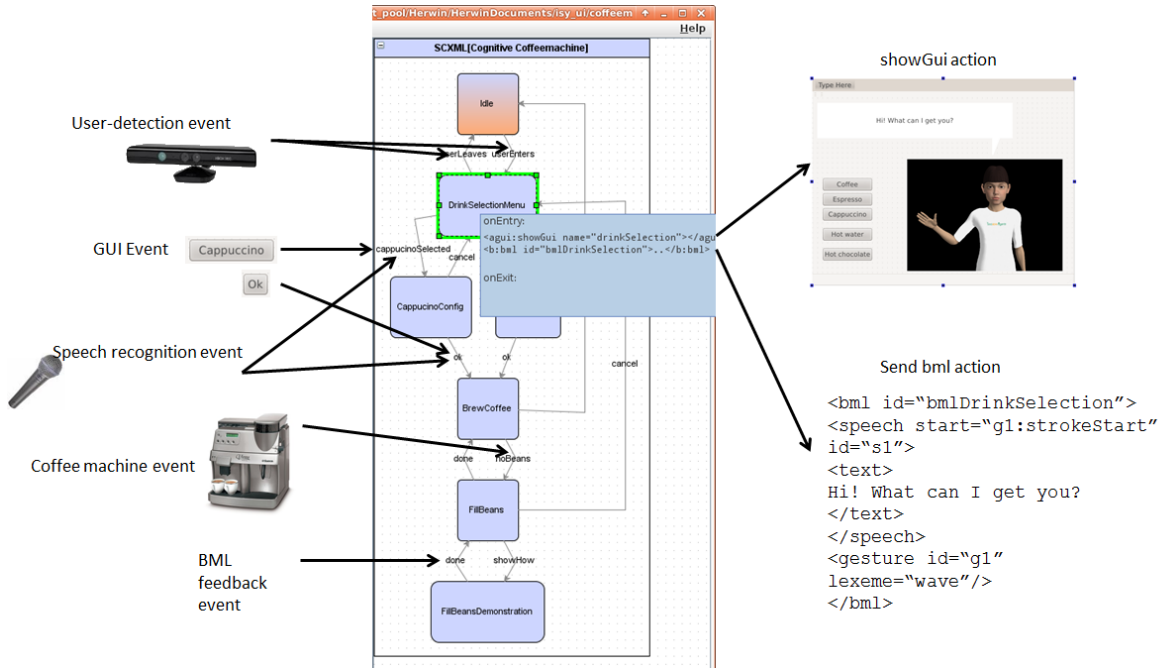

Abb. 3.1: Interaktionsentwurf. *Ereignisse* (links) sind an Transitionen im Zustandsgraphen gekoppelt (mitte). Die Darstellung und Bearbeitung wird unterstützend in einem grafischen Editor angeboten. Die Information im Zustandsgraphen senden als *Aktionen* BML an Billie, der als Widget in der Benutzerschnittstelle Platz findet.

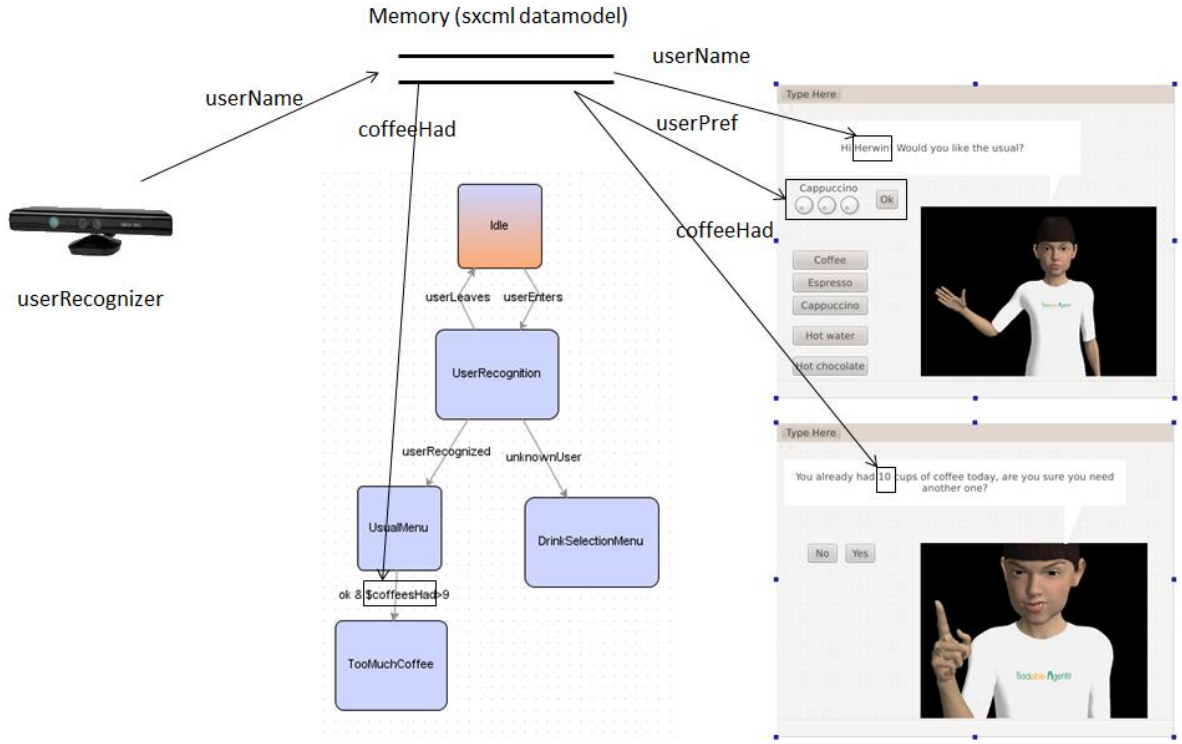

Abb. 3.2: Prototyp einer intelligenten Kaffeemaschine. Das Beispiel zeigt die Nutzung des Speicherinhalts in SCXML. Nachdem ein ‘UserDetection’-*Ereignis* auftritt, wird die User-ID mit dem Inhalt im Speicher abgeglichen. Das System weiß, dass der Nutzer schon zu viel Kaffee geordert hat. Billie reagiert entsprechend besorgt.

3.2 Kopplung mit der Benutzerschnittstelle

Es gibt viele exzellente Werkzeuge für das Design von Benutzerschnittstellen. Wir nutzen QtDesigner in unseren Designprozess. Designer können so ein bekanntes Programm für die Gestaltung der Oberfläche nutzen. Zusätzlich zu den typischen Qt-Widgets (Schaltflächen, Textfelder, Slider) bieten wir ein dediziertes Widget für Billie. Durch das Widget kann ein Fenster für Billie positioniert und in der Größe variiert werden. Das Widget ist jedoch nicht obligatorisch für ein Fenster, wodurch das Design für die Interaktion mit verschiedenen Fenstern ermöglicht wird, die Billie einbetten oder nicht einbetten. Wenn SCXML ausgeführt wird (siehe Kapitel 3.1), werden Fenster mit Hilfe von *Aktionen* verändert. Wird ein neues Fenster geladen, werden alle Elemente automatisch an den Zustandsgraphen gebunden (Abb. 3.3). Zum Beispiel können alle Buttons *Ereignisse* im Zustandsgraphen feuern und jede Eingabe in Textfelder wird automatisch in das Datenmodell des Zustandsgraphen übertragen. Die *Aktionen* und der Speicherinhalt werden anschließend mit Hilfe des Zustandsgraphen-Editors für das Interaktionsdesign genutzt.

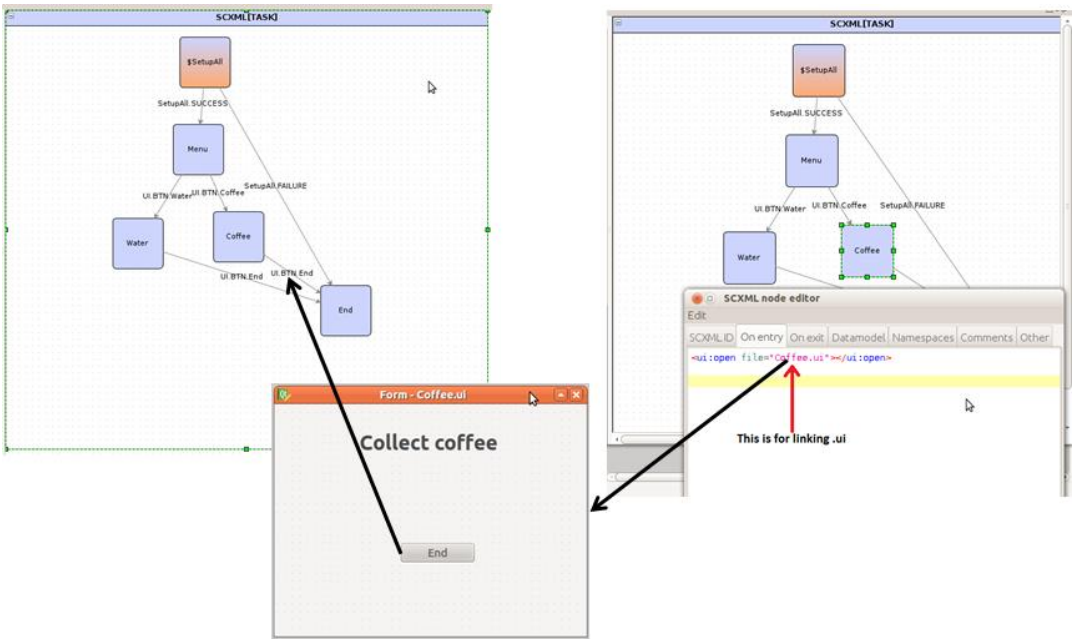

Abb. 3.3: Das „Collect coffee” Benutzerschnittstellen-Fenster wird mit einem ‘Bei Eintritt’-*Ereignis* im ‘Kaffee’-Zustand geöffnet. Wenn das Fesnter geöffnet wird, wird der Button automatisch an den Zustandsgraphen gebunden und der Buttonpress erzeugt innerhalb des Interaktionsdesigns eine Transition zum Endzustand.

4 Evaluation

In einer assistierten Nutzerstudie wurde analysiert, wie MIPT von Nutzern aufgenommen wird, die keine Erfahrung mit dem Agenten Billie und seiner Architektur haben. Um die Dauer der Prozedur zu verkürzen, wurde das Benutzerschnittstellen-Design in Qt ausgeklammert und ein einfaches Fenster vorgegeben. Somit lag der Fokus auf dem Interaktionsdesign, einem zentralen Bestandteil von MIPT. Da der Realizer von Billie in MIPT im Hintergrund läuft, nutzten die Teilnehmer hauptsächlich den SCXML-Editor und eine Dokumentation für MIPT. Acht Teilnehmer (Alter zwischen 22 und 32 Jahren) nahmen an der Studie teil. Sie gaben an, keine oder geringe Erfahrungen mit SCXML, Qt und virtuellen Agenten zu haben.

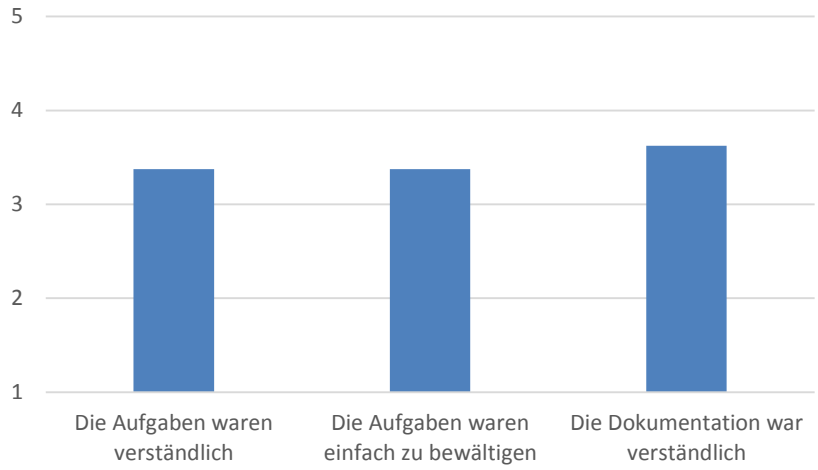

Abb. 4: Bewertungen der Aufgaben und Dokumentation in der Nutzerstudie (1: stimme überhaupt nicht zu, 5: stimme voll zu).

Die Teilnehmer wurden gebeten, ein Prototyping-Tool zu testen und anschließend zu bewerten. Ihnen wurde ein einfacher und unvollständiger Prototyp bestehend aus einer Benutzerschnittstelle und einem Zustandsgraphen vorgegeben. Die Gesamtaufgabe war, den Prototypen unter Zuhilfenahme einer Dokumentation (Wiki-basiert, Text und Screenshots) zu vervollständigen. Die hierfür nötigen Schritte wurden in vier Teilaufgaben gegliedert: 1. Starten von MIPT mit dem Zustandsgraphen und Nachvollziehen der Struktur. 2. Verbindung zur Benutzerschnittstelle innerhalb des Zustandsgraphen herstellen. 3. Transition zwischen zwei Zuständen einbauen. 4. BML-Kommando für eine verbale Nachricht an Billie senden. Zwei Versuchsleiter waren anwesend; einer von ihnen kontrollierte die vorgenommenen Änderungen, führte durch die Prozedur und stellte sicher, dass die Teilnehmer ausreichend Verständnis entwickeln konnten. Der zweite Versuchsleiter beobachtete und machte Notizen. Nachdem die Bearbeitung abgeschlossen war, füllten die Teilnehmer einen kurzen Evaluationsbogen aus. Darin notierten sie ihre Zustimmung zu drei Aussagen: „Die Aufgaben waren verständlich”, „Die Aufgaben waren einfach”, „Die Dokumentation war verständlich” (5-Punkt Likert-Skalen). Zusätzlich wurde die benötigte Zeit gemessen. Eine Teilaufgabe galt als nicht erfüllt, wenn sie von einem Versuchsleiter zu Ende gebracht werden musste.

Die durchschnittliche Bearbeitungszeit für die Gesamtaufgabe betrug 28:28 Minuten. Alle Teilnehmer konnten den Prototypen ohne signifikante Probleme fertigstellen, jedoch nicht ohne Hilfestellung, zum Beispiel Hinweise auf bestimmte bereits gelesene Teile der Dokumentation. Die Aufgaben und die Dokumentation wurden insgesamt tendenziell positiv bewertet (Abb. 4). Auffällig war die tragende Rolle der Benutzerfreundlichkeit des Zustandsgraph-Editors. Traten kleine Probleme auf bei Auswahl und Öffnen der Zustände und Transitionen, musste überprüft werden, ob die Informationen korrekt eingetragen wurden, d. h. Verbesserungen beim Drag & Drop, der Informationsdarstellung und Fehlerhandhabung sind nötig. Abschließend ist erwähnenswert, dass die Teilnehmer die Manipulation des Agenten Billie besonders interessant bewerteten, was gelegentliche und spontane Rückmeldung zeigte. 5 Zusammenfassung und Ausblick

Wir konnten zeigen, dass Nutzer grundsätzlich in der Lage sind, Arbeitsschritte in MIPT nachzuvollziehen und durchzuführen. Trotz positiver Rückmeldung von den Studienteilnehmern stehen wir noch am Anfang unserer Bemühung, ein integriertes Prototyping-Tool für Designer bereitzustellen, für die virtuelle Agenten normalerweise eine Technologie mit zu hohen Einstiegshürden sind. Detailliertere Untersuchungen zu den einzelnen Schritten des Prozesses, insbesondere dem Interaktionsdesign sind nötig, um das Tool an die Anforderungen der Nutzer anzupassen. Da MIPT eine Reihe von Prozessen vereint, bei denen Designer ihrer Aufgabe entsprechend beliebig weit in die Tiefe gehen können, wird es eine Herausforderung sein, die logische Verbindung zwischen diesen Prozessen konzeptionell ersichtlich zu machen und technisch zu vereinfachen. Laufende Arbeitsschritte an MIPT konzentrieren sich auf die Erweiterung des Inputs. Zunächst sollen Designer die Möglichkeit haben, einfache Touch-Anwendungen bauen zu können. Später soll eine zusätzliche Erweiterung für Gesten und einfache sprachliche Kommandos („Ja“, „Nein“, „in Ordnung“, etc.) stattfinden. Des Weiteren wollen wir die Evaluation von Prototypen unterstützen, indem die Interaktion aufgezeichnet und mit Markern (automatisch, basierend auf Ereignissen, Transitionen, aber auch arbiträr, basierend auf misslungener Interaktion mit Billie) versehen werden kann.

Wo liegen die natürlichen Grenzen unserer auf Zustandsgraphen basierten Lösung? Zustandsgraphen mit einem Datenmodell können alles darstellen, was eine Programmiersprache ausdrücken kann (Turing-Vollständigkeit). In der Praxis jedoch können womöglich nicht alle Interaktionsprozesse plausibel mit Zustandsgraphen modelliert werden. Es wäre zum Beispiel umständlich, probabilistische Modelle von modernen Dialogsystemen in diese Form zu übertragen. Wir evaluieren derzeit verschiedene Szenarien um diese Frage zu beantworten. Eine weitere Lösung könnte sein, mehrere Ansätze zu kombinieren, so dass ein bestimmter Teil bzw. Level des Dialogs mit einer Zustandsmaschinen und ein anderer Teil mit einem konventionellen Dialogsystem umgesetzt wird. Zustände für Begrüßung, Dialog, weitere Interaktion und Verabschiedung wären auf einem hohen Level realisiert, wobei ein Dialogsystem die Kommunikation während des Dialog-Zustandes bedient.

Danksagung
Dieses Forschungs- und Entwicklungsprojekt wird mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) innerhalb des Spitzenclusters „it’s OWL“ gefördert und vom Projektträger Karlsruhe (PTKA) betreut, sowie von der DFG innerhalb des Exzellenzclusters Cognitive Interaction Technology (CITEC). Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Literatur

[1] Yaghoubzadeh, R.; Kramer, M.; Pitsch, K.; Kopp, S.: Virtual agents as daily assistants for elderly or cognitively impaired people. Intelligent Virtual Agents; Springer: Berlin Heidelberg (2013), S. 79-91.

[2] Krämer, N. C.: Soziale Wirkungen virtueller Helfer. Kohlhammer: Stuttgart (2008).

[3] Myers, B.; Park, S. Y.; Nakano, Y.; Mueller, G.; & Ko, A.: How designers design and program interactive behaviors. Symposium on Visual Languages and Human-Centric Computing (2008), S. 177–184.

[4] Wachsmuth, I.; Mensch-Maschine-Interaktion. Handbuch Kognitionswissenschaft, Hrsg.: Stephan, A.; Walter, S.; J. B. Metzler: Stuttgart, Weimar (2013), S. 361-364.

[5] Kopp, S.; Krenn, B.; Marsella, S.; Marshall, A. N.; Pelachaud, C.; Pirker, H.; et al.: Towards a Common Framework for Multimodal Generation: The Behavior Markup Language. Proceedings of the 6th International Conference on Intelligent Virtual Agents, Hrsg.: Gratch, J. et al.; Springer: Berlin, Heidelberg (2006), S. 205-217.

[6] van Welbergen, H.; Reidsma, D.; Kopp, S.: An incremental multimodal realizer for behavior co-articulation and coordination. Proceedings of the 12th International Conference on Intelligent Virtual Agents, Hrsg.: Nakano, Y. et al.; Springer: Berlin, Heidelberg (2012), S. 175-188.

[7] Hartholt, A., Traum, D., Marsella, S. C., Shapiro, A., Stratou, G., Leuski, A., et al. All together now: Introducing the Virtual Human Toolkit. In Intelligent Virtual Agents (2013) S. 368–381

[8] Harel, D.: Statecharts: A visual formalism for complex systems. Science of Computer Programming, 8, (1987), S. 231-274.

[9] Kronlid, F.; Lager, T.: Implementing the information-state update approach to dialogue management in a slightly extended SCXML. Workshop on the Semantics and Pragmatics of Dialogue (2007).