technische universität
dortmund

Arbeit zur Erlangung des akademischen Grades
Bachelor of Science

Graphisch-interaktive Entwicklungsumgebung für Robotersysteme zum Einsatz in den Rehabilitationswissenschaften

Benjamin Matthias Biehler
geboren in Nürnberg

2021

Lehrstuhl für Computergraphik VII
Fakultät Informatik
Technische Universität Dortmund

Erstgutachter:	PD Dr. Weichert
Zweitgutachter:	Prof. Dr.-Ing. Bühler
Abgabedatum:	31. September 2021

Kurzfassung

Hier steht eine Kurzfassung der Arbeit in deutscher Sprache inklusive der Zusammenfassung der Ergebnisse. Zusammen mit der englischen Zusammenfassung passt sie auf eine Seite.

Abstract

The abstract is a short summary of the thesis in English, together with the German summary it has to fit on this page.

Inhaltsverzeichnis

1 Einleitung

Kurze Einleitung in die Thematik der Bachelorarbeit (Thema vorstellen, Leser*in neugierig machen, Relevanz der BA in Kontext setzen). Das ganze Kapitel sollte auf etwa 3 Seiten machbar sein.

Was macht diese Arbeit interessant?

1.1 Motivation

Wieso ist die entstandene Entwicklungsumgebung entstanden? Was war der Anstoß dazu den Ansatz zu verfolgen? Wer ist die Zielgruppe für diese Anwendung?

1.2 Struktur der Arbeit

Wie ist diese Arbeit aufgebaut? Was ist in den einzelnen Kapiteln zu erwarten? (Wie war die Arbeit aufgebaut?)

2 Mensch-Roboter-Interaktion

2.1 Herausforderungen

Was sind die Probleme, die mit der Anwendung gelöst werden sollen? Was waren die Hindernisse, die sich für diese Arbeit gestellt haben?

2.2 Stand der Technik

Werden ähnliche Tools bereits eingesetzt, wenn ja mit welchem Erfolg? Was war der bisherige Ansatz in dieser Domäne (Pepper-Programmierung)? Wie funktioniert Choregraphe?

2.3 Kollaborations-Schnittstelle

Was ergibt sich aus 2.1 und 2.2 für die Arbeit? Welche grobe Zielsetzung wurde festgelegt?

3 Interaktive Visuelle Robotersteuerung

1-2 Absätze die dieses Kapitel einleiten (wahrscheinlich längstes Kapitel der Arbeit).

Beschreibung des GUIs und der Prinzipien dahinter. Unterkapitel gerne nochmal durchmischen.

3.1 Endbenutzer-Entwicklung

Damit die Robotersteuerung durch Endbenutzer ermöglicht werden kann muss die Entwicklungsumgebung Endbenutzer-Entwicklung umsetzen. Dies ist ein aktueller Forschungszweig, der sich mit Methoden, Techniken und Werkzeugen beschäftigt, die Endbenutzern anstelle von professionellen Software-Entwicklern die Anpassung, Erweiterung oder Entwicklung von Software ermöglichen [4].

Da es, wie in Kapitel 2.3 beschrieben, das Ziel der Entwicklungsumgebung war, Endbenutzern die Möglichkeit zu geben eigenständig ebendiese Aufgaben zu erfüllen ist es wichtig die verbreiteten Ansätze dieses Forschungsfeldes zu betrachten.

Endbenutzer-Entwicklung beschäftigt sich mit einem breiten Feld von verschiedenen Anwendungsfällen. Diese reichen von Tabellenkalkulation, über mobile Anwendungen und Websiten, bis hin zu der Programmierung von industriellen Robotern. Diese verschiedenen Domänen ziehen ebenso verschiedene Lösungsansätze mit sich. Im folgenden werden mehrere dieser Ansätze kurz eingeführt, mit denen Roboterprogrammierung ermöglicht werden könnte. Daraufhin wird erklärt welcher dieser Ansätze genutzt wurde um die Entwicklungsumgebung umzusetzen, und wieso dieser den anderen vorgezogen wurde.

3.1.1 Arten von Endbenutzer-Entwicklung

Low-Code Plattformen verringern, meistens durch graphische Benutzerschnittstellen, den Bedarf an von Hand geschriebenen Code. Je nach Plattform müssen jedoch einige Teile von Anwendungen noch selbst geschrieben werden. Dies vereinfacht und beschleunigt die Entwicklung für professionelle Programmierer und senkt die

Barriere für Endbenutzer . Im Falle von No-Code Plattformen entfällt der Bedarf Code zu schreiben komplett und macht es Endbenutzern möglich diese Plattformen eigenständig zu benutzen. Diese Plattformen nutzen meist einen graphischen Ansatz, um die Programmierung ohne Code zu ermöglichen, und sind mit dem Begriff der Visuellen Programmierung verwandt.

Natural Language Programming ist der Ansatz, dass Endbenutzer Programme in normalen deutschen oder englischen Sätzen schreiben. Die Sätze und Programm-Dateien folgen dabei trotzdem noch einer klar definierten Struktur, um die Interpretation durch den Computer zu ermöglichen. Aus den so geschriebenen Dateien generiert die Programmierumgebung dann für die Maschine lesbaren Code [5].

Programming By Example - bei der Anwendung mit Robotern auch Programming By Demonstration genannt - versucht es zu ermöglichen, dass der Endbenutzer dem Roboter oder Computer vorführt was dieser tun soll. Dies kann entweder durch Datenpaare von Ein- und Ausgabe [10] oder durch Sensoren geschehen, die beispielsweise Robotern gewünschte Bewegung vorführen. Daraus kann sich dann ein Anlern-Prozess entwickeln, in dem die Maschine selbst Beispiele ausführt, deren Fehler der Endbenutzer dann korrigieren kann [2].

Evtl. weitere Konzepte kurz umreißen

3.1.2 Auswahl des Enbenutzer-Entwicklungs Konzepts

Um die Entwicklungsumgebung zu erstellen wurde die Entscheidung getroffen eine No-Code Plattform zu nutzen. Einerseits sind sowohl Programming By Demonstration, als auch Natural Language Programming weitaus komplexer zu realisieren, als visuelle No-Code Plattformen. Weiterhin sind Natural Language Programming Ansätze eher auf textbasierte Anwendungen, wie maschinelle Übersetzungen und Informationsauswertung, optimiert [3]. Dagegen sind visuelle No-Code Plattformen auf prozedurale Programmierung spezialisiert und mit Hilfe von einfachen Bibliotheken leicht zu realisieren. Auch sind visuell dargestellte Programme leichter zu verstehen als Textbasierte und bieten mehr Überblick über den Ablauf des Programmes.

Weiter ausführen. Studien lesen.

3.2 Visuelle Programmierung

Visuelle Programmierung ersetzt die textuellen Elemente traditioneller Programmiersprachen ausschließlich durch graphische Bausteine [8]. Historisch gesehen werden sie besonders in pädagogischen Umfeld [7], Benutzeroberflächen [6] und Robotik [11]

und sind meist dazu ausgelegt von nicht professionellen Endbenutzern bedienbar zu sein.

Dementsprechend eignet sich ein visueller Programmieransatz für die Realisierung der Entwicklungsumgebung für Pepper.

23.8.

3.2.1 Visuelle Programmierung in der Robotik

3.2.2 Eigenschaften von Visueller Programmierung

3.2.3 Block-Programmierung

3.3 Iterativer Entwicklungszyklus

4 Roboter-Ansteuerung

4.1 Sprachausgabe

4.2 Bewegungssteuerung

4.3 Spracherkennung

5 Benutzerzentrierte Realisierung

Damit die Realisierung der graphisch-interaktiven Entwicklungsumgebung auf die Nutzenden angepasst werden kann, wurden die Vorlieben der Nutzenden mit verschiedenen Methoden abgefragt. Die daraus entstandenen Ergebnisse wurden bei der Entwicklung berücksichtigt. Das folgende Kapitel führt die beiden Konzepte ein und erklärt, wie diese angewendet wurden.

In Abschnitt 5.1 wird das Konzept des Fragebogens erläutert, mit dem die Anforderungen der Nutzenden erhoben wurden. Daraufhin wird in Abschnitt 5.2 die Struktur der teilnehmenden Beobachtung aufgeführt, mit der das Feedback der Nutzenden gesammelt wurde. Zuletzt wird in Abschnitt 5.3 die Anwendung dieser beiden Konzepte und die Rekrutierung der Nutzenden dargelegt.

5.1 Gestaltung der Anforderungserhebung

Um die Entwicklungsumgebung auf die Bedürfnisse der Nutzenden anzupassen, müssen deren Anforderungen abgefragt werden. Zu diesem Zweck bietet sich besonders durch die Möglichkeit der Einbindung von zusätzlichen Medien eine gezielte Online-Befragung an [9]. Aus den Ergebnissen dieser Befragung entstand dann ein anfänglicher Style-Guide für die Benutzeroberfläche.

Damit die Befragung nützliche Ergebnisse liefert, müssen alle Fragen verständlich, knapp und neutral formuliert sein [1]. Der Fragebogen wurde in einzelne Abschnitte unterteilt, die alle nötigen Informationen enthielten und ohne Scrollen angezeigt werden konnten. Um technischen Problemen, die durch verschiedene Endgeräte und Browser entstehen könnten [9], vorzubeugen, wurde die Online-Umfrage-Applikation LimeSurvey verwendet. Diese bot alle nötigen Funktionen wie die Möglichkeit, Teilnehmende an der Umfrage zu verwalten und die Daten in der Anwendung auszuwerten.

Bevor den Teilnehmenden an der Umfrage Fragen gestellt wurden, wurde Pepper, siehe Kapitel 4, und das Anwendungsbeispiel in Abbildung 5.1 vorgestellt und erklärt. Das Anwendungsbeispiel, das die Umfrage begleitet, besteht nur daraus, dass Pepper sich vorstellt und dann das Gegenüber fragt, ob Pepper sich bewegen

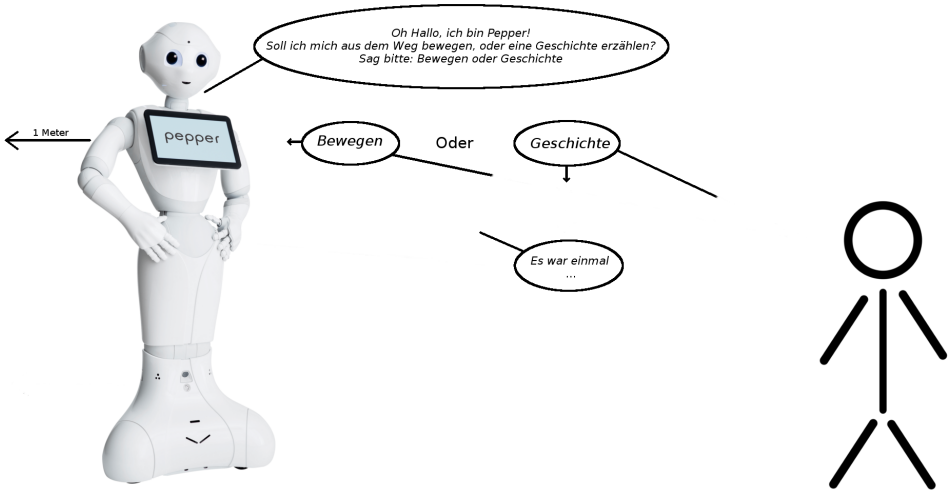


Abbildung 5.1: Anwendungsbeispiel der Anforderungserhebung.

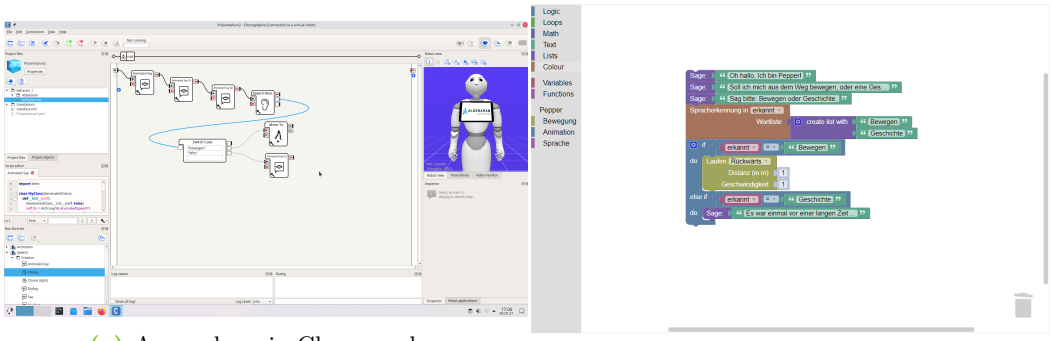
oder eine Geschichte erzählen soll. Sagt die Person ”Bewegen” oder ”Geschichte”, führt Pepper die gewünschte Aktion aus.

In den folgenden Abschnitten werden die einzelnen Teile des Fragebogens und deren Nutzen für den Entwicklungsprozess aufgeführt. Die Auswertung der Ergebnisse des Fragebogens folgt in Kapitel 6.2.

5.1.1 Vergleich mit bestehenden Anwendungen

Der erste Anschnitt der Umfrage stellt Choregraphe, siehe Abschnitt 2.2, und das Konzept der Entwicklungsumgebung mithilfe eines Mock-ups vor. Dabei werden die grundlegenden Funktionen beider Programme erläutert. Dazu wird jeweils die Realisierung des Anwendungsbeispiels aus Abbildung 5.1 in beiden Entwicklungsumgebungen, siehe Abbildung 5.2, dargestellt.

Die Teilnehmenden an der Umfrage werden daraufhin aufgefordert, neun Aussagen auf je einer Skala für beide Entwicklungsumgebungen zu bewerten. Zu diesem Zweck wurde eine 5-Punkte Likert-Skala [9] verwendet, die eine neutrale Antwortmöglichkeit zulässt. Diese sollte existieren, damit die Teilnehmenden nicht gezwungen sind, eine Entscheidung zu treffen [12], falls sie wegen ihrer fehlenden Erfahrung unsicher sind. Die beiden Extrema der Skala sind mit ”1: Stimme überhaupt nicht zu” und ”5:



(a) Anwendung in Choregraphe.

(b) Anwendung im Blockly Mock-up.

Abbildung 5.2: Das Anwendungsbeispiel in Choregraphe und dem Blockly Mock-up umgesetzt.

Stimme voll zu" beschriftet. Diese Skala wird im weiteren Verlauf des Fragebogens weiterhin verwendet, um Konsistenz zu schaffen.

Die Aussagen beziehen sich insbesondere auf Verständlichkeit der Benutzeroberfläche, Einschätzung der eigenen Fähigkeit, die Entwicklungsumgebung zu nutzten und Interesse an der Nutzung der Anwendungen.

Die hier erhobenen Daten wurden dazu genutzt, eine erste Einschätzung des Mock-ups zu erlangen. Außerdem gaben sie eine Indikation, ob ein direkter Vergleich der beiden Entwicklungsumgebungen in den Nutzertests nötig ist.

5.1.2 Blockgestaltung

Weiter, Übergänge dazwischen finden

Wie bereits in Kapitel 3.2 dargelegt, gibt es verschiedene Möglichkeiten Blöcke in Blockly zu gestalten. Um diese Designentscheidungen den Nutzenden anzupassen, werden im zweiten Abschnitt der Online-Umfrage den Teilnehmenden jeweils verschiedene Mock-ups von funktional gleichen Blöcken präsentiert. Sie sollen dann den für sie am einfachsten verständlichen Block auswählen.

Die Auswahl der verschiedenen Optionen wurde so gewählt, dass die Mock-ups, die zur Auswahl stehen, jeweils Designentscheidungen repräsentieren. Beispielsweise steht die Entscheidung in Abbildung **TODO** repräsentativ für die Option einzelne Blöcke mit Hilfe von Drop-Down-Menüs zusammenzufassen und die Entscheidung Parameter-Slots generell extern (rechts) oder intern (mitten im Block) bereitzustellen.

Die gesammelten Antworten werden in den Style-Guide übersetzt, der die Vorlieben der Teilnehmenden an der Umfrage repräsentiert.

5.1.3 Funktionale Anforderungen

Im darauf folgenden Abschnitt des Fragebogens werden konkrete Anforderungen abgefragt. Den Teilnehmenden werden verschiedene Aussagen präsentiert, die sie auf einer 5-Punkte Likert-Skala bewerten sollen. Da die Teilnehmenden bereits eine Einführung in die Entwicklungsumgebung erhalten haben und sie im vorangegangenen Abschnitt des Fragebogens verschiedene Mock-ups und Kontrollelemente kennengelernt haben, haben sie nun ein Verständnis der Benutzeroberfläche um diese Anforderungen zu bewerten.

Die abgefragten Anforderungen bezogen sich auf Barrierefreiheit, Sprache der Benutzeroberfläche, integrierte Funktionen in der Oberfläche, Benutzungsweise der Entwicklungsumgebung und weitere Funktionen dieser.

Teile dieser Anforderungen, wie die Sprache und Benutzungsweise der Benutzeroberfläche, flossen ebenso in den Style-Guide, siehe B, ein. Weitere Teile dieser Ergebnisse, wie etwa die Aussagen zu Barrierefreiheit, Erklärungen und Funktionen in der Benutzeroberfläche, bildeten eine Prioritäten-Liste von Funktionen, die in diese Anwendung eingefügt werden können.

5.1.4 Sonstige Datenerhebung

Im abschließenden Teil der Umfrage wurden noch demographische Informationen, wie Studiengang und Semester, erfasst. Ebenso wurde die Programmier-Erfahrung der Teilnehmenden einfachen Ja-Nein-Fragen abgefragt.

Um Teilnehmende für die Nutzertests an den entstehenden Prototypen zu rekrutieren, wurden diese auf der letzten Seite der Online-Befragung dazu Aufgefordert eine E-Mail Adresse anzugeben. Da diese dann auch mit den Daten der vorangegangenen Fragen verknüpft sind, kann auch die Teilmenge der Nutzenden, die an den Nutzertests teilgenommen haben, in Kapitel 6.2 einzeln betrachtet werden.

5.2 Instanziierung der Testsettings

Wie wurden die entstandenen Prototypen getestet? Welche Aufgaben sollten die Nutzer durchführen? Welche Involvierung hatte der Beobachter der Tests?

5.3 Operationalisierung

Welche Nutzer?

6 Evaluierung

Wie wurden die Ergebnisse dieser Arbeit ausgewertet?

6.1 Anforderungsanalyse

Konzepte zum Auswerten

6.2 Quantitative Auswertung

Welche Ergebnisse haben die Anforderungsanalyse und Nutzertests ergeben? Wie kann man diese interpretieren?

6.3 Qualitative Auswertung

6.4 Softwarequalität

Ist es deterministisch? Software-Bewertung

7 Zusammenfassung und Ausblick

Was hat die Auswertung ergeben? Was lässt sich aus den Ergebnissen der Arbeit schließen?

7.1 Zusammenfassung

7.2 Ausblick

War der Ansatz hilfreich? Welche Bedeutung haben die Ergebnisse für ähnliche Projekte? Welche weiteren Veränderungen sollten an diesem Projekt vorgenommen werden, um die Benutzbarkeit zu verbessern?

A Statistik

Statistische Auswertung der Nutzerbefragung

Pseudonymisierte statistische Auswertung der Nutzertests

B Style-Guide

Der Entstandene Style-Guide für die Benutzeroberfläche. Etwas schön aufgearbeitet

Literatur

[1] L. Jacobsen J. und Meyer. *Praxisbuch Usability und UX*. Rheinwerk Verlag, 2019. ISBN: 9783836269551.

[2] D. Kurlander, A. Cypher und D. Halbert. *Watch what I do: programming by demonstration*. 1993. URL: http://acypher.com/wwid/index.html.

[3] Elizabeth D Liddy. „Natural language processing“. In: (2001).

[4] H. Lieberman et al. „End-user development: An emerging paradigm“. In: *End user development*. Springer, 2006, S. 1–8.

[5] P. Nadkarni, L. Ohno-Machado und W. Chapman. „Natural language processing: an introduction“. In: *Journal of the American Medical Informatics Association* 18.5 (Sep. 2011), S. 544–551. ISSN: 1067-5027. DOI: 10.1136/amiajnl-2011-000464.

[6] J. Rode et al. *An end-user development perspective on state-of-the-art web development tools*. Techn. Ber. Department of Computer Science, Virginia Polytechnic Institute & State ..., 2005.

[7] J. Sáez-López, M. Román-González und E. Vázquez-Cano. „Visual programming languages integrated across the curriculum in elementary school: A two year case study using “Scratch” in five schools“. In: *Computers & Education* 97 (2016), S. 129–141. ISSN: 0360-1315. DOI: https://doi.org/10.1016/j.compedu.2016.03.003. URL: https://www.sciencedirect.com/science/article/pii/S0360131516300549.

[8] Stefan Schiffer. „Visuelle Programmierung - Potential und Grenzen“. In: *GI Jahrestagung*. 1996.

[9] R. Schnell, P. Hill und E. Esser. *Methoden der empirischen Sozialforschung*. De Gruyter Oldenbourg, 2018. ISBN: 9783110577327.

[10] S. Singh R. und Gulwani. „Predicting a Correct Program in Programming by Example“. In: *Computer Aided Verification*. Hrsg. von C. Kroening D. und Păsăreanu. Cham: Springer International Publishing, 2015, S. 398–414. ISBN: 978-3-319-21690-4.

[11] D. Weintrop et al. „Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices“. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2018, S. 1–12. ISBN: 9781450356206. URL: https://doi.org/10.1145/3173574.3173940.

[12] F. Willits, G. Theodori und A. Luloff. „Another look at Likert scales“. In: *Journal of Rural Social Sciences* 31.3 (2016), S. 6.

Abbildungsverzeichnis

Eidesstattliche Versicherung

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem Titel „Graphisch-interaktive Entwicklungsumgebung für Robotersysteme zum Einsatz in den Rehabilitationswissenschaften“ selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift

Belehrung

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50 000 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden (§ 63 Abs. 5 Hochschulgesetz –HG–).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z. B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen.

Ort, Datum	Unterschrift