Karlsruhe Institute of Technology

Konzeption und Entwicklung einer intuitiven Modellierungssprache für digitale Therapien mittels Chatbots

Masterarbeit
von

Luisa Christine Andre

Lehrstuhl für Pervasive Computing Systeme/TECO
Institut für Telematik
Fakultät für Informatik

Erstgutachter:	Prof. Dr. Michael Beigl
Zweitgutachter:	TODO: Eintragen
Betreuer:	PD Dr. Andrea Schankin

Projektzeitraum: 01.01.2019 – 30.06.2019

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, den TODO: date

Zusammenfassung

TODO: Zusammenfassung (Deutsch)

Abstract

TODO: Zusammenfassung (Englisch)

Contents

1. Einleitung

Sie geben Auskunft über das Wetter, nehmen Bestellungen entgegen oder wirken als Coach - Chatbots werden bereits vielseitig im Alltag eingesetzt. Auch die Psychologie profitiert von diesen Entwicklungen. 1966 entwickelte Joseph Weizenbaum mit *ELIZA* den ersten Chatbot. *ELIZA* sollte seinem menschlichen Gesprächspartner das Gefühl geben, dass dieser mit einem Psychiater über eine Chatoberfläche kommuniziert. Entwickelt wurde *ELIZA* allerdings nicht mit der Absicht Psychotherapie zugänglich zu machen, sondern um ein Modell zur maschinellen Verarbeitung von natürlicher Sprache zu implementieren. Was mit Joseph Weizenbaums *ELIZA* begann, brachte mit der Entwicklung der Forschung und Technik schließlich einige Chatbots, wie *Wysa*, *Woebot* und *Tess*, im Bereich der psychischen Gesundheit hervor. Sie stellen heutzutage verschiedene Methoden der Kognitiven Verhaltenstherapie bereit, die Nutzern helfen können deren Introspektion zu verbessern und die Methoden direkt anzuwenden. Dabei wirken sie wie ein Coach der jederzeit erreichbar ist.

In den 1960-ern hatten nur wenige Zugang zu Computern. Durch ihre Bauweise benötigten diese nicht nur viel Platz, sie waren zu dieser Zeit auch sehr kostspielig. Die Technik hat sich allerdings über die Jahrzehnte hinweg stark verändert. Nicht nur wurden Computer erschwinglich und haben eine deutlich größere Rechenleistung, sie begleiten uns mittlerweile auch in Form eines Tablets oder Laptops als Personal Computer durch den Alltag. Seit Apple ihr erstes Smartphone *Iphone* im Jahr 2007 einführte, eröffneten sich durch diese Geräte noch weitere technische Möglichkeiten. Smartphones entwickelten sich zu kleinen, handlicen Geräten die nahezu in jeder Tasche Platz finden. Außerdem beinhalten die Geräte heutzutage verschiedene Sensoren, haben Zugriff auf eine Vielzahl von Anwendungen und können sich mit dem Internet verbinden. Die Handlichkeit und Vielzahl an mitgebrachten Funktionen führte dazu, dass Smartphones im Jahre 2018 allein in Deutschland von bis zu 57 Millionen Personen genutzt wurden.

Chatbot-Entwickler nutzen die technischen Vorteile die Smartphones und Personal Computer mit sich bringen. So begleitet *Woebot* Menschen mit Depressionen oder inneren Unruhen mit Techniken aus der Kognitiven Verhaltenstherapie als Selbsthilfe durch den Alltag. Der Nutzer kann dabei auswählen, ob dieser über eine

Iphone-App, *Android*-App oder via *Facebook Messenger* mit *Woebot* kommunizieren möchte. Letzteres ist auf jedem browserfähigen Gerät nutzbar. Eine Studie der *Stanford School of Medicine* untersuchte den Einsatz von *Woebot* hinsichtlich seiner Realisierbarkeit, Nutzerakzeptanz und die vorläufige Wirksamkeit des bereitgestellten Selbsthilfeprogramms. Das Ergebnis der Studie zeigte, dass *Woebot* beinahe täglich von den Probanden genutzt wurde. Außerdem ließ sich bei diesen ein positiver Einfluss hinsichtlich ihrer Depressionsbewältigung und dem Umgang mit inneren Unruhen messen.

Diese Ergebnisse zeigen auf, dass Chatbots im Bereich der Psychologie und Psychotherapie nützliche Werkzeuge sein können. Allerdings ist das Entwickeln solcher Chatbots für Psychologen noch immer eine Hürde. Zwar gibt es zahlreiche Baukästen zur Entwicklung von Chatbots die keine tiefgreifenden Programmierkenntnisse benötigen. Diese sind jedoch überwiegend auf den Bereich des Marketings ausgerichtet, weshalb sie in ihrem Funktionsumfang meist eingeschränkt sind. Baukästen die mehr Funktionalität bieten, benötigen lange Einarbeitungszeit und Expertenwissen in Bezug auf ihre Programmierung. Eine einfache und schnelle Umsetzung ist daher oft nicht möglich. Auch die Entwicklung eines eigenen Produktes birgt für Psychologen und Softwareunternehmen Probleme. So scheitert die Umsetzung unter anderem an Kommunikationshürden zwischen Entwicklern und Psychologen oder den komplexen Anforderungen, die medizinische Produkte zu erfüllen haben.

Das Unternehmen *movisens GmbH* entwickelt derzeit das Projekt *TherapyBuilder* welches Psychologen sowie Psychotherapeuten die Möglichkeit bieten soll, Chatbots für Studien sowie zur Therapiebegleitung einzusetzen. Im Rahmen dieser Masterarbeit wird für das Projekt TherapyBuilder die Modelliersungssprache *TML* (Therapy Modelling Language) entwickelt. Ziel dieser *TML* ist es, Psychologen die autonomie zu geben, ohne Expertenwissen, Chatbots zu erstellen um diese in Studien und Therapiebegleitend einzusetzen.

1.1 Problemstellung und Zielsetzung

Ziel der Arbeit ist die Konzeption einer Therapiemodellierungs-sprache. Diese soll es erlauben, technisch wenig versierten Psychologen ihre Therapieideen in einer Art und Weise zu formulieren, die eine Maschine verstehen und ausführen kann. Dadurch entfällt der hohe und fehleranfällige Abstimmungsaufwand zwischen Forschern und Entwicklern. Durch den Einsatz der Therapiemodellierungs-sprache sollen MPG konforme Anwendungen mit einer conversational UI entstehen, welche eine für den Patienten vertraute, Gesprächsähnliche Kommunikation ermöglicht. Dies erlaubt es eine stärkere persönliche Bindung zwischen App und dem Patienten herzustellen, was den Therapieerfolg unterstützen soll.

In der Arbeit gilt es vor allem die komplexen Konfigurationsmöglichkeiten der Domäne einer digitalen Therapie funktional abzubilden. Durch eine Befragung der Anwendergruppen soll Domänenwissen erarbeitet werden und im Folgenden die Therapiemodellierungssprache und dessen grafische Repräsentation iterativ entworfen werden. Dabei gilt ein hohes Augenmerk der Usability, die gerade in der Therapie-Domäne einen hohen Stellenwert einnehmen muss.

Diese entworfene Modellierungssprache soll prototypisch umgesetzt werden und in einer kleinen Usability-Studie evaluiert werden.

1.2 Umfeld

1.3 Methodisches Vorgehen

1.4 Gliederung

- Grundlagen (Kapitel 2, Definitionen, Rahmenbedingungen, Chatbots)
- Stand der Forschung und Technik (Kapitel 3, Chatbot-Plattformen, Grafische Programmiersprachen, Auszeichnungssprachen, Experience Sampling Software, Domänenspezifische Sprachen)
- Konzeption (Kapitel 4, Anforderungsanalyse, Ausarbeitung verschiedener Konzepte)
- Prototypische Entwicklung der Modellierungssprache (Kapitel 5, Konzept, Umsetzung, Evaluation)
- Ergebnisse (Kapitel 6, Zusammenfassung, Kritische Reflexion)
- Ausblick

2. Grundlagen

2.1 Definitionen

2.2 Rahmenbedingungen von Psychotherapien

2.3 Chatbots

3. Stand der Forschung und Technik

Recherchen haben ergeben, dass derzeit noch keine Sprache existiert, die speziell zur Modellierung von Therapien mit Chatbots entwickelt wurden. Für die Bearbeitung der Forschungsfrage werden verschiedene Technologien bewertet. Zunächst werden Plattformen beleuchtet, die eine Erstellung von Chatbots ermöglichen, ohne weitere Programmierkenntnisse zu benötigen. Nachfolgend werden diverse grafische Programmiersprachen betrachtet. Diese bilden Programmstrukturen grafisch ab und werden somit nachvollziehbarer für den Anwender. Eine weitere Kategorie bilden die Auszeichnungssprachen. Diese ermöglichen eine vereinfachte Programmierung zur Strukturierung von Texten. Abschließend werden verschiedene Ansätze im Bereich des Experience Samplings behandelt. Zwar dienen diese zur Erstellung von Studien, allerdings beinhalten sie eine Schnittmenge an Funktionalitäten, die ebenfalls für das Designen von Therapien relevant sind.

TODO: Weiter ausführen

3.1 Chatbot-Plattformen

Es gibt eine Vielzahl verschiedener Plattformen zur Entwicklung von Chatbots. Diese sollen insbesondere Personengruppen adressieren, die keine oder nur wenige Programmierkenntnisse besitzen. Die entsprechenden Plattformen verwenden unterschiedliche Ansätze für die Entwicklung.

Der Konversationsfluss der Chatbots wird auf den Plattformen unter anderem als eine Art Baum, ähnlich zur bekannten Ordnerstruktur unter Windows Betriebssystemen, angelegt und dargestellt (vgl. [10] [12]). Chatbot-Plattformen, wie *ManyChat*, *Converse.ai* und *Chatfuel* verwenden Diagramme zur Darstellung eines Chatverlaufes (vgl. [8] [22]) oder Blocksysteme mit Referenzen auf nachfolgende Blöcke (vgl. [6]). Andere Plattformen nutzen keine Darstellung des Verlaufs (vgl. [5]). Im Beispiel von *Botsify* oder *Recast.ai* werden nur Verhaltensweisen angelegt, die durch bestimmte Nutzereingaben getriggert werden.

Auch in der Handhabung der Nutzereingaben gibt es verschiedene Ansätze. So bieten manche Plattformen die Möglichkeit Antworten für den Nutzer des Chatbots vorzugeben (vgl. [6] [22]). Andere hingegen verwenden natürliche Sprachverarbeitung um Phrasen einzutrainieren. Der Ersteller des Chatbots legt fest, wie der Chatbot auf die entsprechenden Phrasen reagiert. Somit kann ein Chatbot auf Synonyme oder Redewendungen eingehen. (vgl. [5] [10] [12]) Die Chatbot-Plattform *Chatfuel* verwendet beide Ansätze. So können hier Antworten vordefiniert oder Phrasen festgelegt werden. (vgl. [6]).

Damit Nutzerdaten abgespeichert und verarbeitet werden können, bieten einige Plattformen Variablen an. Dort können unter anderem Nutzername sowie Aufenthaltsort des Nutzers gespeichert und weiterverwendet werden. Der Nutzer kann auf bereits definierte Variablen zurückgreifen oder eigene anlegen. (vgl. [6] [8] [10] [12] [22]).

TODO: Weiter ausführen

3.2 Grafische Programmiersprachen

Diese Art der Programmiersprache bedient sich visueller Elemente um Programmstrukturen verständlich abzubilden. Die visuellen Elemente können auf eine bestimmte Domäne zugeschnitten sein (vgl. [21]) oder beschränken sich auf die Visualisierung gängiger Programmanweisungen (vgl. [3]). Die grafische Programmiersprache *Labview* konzentriert sich auf die Domäne System-, Steuer- und Regelungstechnik (vgl. [21]). Programmiert wird, indem Elemente miteinander kombiniert werden, die als Schaltzeichen aus der Elektrotechnik bekannt sind. Nach diesem Prinzip arbeiten auch die Editoren *Matlab Simulink* und *Choreograph*. (vgl. [7] [17])

Ist keine domänenspezifische grafische Programmiersprache gewünscht oder bekannt, ist es dennoch möglich ohne tiefergreifende Programmierkenntnisse Programme zu entwickeln. Ermöglicht wird dies durch Programmiersprachen, die Programmanweisungen durch Diagramme oder Blöcke visualisieren. Für Diagramme werden unter anderem Zustandsdiagramme oder eine Form von Flussdiagrammen verwendet. (vgl. [19] [9] [15]) Durch diese Vorgehensweisen lassen sich Schleifen oder Bedingte Anweisungen leicht erkennen. Eine Visualisierung mit Blöcken hingegen folgt dem Steckprinzip. So können Anweisungen in Blockform nebeneinander wie untereinander angeordnet werden. Schleifen oder Bedingungen werden durch Blöcke dargestellt, die andere Blöcke beinhalten. Diese Blöcke stellen Anweisungen dar, die innerhalb dieser Schleife oder jeweiligen Bedingung ausgeführt werden. (vgl. [4] [13] [18] [24]) *Lego Mindstorms NXT* verbindet das Steckprinzip der Blöcke mit domänenspezifischen Elementen der Lego Mindstorms Bausätze. Insbesondere Schleifen und Bedingungen werden als eine Art Blocksystem genutzt. (vgl. [13])

TODO: Weiter ausführen

3.3 Auszeichnungssprachen

Eine weitere Möglichkeit zur komplexen Programmierung sind die sogenannten vereinfachten Auszeichnungssprachen. Diese arbeiten mit Text der anhand einfacher Befehle formatiert und strukturiert wird. So kann anhand eines vorangehenden

Symbols Text als Überschrift definiert werden. Insbesondere *Markdown* verwendet Sonderzeichen um Textabschnitte zu formatieren und strukturieren. (vgl. [11])

Auch *YAML* nutzt Sonderzeichen, um Listen und größere Mengen von Daten zu beschreiben. (vgl. [20]) *BBCode* hingegen verwendet einfache Anweisungen die mit eckigen Klammern eingeleitet und abgeschlossen werden. Die Anweisung selbst wird in Form eines Buchstabens angegeben. (vgl. [1])

HTML ist die bisher geläufigste Auszeichnungssprache. Diese wird zur Strukturierung von Websites benötigt. Dabei können verschiedene Bereiche Definiert und deren Inhalte Strukturiert werden. *HTML* hat die Fähigkeit durch die Verwendung von Tags komplexe Inhalte, wie Texte, Bilder, Listen und Tabellen zu strukturieren und zu formatieren. Die Tags werden mit spitzen Klammern gekennzeichnet. Im Rahmen einer Studie wurde *HTML* eingesetzt um Ambulante Assesment Protokolle zu erstellen, die sowohl vom Menschen lesbar als auch vom Computer ausführbar sind. (vgl. [2])

TODO: Weiter ausführen

3.4 Experience Sampling Software

Psychotherapeuten und Psychologen haben die Möglichkeit anhand bestimmter Experience Sampling Software Fragebögen für mobile Geräte zu entwickeln. (vgl. [14]) Hierbei werden auch Lösungen angeboten, die Auszeichnungssprachen verwenden. Die Software *Experience Sampler* verwendet die Auszeichnungssprache *JSON*, aufbauend auf *YAML*, um Fragen, Anzeige- sowie Eingabeformate zu definieren. (vgl. [14]) *MyExperience* verwendet einen ähnlichen Ansatz. Als Auszeichnungssprache zur Entwicklung der Fragebögen wird hier auf *XML* zurückgegriffen. (vgl. [25])

Ein weiteres Projekt zur Erstellung von Experience Sampling ist *Jeeves*. Fragebögen werden mit diesem Programm über eine grafische Programmiersprache definiert. Verwendet wird hauptsächlich die grafische Programmierung mit Blöcken. Über eine weitere Oberfläche werden die Eingabeformate der Antworten festgelegt. So ist es möglich Formate wie Likert Skala, Checkboxes, Radiobuttons, Ortsabfragen und weitere zu verwenden. (vgl. [16])

Die Experience Sampling Software *movisensXS* nutzt Diagramme zur Beschreibung des Ablaufs eines Fragebogens. Diese werden nach einem Puzzle-Prinzip angeordnet. Die Fragen selbst, sowie Formate der Antworten, werden separat angelegt und können später im Diagramm ausgewählt werden. (vgl. [23])

TODO: Weiter ausführen

3.5 Domänenspezifische Sprachen

TODO: Muss noch recherchiert werden! Was gibt es im Bereich der Domänenspezifischen Sprachen? Welche Ausprägungen gibt es? Wie sind diese strukturiert?

3.6 Fazit

TODO: Fazit aller Recherchen erstellen! Was nehme ich aus diesen Recherchen mit? Was hat sich als positiv herausgestellt, was als negativ?

4. Konzeption

4.1 Anforderungsanalyse

4.2 Ausarbeitung verschiedener Konzepte

4.2.1 Beschreibung dieser

4.2.2 Evaluation

5. Entwicklung der Modellierungssprache

5.1 Konzept

5.2 Umsetzung

5.3 Evaluation

6. Ergebnisse

6.1 Zusammenfassung

6.2 Kritische Reflexion

7. Ausblick

Bibliography

[1] *BBCode.org, bbcode users guide and tricks on implementing it.* https://www.bbcode.org/. (Accessed on 11/28/2018).

[2] Nikolaos Batalas et al. "Formal representation of ambulatory assessment protocols in HTML5 for human readability and computer execution". In: *Behavior Research Methods* (2018). ISSN: 15543528.

[3] *Blockly | Google Developers.* https://developers.google.com/blockly/. (Accessed on 11/27/2018).

[4] *Blockly | Google Developers.* https://developers.google.com/blockly/. (Accessed on 11/28/2018).

[5] *Botsify - Create artificial intelligent chatbots without coding.* https://botsify.com/. (Accessed on 11/27/2018).

[6] *Chatfuel dashboard.* https://dashboard.chatfuel.com/#/bot/5bea8fd30ecd9f4c8c0dd458/structure/5bea8fd30ecd9f4c8c0dd466. (Accessed on 11/27/2018).

[7] *Choregraphe Suite — Aldebaran 2.4.3.28-r2 documentation.* http://doc.aldebaran.com/2-4/software/choregraphe/index.html. (Accessed on 11/28/2018).

[8] *Converse.AI.* http://www.converse.ai/. (Accessed on 11/27/2018).

[9] *DRAKON Editor.* http://drakon-editor.sourceforge.net/. (Accessed on 11/28/2018).

[10] *Dialogflow.* https://console.dialogflow.com/api-client/#/login. (Accessed on 11/27/2018).

[11] *Getting Started | Markdown Guide.* https://www.markdownguide.org/getting-started/. (Accessed on 11/28/2018).

[12] *Katalog - IBM Cloud.* https://console.bluemix.net/catalog/?category=ai&search=label:lite. (Accessed on 11/27/2018).

[13] *NXT Software Download - Downloads - Mindstorms LEGO.com.* https://www.lego.com/de-de/mindstorms/downloads/nxt-software-download. (Accessed on 11/28/2018).

[14] *OSF | Sabrina Thai - ExperienceSampler SPSP San Antonio 2017 Draft 1 FINAL To Post.pdf.* https://osf.io/3am7g/. (Accessed on 11/28/2018).

[15] *Pure Data — Pd Community Site.* https://puredata.info/. (Accessed on 11/28/2018).

[16] Daniel Rough and Aaron Quigley. "Jeeves - An Experience Sampling Study Creation Tool". In: *BCS Health Informatics Scotland (HIS)* (2017), pp. 1–10.

[17] *Simulink - Simulation und Model-Based Design - MATLAB & Simulink*. https://de.mathworks.com/products/simulink.html. (Accessed on 11/28/2018).

[18] *Snap! (Build Your Own Blocks) 4.2*. https://snap.berkeley.edu/. (Accessed on 11/28/2018).

[19] *SwissEduc - Informatik - JavaKara: Einführung in Java*. https://www.swisseduc.ch/informatik/karatojava/javakara/. (Accessed on 11/28/2018).

[20] *The Official YAML Web Site*. http://yaml.org/. (Accessed on 11/28/2018).

[21] *Was ist LabVIEW? - National Instruments*. http://www.ni.com/de-de/shop/labview.html. (Accessed on 11/27/2018).

[22] *Welcome Message*. https://manychat.com/fb500208663808259/automation/welcome/edit. (Accessed on 11/27/2018).

[23] *movisensXS - eXperience Sampling for Android!* https://xs.movisens.com/. (Accessed on 11/28/2018).

[24] *squeakland : home of squeak etoys*. http://www.squeakland.org/. (Accessed on 11/28/2018).

[25] *the MyExperience tool*. http://myexperience.sourceforge.net/. (Accessed on 11/28/2018).