Institute of Software Technology

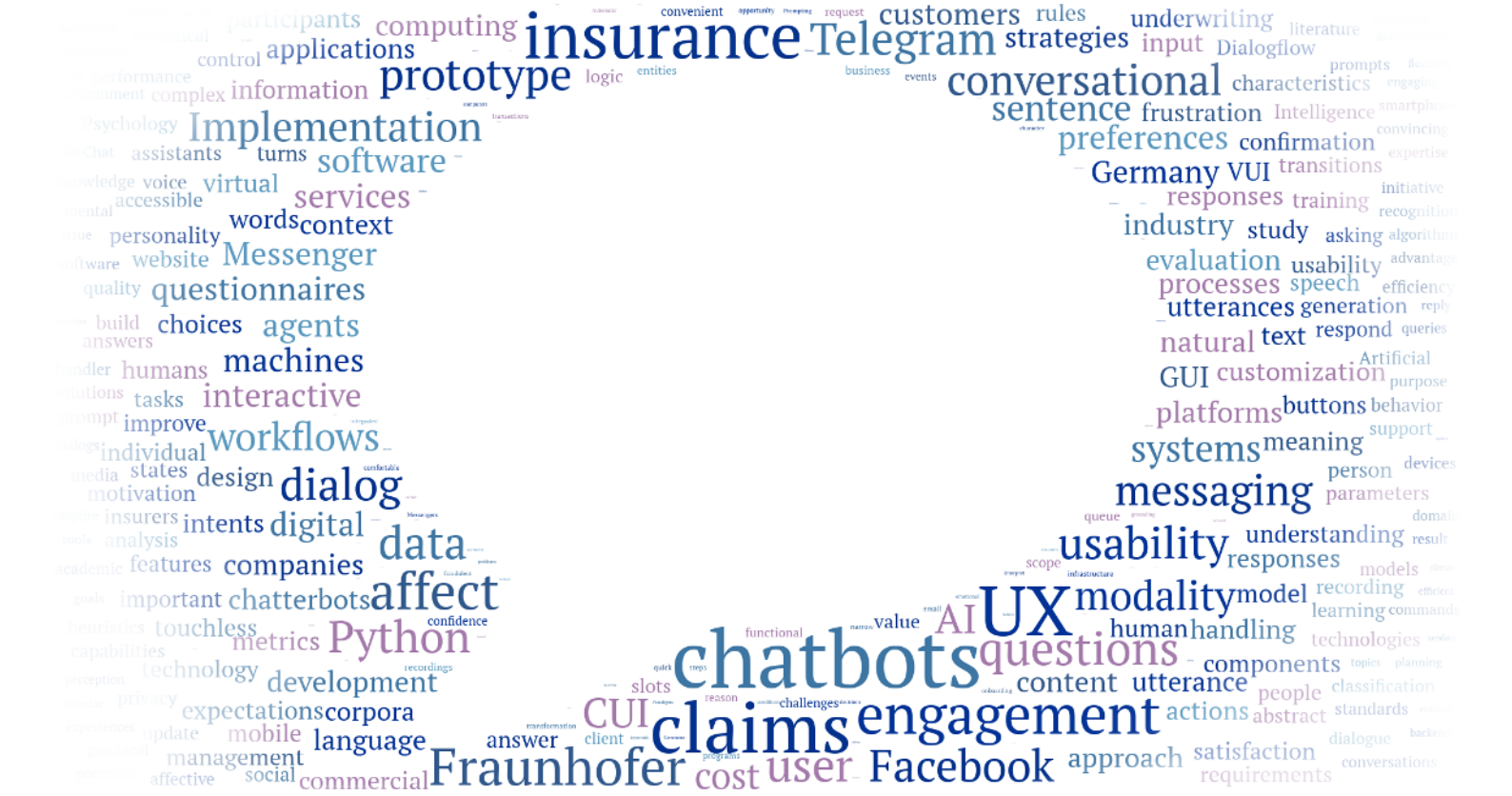
University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Engineering and User Experience of Chatbots in the Context of Damage Recording for Insurance Companies

Joscha Götzer

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Stefan Wagner
Supervisor:	Dr. Daniel Graziotin, Dr. Falko Kötter
Commenced:	October 18, 2017
Completed:	April 18, 2018
CR-Classification:	I.7.2


chatbots
insurance
claims
engagement
UX
questions
Fraunhofer
Telegram
conversational
Implementation
prototype
messaging
usability
modality
dialog
data
affect
Facebook
user
cost
CUI
AI
Python
workflows
Germany
sentence
preferences
evaluation
processes
utterances
natural
GUI
platforms
systems
machines
questionnaires
agents
interactive
digital
intents
companies
chatterbots
development
expectations
corpora
language
management
commercial
social
mobile
metrics
touchless
technology
components
content
utterance
actions
approach
satisfaction
requirements
people
abstract
standards
classification
dialogue
conversations
customers
strategies
rules
underwriting
input
Dialogflow
characteristics
frustration
Intelligence
confirmation
transitions
responses
training
industry
study
asking
usability
speech
generation
respond
text
customization
buttons
meaning
person
parameters
understanding
responses
models
recording
learning
handling
human
value
functional
slots
reason
challenges
client
answer
cost
computing
applications
information
logic
entities
business
events
participants
control
complex
Psychology
assistants
turns
software
virtual
services
voice
accessible
words
context
personality
website
Messenger
quality
build
choices
answers
humans
tasks
improve
individual
states
design
motivation
insurers
analysis
features
important
touchless
capabilities
technology
privacy
update
affective
expectations
mobile
management
social
commercial
language
Fraunhofer
answer
client
CUI
slots
development
chatterbots
metrics
Python
confidence
affect
data
companies
features
analysis
digital
intents
insurers
motivation
dialog
design
states
media
workflows
improve
individual
interactive
tasks
humans
machines
build
choices
agents
answers
questionnaires
quality
website
Messenger
personality
words
context
services
virtual
accessible
voice
software
turns
assistants
Implementation
Psychology
complex
information
prototype
applications
control
participants
computing
insurance
convenient
request
customers
rules
underwriting
literature
Telegram
strategies
input
Dialogflow
prompts
logic
entities
business
events
conversational
characteristics
sentence
frustration
Intelligence
preferences
confirmation
Germany
VUI
transitions
responses
training
industry
study
asking
evaluation
usability
processes
speech
utterances
generation
natural
text
respond
GUI
customization
platforms
buttons
systems
meaning
messaging
person
parameters
usability
understanding
responses
scope
UX
modality
model
recording
learning
human
handling
AI
value
functional
chatbots
questions
components
content
utterance
actions
approach
satisfaction
requirements
people
abstract
standards
classification
dialogue
conversations
claims
engagement
Facebook
user
cost
Fraunhofer
commercial
social
management
affective
update
mobile
expectations
corpora
language
answer
client
CUI
slots
reason
challenges
development
chatterbots
metrics
Python
confidence
affect
data
companies
features
analysis
digital
intents
insurers
motivation
dialog
design
states
media
workflows
improve
individual
interactive
tasks
humans
machines
build
choices
agents
answers
questionnaires
quality
website
Messenger
personality
words
context
services
virtual
accessible
voice
software
turns
assistants
Implementation
Psychology
complex
information
prototype
applications
control
participants
computing
insurance
convenient
request

Abstract

With ever increasing amounts of data and changing consumer expectations, the German insurance sector is undergoing immense transformation. While insurance has traditionally been an industry with very low customer engagement, insurers now face a young generation of consumers who expect quick and on-demand services at a time suitable for them.
For this reason, *specialized digital assistants* (chatbots) could become a first line of support, driven by rapid advancements in artificial intelligence as well as vigorous growth in the adoption of messaging services. They promise to be scalable, accessible around the clock, and to improve customer engagement by orders of magnitude as opposed to traditional channels such as email or telephone.

Another key issue is that insurance claims are currently touched by multiple employees in a process referred to as the *traditional workflow*. In order for insurance companies to remain competitive and become truly forward-leaning carriers, they need to reduce their loss adjustment expenses (LAE) while delivering customer-pleasing solutions in a digital age. Given the prospect of modern data analysis capabilities and conversational user interfaces (CUIs), claims processing in the future may be performed without any human intervention at all, which is known as the *virtual* or, most desirably, *touchless handling*.

First explorations have shown that chatbots are indeed a feasible solution to the aforementioned problems, however they indicated several issues of frustration in terms of user experience, resulting in a direct impact on user satisfaction and task success.
Therefore, this thesis aims to explore the constituents of an engaging and satisfying conversational interface and ways to achieve a better user experience.

In an ongoing project on innovation management in collaboration with insurances, the industry partner Fraunhofer IAO continuously explores ways to advance the business processes of their customers. For this reason, they commissioned a prototypical implementation of a chatbot for smartphone incident recording as part of this examination.
This high-fidelity prototype should be able to record cases of damage in a convenient manner, and ultimately serve as a showpiece to inspire confidence in German insurers that employing chatbots in their workflows would be a viable asset to *target increasing consumer expectations*, *improve engagement*, and *cost-effectively scale their customer support*.

As the details of the eventual solution were not prescribed in any way, it was possible to be more experimental in terms of utilized technologies and a methodical approach to requirement analysis. Hence, the technical and human-centric requirements, which deemed useful to realize in order to achieve the goal of a satisfying user experience, were drawn from systematic review and analysis of existing academic literature and best practices from the industry.
The implemented requirements are thus presented in their appropriate chapters, followed by challenges and design decisions, as well as steps undertaken to assure software quality in past and future iterations of engineering on the prototype.
Thereupon, in addition to an assessment of the functional requirements, a survey on the basis of quality attributes was conducted to evaluate the fulfillment of soft requirements.

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

AHT average handling time. 21

AI artificial intelligence. 17

ASR automatic speech recognition. 45

CA conversational agent. 33

CUI conversational user interface. 5

DA dialog act. 66

DM dialog manager. 56

ECA embodied conversational agent. 34

FSA finite state automaton. 69

GPA general personal assistant. 32

GUI graphical user interface. 39

HCI human-computer interaction. 32, 39

HMM hidden-markov models. 69

IoT Internet of Things. 32

ISU information state update. 65

IVR interactive voice response system. 19

LAE loss adjustment expense. 5

ML machine learning. 32

MPCS multi-party conversational system. 33

NLG natural language generation. 56

NLP natural language processing. 20

NLU natural language understanding. 45

NN neural network. 69

ORM object relational mapper. 64

PDA push down automaton. 72

PIP Python Package Index. 63

POMDP partially observable markov decision process. 69

QA quality assurance. 89

RG response generation. 61

RL reinforcement learning. 69

SDA specialized digital assistant. 32, 34, 37

SDS spoken dialog system. 33

SLU spoken language understanding. 33

STT speech-to-text. 38

TTM time to market. 68

TTS text-to-speech synthesis. 56

UX user experience. 42

VPA virtual personal assistant. 19

VUI voice user interface. 33

YAML YAML Ain't a Markup Language. 65, 73

CHAPTER 1

Motivation

1.1 Insurance and Artificial Intelligence

Since the beginning of the digital transformation era, advancements in software technology have transformed entire industries; from manufacturing to banking, music, travel, and even taxis. The insurance sector is next in line to grapple with the *risks and opportunities* of emerging technologies, in particular artificial intelligence (AI). [NDH+17]

In the mid-1950s, AI researchers first started tackling the challenge of creating computer programs that were capable of *intelligent behavior*. In the following decades, "AI has gone through cycles of euphoria and rejection with some initial successes followed by some spectacular failures." [MCG16, p. 16]
Constantly increasing amounts of data on the Internet, as well as fast GPUs then allowed for more data-driven approaches to AI. Today, faster wifi, high availability of internet, and increasing prevalence of mobile devices allow for resource-intensive algorithms to be performed in the cloud using large banks of powerful computers, meaning that consumers now have ubiquitous access to AI capabilities. [MCG16]
On top of that, "rapid advances in cloud and mobile connectivity are dismantling the technological barriers and reducing the costs associated with establishing global platforms." [16, p. 6]

German Insurance Faces a Technological Gap For the greater part, German insurance companies have been driving very conventional strategies with a strong resistance to change in the past. Inert processes and the absence of competition characterized this industry, caused by already secured revenues through profitable capital markets. [ZR15]

In the recent years however, the conditions have changed tremendously. The capital markets faltered and user expectations started to differ from the traditional to more user-centric experiences, resulting in an strong pressure for change. [ZR15]
Shaped by their experiences with other industries, insurance customers, particularly millennials, now expect quick and on-demand services. [16] "These customers are looking for easier, more convenient ways to meet their needs, be it shopping for favorite items, managing bank transactions or checking out the day's news." [17b, p. 2]

Yet, unlike Amazon and many other product and service providers, insurance is an industry with low customer engagement, as an insurer traditionally has just two touch points to interact with customers: the first is when *selling a product*, and the second is *during the claims process*. A 2014 Morgan Stanley Research and Boston Consulting study found that *consumers interact less with insurers than with any other industry*, so the consumer experience with insurers tends to lag behind others. [NBGC14]

An equally important issue are the organically-grown, aging and legacy IT infrastructures at levels of disrepair with diverse and heterogenous computer systems that need to be replaced or upgraded. [16]
These systems have and continue to produce lots of semi-structured data that lies in most different formats and database systems. As the transformation from big data (simple data collection) to data analysis is now starting to ramp up however, this increase in accessible data and the improved computing capabilities result in an *opportunity for AI to manage this data stock*. [NDH+17]

By and large, as German insurers are in the early stages of catching this wave, nevertheless the adoption of AI in insurance is accelerating and catalysing efforts to improve services and remain competitive. [NDH+17]
"Three quarters of insurance executives believe that AI will either significantly change or completely transform the industry over the next three years, especially through data analysis and insight", according to a report by Accenture [17a].
On the other hand, the industry faces challenges integrating AI into its existing technological infrastructure due to issues such as data quality, privacy, and compatibility. [16]

The report notes that "the abundance of [structured] data fused with unstructured data" can be leveraged to "increase customer engagement, create more personalized services and more meaningful marketing messages, sell the right product to customers and actually target the right customer." Insurers have begun to use AI technologies as a tool to help improve overall customer experience, with the technology enhancing the way sales and services are executed, facilitating faster claims processing, and enabling more accurate individual risk-based underwriting processes.

The report concludes that "Artificial Intelligence has the potential to disrupt the entire insurance value chain."

Chatbots to Rescue Customer Engagement Along with new ways of marketing, pertaining to underwriting and fraud detection, a particularly interactive kind of AI technology

has gained momentum in the areas of customer service and claims in the recent years: *Chatbots* and *Digital Assistants*.

Chatbots are digital services capable of holding natural conversations with humans that answer questions, give basic advice, and address common inquiries and transactions – thereby allowing human agents to handle the more complicated situations and freeing consumers from the rather dreary task of "navigating their way around complicated websites or contact centers" [NDH+17, p. 5].

Chatbots "live" inside of messaging apps, chat windows and, increasingly, digital voice assistants, where they integrate with other services such as workforce management systems [GP16] and product databases to provide automation capabilities for assisting humans. [PG17] Dale defines the basic concept of bots as "[achieving] some result by conversing with a machine in a dialogic fashion, using natural language." [Dal16, p. 1]
As these bots reply back using the same application, they "alleviate the interface diversity concern through their familiar, intuitive, and widely used interface" [HFSA16, p. 2]. CUIs allow customers and companies to interact via text messages, which is "a universally understood method of communication. [...] Instead of calling, emailing or opening an app, customers can easily reach out to companies by texting, at a time suitable and comfortable for them. "[Eeu17, p.2]

For example, airline customers can now check in, ask questions, and request flight information updates via Facebook Messenger. Likewise, fashion shops may offer styling advice for their customers by integrating previous purchases or personal preferences, all inside the same messaging app. Moreover, insurance companies start offloading their claims handling to messaging platforms. [Eeu17]

Chatbots in the Industry In the industry, these bots continue to incorporate ever more sophisticated techniques to better understand user questions and provide more relevant and useful responses [GP16], while their areas of expertise can range from very broad services in general AI assistants to more narrow, task-specific helpers.

In a rather focused scope, i.e. a specific area where solutions are well-known and predictable, chatbots in the industry often generate their answer by traversing decision trees, rather than requiring more general intelligence. [MCG16]

Previously, the interest in conversational interfaces was limited to relatively small niche companies and enthusiasts, mostly as interactive voice response system (IVR) systems over telephone in the 1990s. [MCG16]

These programs could automatically handle *simple* requests on the phone and allow large corporations to efficiently scale up their customer care services at a reasonable cost. With the evolution of speech recognition and natural language technologies however, IVR systems rapidly became more advanced and enabled the creation of complex dialog systems that could handle natural language queries and many turns of interaction. [MCG16]

Now, many of the largest companies in the world compete to create their own virtual personal assistants (VPAs). These systems allow profiling of users, enabling to promote e-commerce services and thus gaining a competitive advantage. As has been predicted in a number of studies, the global market for VPAs will "increase dramatically" over the course of the next few years. [MCG16, p. 20]

Calling bots or VPAs "messaging services" understates what they provide, as they now include integration capabilities such as ordering, booking or payments, which would normally require their own website or app. Therefore, instead of acquiring a number of apps for the task, users can order goods, book restaurants or plain tickets, and pay for these services without ever leaving the messaging application.

On top of that, the phenomenon called *cycle of increasing returns* ensures that, as the performance of these systems improves, more users are likely to be drawn to them, yielding more data to analyze and thus the ability for providers to improve their systems; which in turn causes even more people to want to use it. [MCG16]
Therefore, a large uptake with enhanced functionalities and performance can be expected for some time to come. Two thirds of the insurance executives surveyed in the Accenture report said they now use some sort of chatbot in at least one business area to create customer interactions. [NDH+17]

Messaging as an efficient means of Communication One characteristic of the changing consumer expectations is the *preference to communicate via messaging* as opposed to on the phone or via email [Dal16], which can be deduced from a significant rise in the adoption of messaging apps (refer to Section 2.1 on page 31).

As Nordman et al. put it: "For most people, calling an insurance company isn't among their favorite activities. That's because the insurance industry is one of the least innovative areas for customer experience, meaning that customers typically come away from their interactions disappointed and dissatisfied. With chatbots, users are able to receive the information they need, when they need it, in their preferred medium. [...] Insurance can be intimidating, especially for young consumers. Chatbots help make the entire experience of purchasing insurance and making claims more user friendly." [NDH+17, p. 5-7]

Through natural language processing (NLP) and AI, bots have the ability to ask the right questions and make sense of the information they receive. Chatbots are able to take customers through a conversational path to receive the information they need, all without waiting or needing to talk to a person. These properties give bots a valued advantage over a website or an email campaign [NDH+17] (see Section 1.4 on page 24).

Human Handover Customers often use social media channels to escalate an issue in the expectation of a *human* response, instead of an *automated* one. Well-designed systems may use sentiment analysis to identify these cases and divert them to human agents. By using this approach, emotional issues can be mitigated, as "customers are not likely to be already stressed." [GP16, p. 7]

Taking a customer-first perspective and employing a chatbot for distribution of their services, insurers can improve efficiency and mitigate risk by loading off frequent use cases to a scalable computer system and freeing up human agents to handle the more complicated issues, as these routine queries make up most service requests. [New17]

Another Accenture Digital report names an example: "[In a] european telco, a chatbot was used in a pilot program on a set of common customer queries and resolved 82% of interactions by itself, rising to 88% of interactions when combined with live intervention by a human agent. This level of performance was reached after approximately five weeks of training [...]" [GP16, p. 5]

Nowadays, most customer service functions are measured on how well they can reduce the number of customer calls and average handling time (AHT). With a well-designed chatbot, "the marginal cost of handling more conversations and taking more time with customers falls to almost zero."[GP16, p. 5]

1.2 Chatbots for Automated Claims

In addition to pressure from customers who are looking to save time and desire an easier, more convenient claims experience, there are a number of dynamics driving change in today's claims industry; these include increased claim frequency and severity, a looming talent shortage, and intensified pressure to reduce the costs associated with claims handling. [17b]

"Mobile and virtual claims capabilities - positioned as time savers for customers - are driving a digital revolution in claims processing. Millennials, with their preference for digital, are demanding an increase in virtual services; and savvy carriers are meeting those needs." [17b, p. 4]

In a 2017 whitepaper about the future of claims, Lexis Nexis describe four terms that emerged to categorize the advancement of insurance companies by the progress of digitization in their claims processes/workflows. Although the paper is oriented at the automobile insurance sector, where properties of higher value are insured than for example in smartphone insurance, the findings nevertheless offer a good impression of where the industry is headed as a whole.

1. Claims are currently touched by multiple employees. This is referred to as the **Traditional** workflow, where an adjuster goes into the field, inspects the subject under claim, and prepares an estimate with the audit and payment also being handled manually.
2. The so-called **Fast-Track** process is designed to expedite claims handling with minimal insurance carrier employee involvement — for example, a direct repair program (DRP) in which a body shop handles the inspection and repair estimate.

3. The more digitized **Virtual Handling** is a process or workflow in which either a customer or vendor captures photos or streams videos of the damage that allow a claims adjuster to conduct the damage assessment remotely. No insurance carrier employee conducts a physical, in-person inspection of the subject.

4. The desirable **Touchless Handling** is a process similar to Virtual Handling, except no human intervention is involved in the process at all. This workflow uses artificial intelligence and other technology to report the claim, capture damage or invoices, audit the system, and communicate with the customer electronically. If the claim meets approved criteria, it is automatically paid without a carrier employee in the loop.
   This approach offers great potential with regards to customer satisfaction, as the process could allow clients to file claims with minimal effort.

Automated claims processing has proven to yield significant reductions in cycle times, and "while there hasn't yet been a complete shift to Virtual Claims handling, carriers who want to remain competitive will need to make the move to virtual and consider touchless processing if customer preferences are any indication." [17b, p. 2]

The report warns about practitioners' strategies resisting to employ touchless handling when it is justified on the concern of giving up the human touch entirely, which it considers "a concern not based so much in fact but rather on the perception of potential disengagement with customers". Customers' perceptions about engagement and brand value might be different from their own, the report goes on to say, in that they preferred to interact with businesses that "embrace the wave of the future when it comes to delivering digital services". [17b, p. 8]

The Fraunhofer IAO[1] corroborates these findings in their 2012 publication "Projekt openX-change", supposing that the optimization of claims processes by insurance companies trends toward an electronic handling of adjustments. They deem important to support all partner companies in their service network in the participation in this area, regardless of available resources for IT investments and technical/domain know-how. [al12]

According to Lexis Nexis, a majority of interviewees in their study mentioned either offering or piloting the option of allowing customers to report claims via a mobile app. They state that a virtual handling approach required *simplicity with minimal clicks* to complete an application, and that the current state of these input processes was not simple enough.
An opinion that emerged throughout the writing of this thesis is that it would be fairly unintelligable to use a mobile app for this purpose, due to the reasons described in Section 1.4 on page 24. Instead, a chatbot interface should be deployed for virtual handling and touchless claims, as bots, with their rich media capabilities, high scalability, availability and no need for installation, are predestined to be the user-facing agent that handles future claims processes. Chatbots also provide a tangible way for companies to understand how the rules of digital customer service change as we move into the Al computing era. Digital

[1] https://www.iao.fraunhofer.de/lang-en/

services are becoming more sophisticated and contextually aware, making them able to anticipate and respond to individual needs.

It can be concluded that there is a strong and growing interest in chatbots for claims, fueled by the promise of intelligent digital assistants being always available to resolve requests cheaply, quickly, and consistently. [GP16]

Fraudulent Claims

Claim processes powered by AI could also fight against one of the most costly elements of the insurance industry: fraudulent claims, which cost the German industry more than 4 billion euros in about 2.4 million cases per year [18], with every tenth claim being fraudulent [Sac16].

Instead of relying on humans to manually comb through reports to catch inaccurate claims, AI algorithms have the power to identify patterns and trends in the data that humans may miss and recognize fraudulent activity, and thus save companies money. [Cam17] In fact, AI does have the potential to outperform humans in this task, currently shown in prototypes through employment of artificial neural networks[2]. [NDH+17]

"In the short term, AI can help insurers automate some of the routine administrative processes done manually and could help with underwriting, fraud and claims processing. In the long term, as AI technology evolves, insurers may need to not only address data quality and privacy concerns, but also revamp their IT architectures to support AI features and technical dependences. They will need to identify partners, hire or train for new skill sets, and put new development processes and infrastructure in place." [NDH+17, p. 7]

1.3 Conversational Commerce

Applying messenger chatbots for commercial purposes is the beginning of a development called *conversational commerce*, where chatbots allow to interact with people, brands, and services by responding with personalized recommendations, call-to-action buttons, product carousels, updates or links. [Eeu17]

As mobile phones are typically owned by an individual, mobile devices are an ideal platform for personalized and targeted marketing. By means of text messaging, mobile advertisements, mobile (user-generated) content, and permission based marketing, *mobile marketing* can be applied to improve the relationship of a customer with a specific brand. [Eeu17]

The value of the bidirectional, asynchronous messaging context is that it allows to combine many different mobile services that would traditionally each have required development of

[2] https://www.raconteur.net/business/how-ai-spots-fraud-quicker-than-people

their own interface. Therefore, mobile marketing allows companies to reach consumers in relatively easy and inexpensive ways. [Eeu17]

Regarding chatbots in the insurance sector, conversational commerce means an opportunity to sell additional products ("cross-sell") or better versions of the product the customer already has ("up-sell") by chiming in with personalized product recommendations in the most appropriate situations. As there is a high potential to annoy the customer when the situation is not fitting for an advertisement, some service providers focus on analytics in conversational commerce to detect the perfect circumstances for selling a specific product by means of machine learning.

Apart from bringing the topic up for completeness, this thesis is not examining conversational commerce further since its focus lies primarily on automated claims processing.

1.4 Choosing Bots over Apps and Websites

> *"CUIs may result in a fundamental shift that is going to change the types of applications that get developed."*
>
> — Sam Lessin on *Bots, Conversational Apps and Fin*

This section outlines the characteristics of typical software solutions for a mobile presence of a business or organization, accessed via handheld devices such as smartphones or tablets as these are the predominant medium of internet access since 2016. [3] [4]

Before the emergence of chatbots, a typical question to be asked when companies wanted to connect with or provide services to their customers was whether to 1) build a responsive website or 2) make a mobile app. [17c] Now, as a third option, "chatbots make it easy and fast for customers to reach companies using the same messaging services they use daily." [Der17]

Compatibility

Whereas mobile apps are usually bound to their respective operating system (Android, iOS), mobile websites and bots are inherently portable across devices since their host applications (browser and messenger, respectively) are typically available on a vast number of platforms. As a result, website URLs and hyperlinks to share a chatbot with others are easily integrated within other mobile technologies such as SMS, QR Codes and near field communication

[3] http://bgr.com/2016/11/02/internet-usage-desktop-vs-mobile/
[4] http://www.telegraph.co.uk/technology/2016/11/01/mobile-web-usage-overtakes-desktop-for-first-time/

(NFC). Facebook Messenger and Kik both even provide their own QR Code implementations to share the globally unique identifier of bots built on their platforms. [5] [6]

Application Sandboxing

Arguably the greatest advantage of bots and websites, and also the most important reason why they have gained acceptance at this rapid pace, is the fact that these kinds of programs *do not need to be installed*.

In mobile websites and especially single-page Javascript apps, code is interpreted in the browser and no binaries need to be shipped, thus enabling immediacy by defeating the need for installation. Furthermore, as content is queried in real time from the server, providers gain the opportunity to *update content quickly*. On top of that, bots and websites are instantly accessible to users across a range of devices, facilitated by the platform agnosticity of the host systems.

This behavior is termed *Application Sandboxing* and lends its name from the children playing pit, depicting that users do not need to make persistent changes to their system environment as all the necessary program data for such applications is kept in memory.

Since the host systems for applications in a sandboxed environment (messenger and web browser) are usually very well tested and the providers of such containers take great precautions to **ensure security**, the same can be assumed for the hosted applications, so long as no major flaws are made in the implementation.

Mobile apps, on the other hand, require the user to first download and install the app from an app marketplace before the content or application can be viewed. Certainly the app may pull content and data from the Internet, in similar fashion to a website, or it may download the content so that it can be accessed without an Internet connection. [Sum] However, the initial discovery and acquisation phase through the marketplace constitutes a significant barrier between initial engagement and action/conversion.

In March 2015, Gartner published a report showing that app usage is going to plateau, as many smartphone users are becoming exhausted and do not want to increase their current usage levels. [Gar15] This development is called “App Fatigue”, which makes it increasingly harder for companies to gain foothold with their target audiences. Within the past decade, the market has become saturated with ecommerce apps, games apps, social media apps, and many more.
Audrey Thompson describes it in the following words in an article on unifiedinbox.com:

> “You can literally become app fatigued! App fatigue is real. Even the most organized person who can always get the job done gets overwhelmed with all the choices and features that apps offer. But just like the gold rush, this too will

[5] https://botsupport.kik.com/hc/en-us/articles/225604127-What-are-Kik-Codes-and-how-do-they-work-
[6] https://developers.facebook.com/docs/messenger-platform/discovery/messenger-codes

> end one day. The market is now hindered by all the technology available. [...] New apps aren't providing consumers the value they expect, most are nothing more than a replication of an existing app with a different design. " [7]

The boundaries blur between the issue of App Fatigue and a related situation called "App Overload", in which users perceive their phones' homescreens as terribly cluttered with large numbers of apps (cases of 100 to 150 apps are commonplace) that need to be cleaned up (i.e. uninstalled) in order to salvage storage space or to regain a good consciousness about and control over their phones. [8]

Hub applications and social networks (Reddit, Facebook, Quora, Messengers) are likely to remain on users' phones as they provide a conglomeration of contents from different sources and thereby offer enough value, but apps for more specific purposes are likely to go under the knife (e.g. venue, company, branded, shopping apps).

In contrast, chatbots run on a shared platform without the need for installation. This means that the conversation with a bot can always be stopped and re-opened by sending a new message, much like your chat thread with an old friend has faded into silence, but is now suddenly relevant again because you reconnected in person.

Information Flow

A website is predominantly used to transmit text, images, audio and video with the information being directed towards the user who must filter the information, navigate the site on his or her own behalf and react to the data presented. The efficiency with which users are able to traverse a website and find the information they are looking for, is a direct result of its *level of usability*.

Mobile Apps have a similar flow of information, but provide slightly better tooling for a frictionless experience, such as standardized locations for the search functionality, easier access to the content with a launch button on the home screen, and features that require an integration with the native capabilities of the operating system (sensors, voice input, intents, hardware). Chatbots and websites cannot compete with the integration capabilities of a mobile app.

However, with chatbots, the problem of requesting specific data is inherently solved by the conversational character of the interface, providing a two-way stream of information between the user and the system. Users can ask a chatbot a specific question and the system will respond with the result immediately, or ask a follow-up question to narrow the search space. That way, chatbots can help customers sift through data and products on the internet and help make decisions.

[7] https://unifiedinbox.com/app-fatigue-is-there-an-app-for-that/
[8] https://hackernoon.com/heres-a-quick-way-to-solve-app-overload-7a16ddbb574

On top of that, “we are entirely comfortable communicating via short typed interactions, and quite unfazed by carrying on several asynchronous conversations at the same time.” [Dal16, p. 5]

Tasks like sending a text message, updating calendar events or setting an alarm would often require multiple steps to complete in traditional GUIs, using touch, scrolling and text input. Using a conversational interface, they can often be achieved with a single written or spoken command.

While this is an excellent way of requesting information in a narrow areas of expertise, some of the positive factors of websites and apps may be lost. For example, the aesthetic beauty of a well-designed website or app is unreached by chatbots always looking the same on their respective conversational platform. As a consequence of the limited space and visual elements available in a chat interface, it can also be a lot harder to create an easily navigatable overview or menu structure. However, this largely depends upon the platform the bot is built upon. The *Telegram Messenger* for instance provides a set of virtual keyboard button markups that are available in the client applications of all supported platforms, and it is very common to see multi-layered nested menus with contextual information in bots built with the Telegram Bot API (a.k.a. “click bots”, Section 2.2.5 on page 37). We are going to examine the issue of information overload in more detail later in Section 3.1 on page 39.

Time and Cost

Considering time and cost-effectiveness, mobile websites are considerably cheaper than development of a native app, especially when a presence on different platforms is needed (requiring development of multiple apps). [9]

While website builders exist, those are usually not very sophisticated and ambitious companies with specific service requirements will probably want to build their own solution or outsource the task to a third party.

Things lay slightly different with self-service bot building platforms. As development is relatively cheap and quick, hosting is often inclusive, and the bot is often deployable to many different messenger platforms without any effort, “self-service platforms are a viable solution to get up and running quickly. Some platforms require a degree of technical knowledge, while others allow to craft a bot without writing a line of code.” [KV17]

If more sophisticated logic and integrations are needed in a bot, a great number of services for specific components of a custom-built chatbot are available. This includes natural language understanding engines (refer to Table 6.1 on page 68), translation between

[9] https://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/

messaging platforms ("build once, deploy everywhere")[10], and bot-to-human handoff integrations [11] [12].

As natural language processing services may have clauses that enable the providers to improve their underlying AI models based off their customers' data, these services are often cheap or free. Therefore, a precautionary stance must be taken if data privacy is essential.

As the prototype implementation in this thesis will show, those readily available tools can be exploited to build a custom backend that ties everything together into a functional chatbot system; by a single developer, in reasonable time.

Prospect of this Thesis

This examination includes engineering of a prototypical chatbot application for recording smartphone claims. Its implementation was fairly free in the details since there had not been any requirements stated other than that it should be a workflow with a pleasing user experience, which allowed to be somewhat experimental in terms of the actual functionality.
Therefore, the actual requirements were methodically gathered throughout writing of this thesis, along with desirable characteristics and use cases. Thus whenever there is a particularly important functional or domain-specific requirement in the context of a chapter, it will be depicted by a box like the following:

Requirement CATEGORY

Example requirement to be engineered or characteristic to be explained.
The prototypical implementation will subsequently be referred to as "*the* prototype" or "*the* bot".

Chapter 2 on page 31 – Background: A historical and functional classification of CUIs.

Chapter 3 on page 39 – User Interfaces and Experience Considerations for the design of satisfying and engaging CUIs.

Chapter 4 on page 55 – Related Work A review of existing literature on CUIs and user experience studies.

Chapter 5 on page 59 – Challenges Technical challenges in the design and engineering of the prototype.

[10] http://message.io
[11] https://chatbotsmagazine.com/the-bot-human-handoff-34fd1808731
[12] https://docs.microsoft.com/en-us/bot-framework/bot-service-design-pattern-handoff-human

Chapter 6 on page 63 – Prototype Implementation Introduces the eventual chatbot and explains its architecture and individual components.

Chapter 7 on page 81 – Evaluation and Discussion Presents the evaluation of the gathered requirements and the system as a whole, as well as a survey on user experience with the prototype.

CHAPTER 2

Background

In order to make educated judgements about chatbot technology, not only is it necessary to address the historical milestones that lead to today's prevalence, but also to classify the characteristics of conversational interfaces. Thus, this chapter examines chatbots in their early days and lists the key developments catalyzing their current growth. In addition, there is an exhaustive set of *classification attributes* and *super categories* for conversational interfaces.

2.1 History

One of the earliest NLP applications was Joseph Weizenbaum's Eliza, which he built in 1966 at MIT to simulate a therapist by using a script to respond to a user's typed questions with simple pattern matching. A simple response mechanism emulated the conversational style of a non-directional psychotherapist. [Cah17; GP16; Wei66] Sparked by the apparent success of Eliza fooling people into thinking that she was a real person rather than a machine, a whole community of interest evolved around the idea of creating a bot capable of passing the Turing Test[1]. The *Imitation Game*, as Turing called it, was a replacement for the question "Can machines think?" and he predicted that with the computing capabilities available by the year 2000, computer programs would be able to fool an average dialog partner into thinking that it was not a machine, but rather a human. This development finally lead to the annual Loebner Prize contest, where developers compete since 1991 to create a chatterbot (2.2.1) that could pass the Turing Test. Ever since then, "the prospect

[1] http://m.mind.oxfordjournals.org/content/LIX/236/433.full.pdf

of having meaningful interactions with AI agents has been firing human imagination" [MJDV16, p. 1]. [Cah17]

IVR systems, first introduced in the 1990s, were the first chatbots that could automatically handle simple spoken requests over the phone and thus allowed corporations to scale up their customer care services at reasonable cost. Soon, these became rather sophisticated and enabled complex dialog systems with many turns of interaction, causing the industry to create standards such as VoiceXML[2] to be able to share domain knowledge and enable collaboration. [MCG16]

In the recent years, two more key developments caused another shift in the CUI industry:

1. **Messaging Services Growth** — As messaging apps such as Facebook Messenger, Telegram, WhatsApp, Kik and WeChat experienced a significant rise in adoption, they now represent the most popular means of communication. In 2015, the total number of active users of messenger applications surpassed the total number of users in social networks, with WhatsApp, Telegram, Messenger and WeChat each reaching between 400 million and 1.2 billion people.[3] Judging from these numbers, it makes sense for companies to reach out to their customers through their preferred medium.
2. **Sophisticated Artificial Intelligence** — With previously only moderate results, AI researchers began to find ways of improving their algorithms, largely due to advancements in machine learning (ML). Since 2006, Deep Learning methods outperform other statistical methods in speech recognition, as well as image and natural language processing. [MCG16]

Finally, the amalgamation of growth in messaging services and advances in AI lead to the modern general personal assistants (GPAs) such as Amazon Alexa, Apple Siri and Google Assistant. With spoken commands, unlimited requests, and operation from the cloud or on a handheld device, these services enriched the human-computer interaction (HCI) by defining new ways to interact with knowledge repositories on the web, and later the Internet of Things (IoT). [MCG16]

In 2016, chatbots emerged as "one of the hot topics of the year" [GP16, p. 2] , bringing the topic to the attention of individuals and smaller, independent companies who then started to create their own bots [GP16]. William Meisel, a renowned representative of the speech technology world, distinguishes the aforementioned GPAs from the vast number of creations by the bot community, which he calls specialized digital assistants (SDAs). He predicts that the latter category will generate global revenues of $7.9 billion in 2016, rising to $623 billion by 2020. [Dal16]

When MIT Technology Review listed conversational interfaces as one of the ten breakthrough technologies of 2016[4], Microsoft CEO Satya Nadella publicly announced chatbots

[2] https://www.w3.org/TR/voicexml20/
[3] https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/
[4] https://www.technologyreview.com/s/600766/10-breakthrough-technologies-2016-conversational-interfaces

to be the “next big thing”, on par with the graphical user interface, the web browser, and the touch screen.[5]

2.2 Classification

For the sake of deciding and reasoning over the feature set and properties of a chatbot (and the eventual prototype), a number of distinctive consideration points were collected to form a framework for the classification of CUIs.

2.2.1 Types of Conversational User Interfaces

There are a lot of terms being used to broadly refer to a conversational user interface, including “chatterbot”, “intelligent virtual assistant”, “cognitive assistant”, “conversational agent”, and many more, whereas “chatbot” is an umbrella term that is used in most different contexts.

The following list is an attempt to clarify the similarities, differences and ambiguities amongst these types of systems. These terms mean that a system is rather specialized in one or another way, and multiple categories may apply to an individual conversational system, for instance a *multi-party conversational agent mainly controlled by voice input* is possible.

(Natural) dialog systems Systems that try to improve usability and user satisfaction by imitating human behavior [Spi05]

Multi-party conversational systems (MPCS) Systems that are able to interact with one or more people or chatbots in a multi-party chat [Spi05]

Conversational agents (CAs) Intend to carry out tasks, have a long term memory and personal knowledge of the user, enabling to create the foundation for a longer relationship between agent and user [Wil10].
Examples: *MOOCBuddy* [Hol16], *knowie* [Bii13] (Personalized Learning Assistants)

Voice user interfaces (VUIs) Broad term to refer to conversational interfaces that are strictly controlled by spoken language understanding (SLU) [MCG16, p. 52]

Spoken dialog systems (SDS) Similar to VUIs, but developed in academic laboratories [MCG16, p. 52]

Social robots Physical machines in a various forms of embodiment that provide services, companionship, entertainment

[5] http://www.businessinsider.com.au/microsoft-to-announce-chatbots-2016-3

Userbots Chatbots that do not use a messaging platform's Bot API, but instead take control over a regular user account and interact in its name[6]

While the above terms are classification attributes of an individual conversational system, the following types of CUIs are usually encountered as mutually exclusive categories and constitute the most popular chatbot systems (Figure 2.1).

Chatterbots Conversational systems with emphasis on providing realistic simulations of casual conversation or small talk; not task-oriented, only "chatter"

(Virtual, Intelligent, Cognitive, Digital, Personal) assistants (VPAs) Intelligent conversational systems that maintain records on personal preferences and perform tasks on behalf of the user by reasoning intelligently over written or spoken input and yielding concise results by sifting through data repositories. [CMR+08; Nao09] Examples: "The Big Four" - Apple's Siri, Microsoft's Cortana, Amazon's Alexa and Google's Assistant [Dal16].

Specialized digital assistants (SDAs) Goal-oriented conversational systems with a narrow focus or area of expertise [Dal16]

Embodied conversational agents (ECAs) A type of SDS in which computer-generated animated characters (animals, avatars, humans, humanoid robots) [RB17] combine speech with other modalities including facial expression, body stance, and hand gestures.

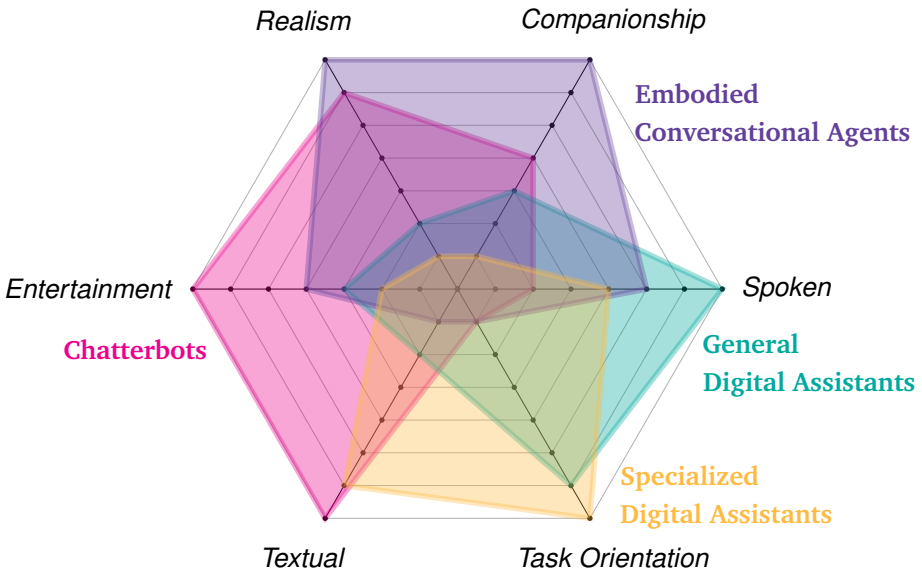


Figure 2.1: Characteristics of the four most common types of conversational user interfaces

[6]For more information see http://telegra.ph/How-a-Userbot-superacharges-your-Telegram-Bot-07-09 - Disclaimer: An article of the author's

This thesis specifically targets Chatbots on conversational platforms, and explicitly not crawlers, spam protection bots, botnets for malicious purposes and so on, since those are programmed to perform repetitive, dreary tasks unobserved and hardly ever interact with humans in a meaningful way. [MJDV16]

Classification

The prototype can safely be inserted into the SDA category, as it allows users to perform the specific task of *recording a claim*. However, it can also hold smalltalk which nudges it closer to the area of Chatterbots, in addition to voice input capabilities, which give it a VUI characteristic.

2.2.2 Conversational Platforms

Conversational platforms are messaging services that host chatbots.

Custom-built Websites One of the earliest forms of chatbot interaction. The bot has its own frame embedded into the website and often also an avatar attached.
Examples: Shopify Virtual Assistants[7], DoNotPay[8]

Messengers Chat-apps for a variety of operating systems.
Examples: WeChat, Telegram, Kik, Facebook Messenger, Line

Intelligent Platforms Voice-driven digital assistants. Users can submit their own *skills* or *actions* which perform in subprocedures as part of the main assistant interface. [MCG16]
Examples: "The Big Four" (2.2.1), as they provide capabilities for third-party developers to submit their own skills or actions.

Bot Building Platforms Fully-featured services that offer tools to build and host chatterbots on their websites.
Examples: *Pandorabots*[9], *Cleverbot*[10]

Conversational Platform INTEROPERABILITY

The prototype is available on both the Telegram and Facebook Messenger applications. Facebook Messenger is the de-facto standard to build bots on, at least in the western hemisphere where it has the largest user base amongst its competitors. As bots are available by simply messaging a Facebook Page, discoverability is easy to

[7] https://freeeup.com/blog/advanced-online-hiring/25-shopify-virtual-assistant-to-hire-for-your-online-store/
[8] http://www.donotpay.com/
[9] https://home.pandorabots.com
[10] http://www.cleverbot.com

achieve.
Demonstrating the bot's interoperability between two platforms is supposed to improve user experience as users are be able to use their favorite platform to interact with the bot. Including Telegram specifically as a supported platform was a decision based on previous experiences with the messenger, the fact that it supports nearly all operating systems including a fully-featured desktop application, as well as the availability of *userbot libraries* to make fully functional integration tests possible by simulating a real user.

2.2.3 Audience

A chatbot might talk in a private chat as a one-to-one conversation, or in a group chat with multiple members. MPCS (2.2.1) are defined as systems that converse with multiple people or chatbots at the same time.

De Bayser et al. criticize that "current existing chatbot engines do not properly handle a group chat with many users and many chatbots. This makes the chatbots considerably less social, which is a problem since there is a strong demand of having social chatbots that are able to provide different kinds of services, from traveling packages to finance advisors. This happens because there is a lack of methods and tools to design and engineer the coordination and mediation among chatbots and humans." [BCS+17, p. 1]

Audience FUNCTIONALITY

The prototype is capable of holding a one-to-one dialog with a user, contingent upon the use case, and the fact that Facebook only recently started rolling out their group messaging capabilities[a].

[a] https://techcrunch.com/2017/04/18/facebook-bot-discovery/

2.2.4 Task and Goal

Every conversation has a goal, and quality of a bot can be assessed by how many users achieve the proclaimed goal. [GP16] Questions to ask may be whether the user has found the information he or she was looking for, or whether a task like booking a flight or hotel was performed successfully. Guzmán and Pathania claim that usually, these metrics are easy to track.

"According to the types of intentions, conversational systems can be classified into two categories: a) *goal-driven or task oriented*, and b) *non-goal-driven or end-to-end systems*." [BCS+17, p. 11]

While VPAs try to cover as many topics and areas of expertise as possible being backed by large companies that have one or another interest in providing these fully-featured services (Google, Apple, Amazon), the industry and independent SDAs normally offer rather specific use cases. These are designed to fit the purpose of the business, are task-oriented, and measured by the task success.

Chatterbots, on the other side of the spectrum, also offer a large area of expertise, but the goal is entertainment and measured by length of the conversation or via the Turing-Test.

Task FUNCTIONALITY

The bot is a Specialized digital assistant (SDA) which allows customers to record smartphone damage claims as convenient at possible. The task success can therefore be measured in terms of how many users are able to successfully fulfill the task and their satisfaction while doing so.

2.2.5 Interaction method

The consistent medium across all messaging interfaces are *text messages*. On top of the purely "text-based" approaches however, various messenger apps offer a multitude of visual containers, controls, and input options for styling and interacting with their respective chat interfaces, such as keyboard buttons, item carousels, inline queries, web views, IRC-like /commands, hyperlinks, voice input, and many more.

For example, a bot can be built to provide very sophisticated nested keyboard menus if it supports editing its own messages as a reaction to button presses. Alternatively (or additionally), a common pattern on many platforms is the "slash-command" (a word with a leading slash), which can be typed in easily with the help of autocompletion and then issues a request or command to the bot. Often, these /commands may also be followed by additional parameters to provide more arguments to the function execution, similar to a terminal command. For example, a weather bot on Slack or Telegram (which both support these commands as marked-up text entities) may accept the message `/weather London`, interpret it as a command, and consequently yield the current weather conditions for the capital of England.

Such chatbots, primarily controlled by structured, graphical interfaces, are commonly referred to as "click-based bots".

However, as many chatbots deploy elements of both worlds, i.e. clickable elements and text/voice input, it is necessary to allow a "hybrid" category that includes the systems not strictly classifiable into either of the above.

In order to create an infrastructure that is platform-agnostic, i.e. the chatbot works on every platform with the same backend, a purely "text-based" interface can be used to provide the cross-platform experience without catering to every platform specifically.

Conversely, experienced users can be expected to have built up a mental model of how bots ought to behave on their platform of choice. They already know how to control bots on this platform, how to get help using the app's interface, and in general how they can get information they need and achieve their goals in a typical interaction.

By adhering to these platform standards and conventions, recurring users will have an easier time getting accustomed in the onboarding phase, as all the "ground work" has been laid out before.

These standards may include:

- using the available help and documentation capabilities
- using the right type of visual container to group information (cards, lists, galleries, carousels, ...)
- providing commands or buttons for the right task
- providing native shortcuts for power users
- reacting to startup commands accordingly (deep-linked URLs, QR codes)
- using web views if available
- providing speech-to-text capabilities

In this sense, we have a tradeoff between time and resources spent developing a cross-platform chatbot system that adheres to the platform standards, and the value of sparing users from having to retrain their mental model.

Interaction Method ACCESSIBILITY

As the prototype runs on both Telegram and Facebook Messenger, a `common subset` of available features needed to be extracted and abstracted. This makes development significantly more complex, as the backend needs to be able to cater the same content to both platforms. Nonetheless, some visual elements were included on top of the mainly text-based interaction: The bot uses *reply buttons* to guide the user and make input easier, and *emoji to convey the state of the system* in a skeuomorphic manner (e.g. when input is necessary, when an action succeeded or failed, or when the bot is about to provide superfluous yet helpful information). Since the prototype should mainly show off what *conversational* interfaces are capable of, a convincing textual interaction was of most importance. However, speech-to-text (STT) capabilities were also included, meaning that users are able to send voice messages that are interpreted as textual input.

CHAPTER 3

User Interfaces and Experience

This chapter presents a collection of considerations for the design of conversational user interfaces compared to ones with a traditional graphical paradigm. Subsequently, requirements, quality attributes and correspdonging metrics, as well as relevant best practices for the claims prototype are derived from the scientific fields of human-computer interaction (HCI), in particular affective computing and conversation design.

3.1 A Shift in UI Paradigms

Today, graphical user interfaces (GUIs) handle the translation between humans and computers. [Sör17]
The most common interaction setup for information visualization are “point and click” WIMPs (graphical user interfaces based on **w**indows, **i**cons, **m**enus and a **p**ointer), which have been the leading way of interacting with computers since they became popularized by Apple and the Macintosh in 1984. However, this way of visualizing and interacting with data is not always suitable for more complex applications and visualizations since the desired data often hides behind multiple button presses and menus. [Sät17]

Ron Kaplan, lead in Nuance Communications NLU R&D Lab in Silicon Valley, gives an example in an article on Wired.com about how “it can take as many as 18 clicks on 10 different screens to make one simple airline reservation while we’re faced with an unwieldy array of buttons, ads, drop-downs, text boxes, hierarchical menus and more.” [Kap13]

3.1.1 Conversational User Interfaces

The medium of bots is a conversation. Consequently, a CUI is a system that accepts text or speech as input, reasons about it with the help of predefined rules or statistical models, and answers the user in an appropriate modality (speech or text); thus creating an artificial dialog between a human and the computer. [Sör17]

Often, the dialogs in CUIs are short *one-shot* interactions consisting of single adjacency pairs per step, i.e. the human asks something or issues a command with the machine responding. Longer interactions are required in many cases however, which may be broken down into manageable subtasks. For instance, in order to make a *flight reservation*, the system may break the task down into *requesting flight details*, and split it again into *asking for departure* and *arrival information*, before making the actual reservation. [MCG16]

Once an utterance has been received, the system has to go through multiple stages of processing. On a high level, the system first has to track and then understand the input in relation to the user's context. The next step is for the system to find an appropriate response to the situation [LM14], often picking from a number of canned sentences in a sentence bank [Sör17]. While there has been some attention in the past towards the automatic generation of responses, particularly with the help of neural networks [Sör17], these response generation systems are not yet advanced enough to be used in production. This holds especially true for the German language, as there was not a single response generation software library readily available by the time of writing.

CUIs are most commonly used in a real-time setting, as chat is by its nature a real-time process. Hence the response time of a question or statement by the user has to be delivered within the next few seconds [LM14].

Response Time FUNCTIONALITY

The prototype should generally deliver answers within a maximum response time of 3 seconds for text, 4 seconds for voice input.

Clarifying the nomenclature, we would in theory have to make a distinction between the terms "conversation" and "dialog".

- A *conversation* is an informal spoken interaction, for instance to exchange news or views. It has the important purpose of development and maintenance of social relationships, i.e. small talk with human-like characteristics and spontaneous natural language. [MCG16]
- A *dialog*, on the other hand, is an interaction with a more transactional purpose. For example, an airline agent in a call center would want you to perform a specific task when calling, such as booking or requesting information about a flight. [MCG16]

Nonetheless, McTear, Callejas, and Griol disregard these distinctions because users are likely to want to speak to conversational interfaces for a variety of purposes, such as performing transactions, asking questions or just chatting, and they use the term *conversation* to refer to all types of interaction with smart devices.

3.1.2 Affordances in traditional GUIs

Definition 3.1.1 (From *The ecological approach to visual perception* by James J. Gibson) *"Affordance describes all actions that are made physically possible by the properties of an object or an environment. A bottle screw cap affords twisting. A hinged door affords pushing or pulling. A staircase affords ascending or descending."*

We can say that GUIs provide a high number of affordances, but tend towards information overload. On top of that, graphical user interfaces are strictly limited to physical screens, whereas most predictions of future technologies project a trend towards the promise of *Zero UI* [1] or *deviceless computing*, where we interact with physical objects by means of gestures and speech and our physical environment becomes our affordance.

Kaplan remarks that "the GUI is forced into a mobile-interface world even as the information and tasks available to us continue to increase. Whether it is because of available real estate or the desire for invisible design, interface screens are increasingly smaller, narrower or simply nonexistent."

3.1.3 Graphical vs. Conversational

In a CUI, the user speaks or writes in a natural way, while the system picks up the user's intent by inferring the most likely meaning of the words.
According to Kaplan, the advantage here is that a CUI allows users "to talk about hypothetical objects or future events that have no graphical respresentation. [...] Instead of pulling up an app like OpenTable, searching for restaurants, tapping to select time, and typing in party size, we can say, 'Book me a table for three at 6 tonight at Luigi's.' " [Kap13]
Shawar and Atwell promised back in 2003 that these systems would make the design of interaction "more intuitive, accessible and efficient." [SA03]

A problem that arises as opposed to traditional GUIs however is the loss of guidance as it becomes harder to convey the current state of the system to the user. The visual affordances of a chat screen are often insufficient to represent the full state of the system and the possible actions therein. Visual elements such as buttons, item carousels and inline web views can help improve, but usually not completely resolve this issue (refer to Sections 1.4 and 2.2.5 on page 26 and on page 37, Section 3.3 on page 49, and Section 3.2 on the following page for deeper examination).

[1] https://www.fastcodesign.com/3048139/what-is-zero-ui-and-why-is-it-crucial-to-the-future-of-design

3.2 Evaluating User Experience and Usability in CUIs

A chatbot conversation is governed by two fundamental facts:

Conversations are ephemeral "Messages are cheap, disposable and become stale with time. A message from a day ago is less valuable than a message from a few seconds ago. Old messages are not up to date. There is no guarantee that older messages accurately reflect the current state of the system. The older the message, the less confident we can be in its relevancy." [Sco16]

Limited real estate "There's a hard limit of how many characters are visible at any one time, and a soft limit of how many words a user can comprehend before becoming overwhelmed." [Sco16]

For these reasons, user experience in chatbots is closely tied to the quality of the interface, which is nothing more than text, rich media, and and the conversation itself. [Sco16]
Now, in order to evaluate the quality of CUIs, a more formal set of definitions is in order:

Definition 3.2.1 (The ISO 9241 concept of usability by Abran et al.)
"The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments."

The definition of User experience (UX) in distinction to Usability by Kashfi, Nilsson, and Feldt is as follows:

Definition 3.2.2 (User Experience analogous to Kashfi, Nilsson, and Feldt)
"UX improves the characteristic of worthwhileness *by focusing on the creation of value and desirable experiences, whereas usability focuses on removing obstacles and preventing frustration and stress."*

In other words, UX is based on the idea that removal of dissatisfaction does not necessarily lead to satisfaction and pleasure, "providing a more holistic approach" to the quality. [KNF17, p. 21]

UX is dynamic and a user's perception of the underlying elements naturally changes over time.
"The user may find a novel feature as old, or a complex feature as simple." [KNF17, p. 4]
Therefore, Kashfi, Nilsson, and Feldt recommend to design and evaluate UX based off different "episodes of experience", which were first introduced by Hassenzahl:

- *Expected experience* (before usage),
- *momentary experience* (during usage),
- *remembered experience* (shortly after usage),
- and *accumulated experience* (over longer period of use).

While the notions of chatbot characteristics are not strictly categorized into *either* the usability *or* the user experience space in the following sections, it seemed necessary to inform the reader about the differences between these terms nonetheless. One cannot do one without the other, and the distinction is largely irrelevant for the purposes of this thesis, as both provide a union of useful guidelines, practices, and heuristics that were exploitable during development and evaluation of the eventual solution.

3.2.1 Evaluation Methodology for Quality Assurance

Industrial chatbots and conversational agents are usually evaluated by applying metrics to certain quality attributes of the "as-is" system, and one or more "to-be" conversational systems under development. This approach allows to compare different versions of a system by highlighting improvements (or deteriorations) which were introduced by changes, and also helps in spotting regression bugs. After picking important quality attributes and selecting relevant metrics, multiple users should participate in sessions with the system to record the selected metrics. [RB17]

3.2.2 Quality Attributes and Metrics

Radziwill and Benton examine the academic and industry literature to provide a comprehensive review of quality attributes for chatbots and conversational agents, and to identify appropriate quality assurance approaches. They gathered an extensive set of 7340 articles, which was refined further by inclusion of the terms "evaluation", "assessment", "quality metrics" and "metrics", eventually yielding 42 scholarly articles, conference papers, and industry/trade magazines relevant to their objectives. They then extracted quality attributes from these sources and grouped them based on similarity, discovering that they were generally aligned with ISO 9241 (3.2.1).

As they recommend to use their quality attributes "as a checklist for a chatbot implementation team to make sure they have addressed key issues" [RB17, p. 13], their recommandation was followed and a number of attributes picked that deemed relevant:

Quality Attributes (and corresponding Metrics) TEST

Category: **Performance**

- Robustness to unexpected input (% of successes)
- Provide appropriate escalation channels to humans (% of successes)

Category: **Functionality**

- Use appropriate degrees of formality (0..100)
- Linguistic accuracy of outputs (0..100)

- General ease of use (0..100)

Category: **Humanity** (the prototype does not have to pass the turing test, but instead discloses its chatbot nature openly)

- Transparent to inspection, discloses its chatbot identity (% of users able to correctly identify)
- Convincing, satisfying, & natural interaction (0..100)
- Able to respond to specific questions (% of successes)

Category: **Affect**

- Provide greetings, convey personality (0..100)
- Make tasks more fun and interesting (0..100)
- Entertain and/or enable participant to enjoy the interaction (0..100)

Category: **Ethics & Behavior**

- Nondeception (0..100)
- Trustworthiness (linked to perceived quality) (0..100)

The implementation phase is followed by a study where participants are asked to interact with the bot in order to record a claim and then answer the above questions. The methodology and results follow in Chapter 7 on page 81. On top of that, the Fraunhofer IAO promised to perform another evaluation by an expert from the insurance industry, which will however happen *after* the release of this examination.

Usability Heuristics

In *"Heuristic evaluation of user interfaces"*, Jakob Nielsen describes a set of ten usability heuristics for the evaluation of user interfaces that "stood the test of time" and are still widely used today. It therefore makes sense to try and base our own evaluations on these well-established heuristics.
Once established which of these heuristics for `classic` user interfaces are directly *applicable to chatbots*, they could be used as best practices in development of the prototype. Conveniently, Kevin Scott had published an article at chatbotsmagazine.com where he reviewed Nielsen's heuristics for the applicability to chatbots. Scott recognizes that standards and best practices for user experience in chatbots are still in their emergence phase ("for now, it's a little bit Wild West"), which might be contingent upon the lack of coverage by the academic community.

Initially, he discovered that a few heuristics may be merged:
"*Visibility of System Status* and *Recognition Rather than Recall* speak to the difficulty of balancing too much information with providing enough information for the user to make informed choices. *User Control and Freedom* and *Error Prevention* both prescribe the same solutions, specifically demanding confirmation for critical steps and providing escape hatches. Finally, *match between system and real world* and *Help users recognize, diagnose and recover from errors* both speak to the necessity of consistency in language."

This leaves six heuristics that are relevant to chatbots:

Usability Heuristics BEST PRACTICES

1. **Visibility of System Status & Recognition rather than recall** — "Keep the user appraised of the system and their options at critical points, and give the user options to request additional information at any point."
2. **Match between system and real world & Help users recognize, diagnose and recover from errors** — "Know your audience. Don't switch communication styles." This is corroborated by Sörensen.
3. **User control and freedom & Error Prevention** — "Get confirmation from the user at critical points, and provide escape hatches for multi step interactions."
4. **Flexibility and efficiency of use** — Usually, this heuristic would include providing accelerators to power users, but this is not applicable to claims recording which is usually a one-time process.
5. **Consistency and standards & Aesthetic and minimalist design** — "Keep the communication style and personality / voice consistent."
6. **Help and documentation** — "Provide help within the bot."

3.3 Designing a Conversation

Interaction Strategies

Interaction strategies describe which party in a human-bot conversation takes the initiative and steers the dialog.

McTear, Callejas, and Griol (2016) present three categories:

User-directed The user always has the initiative, while the system responds to the user's queries. Problems arise when the user is invited with an overly open-ended prompt to *ask anything*, which tends to cause natural language understanding (NLU) or automatic speech recognition (ASR) errors (refer to Section 3.3 for more information).

System-directed The initiative is always on the system's side, while the user answers the questions of the system. The advantage of this approach is that it helps to *constrain user input*, as it allows to interpred users' utterances directly in the context of a question asked previously, leading to more efficient dialogs.
The disadvantage is a *lack of flexibility*, as users are restricted to behave according to the system's expectations; data must be provided in the order specified by the system.

Mixed-initiative Both user and system may take the initiative in the dialog. This is a middle ground in the tradeoff between flexibility and guidance. The system can guide the user through tasks, while the user is allowed to ask questions by himself, allowing to introduce new topics and to provide overinformative responses. However, since the user can say anything, introducing a different topic may *cause system to lose track of its agenda*.
It must be highlighted that the mixed-initiative interaction strategy requires advanced NLU capabilities, as well as the ability to maintain and monitor the dialog history and agenda effectively.

In most commercial systems, the progression through subtasks is strictly controlled, as providers of productive applications cannot afford to lose the trust of customers who end up stuck in undefined branches of the dialog. However, academic research systems may appear to provide more flexible mechanisms for handling transitions between subtasks. [MCG16] For example, the structure of an activity like creating an itinerary does not necessarily need to be predetermined. Instead, the system may have an agenda with default ordering for the traversal of the itinerary tree, but the user could change the focus or introduce a new topic. According to McTear, Callejas, and Griol, this is the case in an SDS called "CMU Communicator".

Interaction Strategy TEST

Considering that damage claims consist of a multitude of questions that easily fill entire pages in questionnaires on insurances websites, it can be assumed that the prototype has to *query for many data slots* aswell (i.e. it has a large agenda with the corresponding traversal tree).
A strictly user-directed interaction strategy would require that, for every possible data slot to be filled, another entity and corresponding training sentences would need to be added; along with callback handlers and tests.
For example, just to record the user's first and last name, all possible utterances of the kind "My name is [value]", "Call me [value]", etc. would need to be collected and added to the corpus.
This is simply not possible with the time and resources available, even if impairs the user's flexibility within this activity.
On top of that, a considerate amount of guidance is necessary to collect all this data effectively. Clients need to be told what data is required from them, having little interest in asking questions themselves when engaged in the process of recording a claim.

Therefore, we exploit the advantage of *system-directed interaction*, because it provides strong guidance and constricts the user input in response to a question to a small subset of possible intents and parameters.
Instead of the user formulating and gradually refining queries, the chatbot assumes control and leads the conversation by following a set of questions, organised inside of questionnaires.

```
System: What's your name?  // sets state
User:   [...]  // next user utterance is interpreted as a name
```

However, when engaging in non-claim-related activities such as "smalltalk" or "getting product information", other strategies should be chosen. As long as it does not change the state or otherwise affect a running claim, the user may still chime in and ask a user-initiated question and get an answer, which moves the damage claim activity slightly closer to the mixed-initiative spectrum.

Grounding and Conversational Repair

When two or more humans participate in a conversation, they engage in a collaborative action called *grounding* in order to come up with an understanding of common ground or mutual knowledge.

In the *presentation phase*, person A speaks to person B and expects that B will indicate that the utterance has been understood.
Following that is the *acceptance phase*, in which B provides an evidence of understanding (which can be as simple as a nod, smile, or verbally confirming the partner).

This process helps with detecting mesrecognized or misunderstood utterances, followed by several dialog turns with sequences of clarification requests and repairs.
The criterion before a conversation or dialog may move forward is that all parties involved agree on mutual understanding of an utterance and that everyone has a clear understanding of the concept being talked about.
McTear, Callejas, and Griol learned that this strategy is used by human agents in contact centers to verify that the peer's requirements have been correctly understood.

In human-machine conversation, the same concept is applied, although the terminology differs slightly as the *acceptance phase* consists of a series of *confirmations* or *verifications*.
As utterances parsed by the NLU component are seldomly 100% reliable, the returned utterance understanding usually has a confidence score attached to it. Consequently, if the confidence score is below a certain threshold, verifications can help to clarify that the utterance was understood correctly by the system.

The literature describes two types of confirmation strategies in CUIs:

Explicit confirmation The system generates an additional dialog turn, explicitly asking whether the data item of the previous user turn was understood correctly:

```
User:   I want to know timetables from Madrid.
System: Do you want to leave from Madrid?
User:   Yes.
```

The *disadvantage* of explicit confirmation is that "the dialog tends to be lengthy due to additional confirmation turns" [MCG16, p. 214], which causes excessive repetition if most data slots have to be confirmed.

Implicit confirmation The system includes the most uncertain data items of the user's previous turn in its next dialogic utterance.

```
User:   I want to know timetables from Madrid.
System: What time do you want to leave from Madrid?
```

It can be assumed that if the user keeps going and answers the system's follow-up question *without correction*, then the previous understanding was implicitly confirmed without additional turns.

As it is on the user to make a correction if the system misrecognized the input, this confirmation strategy often leads to utterances that are beyond the scope of the NLU (ASR) component:

```
User:   I want to know timetables from Madrid.
System: What time do you want to leave from Madrid?
User:   No, I just wanted to know about times from Madrid
        but I might be departing from somewhere else depending
        on whether I have the use of the car next Friday.
System: Sorry, I didn't quite catch that.
```

Another misrecognition situation is complete *non-understanding*, where the confidence score of the NLU component is too low to have any predictive meaning (often this is implemented as the "fallback" intent in NLU components).

In order to repair a conversation where the system is unable to understand the user's intent, the academic literature offers two solutions:

- Ask the user to *repeat the input* (when there is the chance of misinterpretation when deviating from a pre-programmed script [Eeu17])
- Ask the user to *rephrase* [MCG16]

Since the dreary “Sorry, I didn’t get that. Could you say it again?” has become a popular internet meme that spread frustration amongst people who tried the first chatbots [Eeu17], the industry has evolved to incorporate more sophisticated approaches to conversational repair from a usability standpoint.

Balancing AI and UX For instance, Phillips makes a case for balancing the AI capabilities with UX design when building chatbots. He argues that using keyboard buttons for guidance of users can have benefits if the NLU capabilities are not (yet) advanced enough to handle *any* incoming utterance. Users were nowadays still clinging onto their graphical controls and therefore visual aids and buttons were a way to provide a very straightforward UX.

In light of this, it makes sense to display a set of reply buttons that guide the user in case of non-understanding. Instead of asking to repeat or rephrase the question, the system explicitly shows what its current state *allows* the user to do.

Phillips notes, however, that using visual GUI elements such as buttons were “at the expense of text-based user responses that can be aggregated to boost AI, ML, and NLP functionality. If you’re consistently guiding users through your chatbot via response buttons and a visual UI, then you will never be able to aggregate the volume of text-based responses needed to have a meaningful impact on AI, ML, and NLP.”

The “correct” conversation repair strategy depended largely on the use case.

Prompting for Information

Prompts are messages sent by the system with the intent of requesting one or more slots of data from the user. They are categorized into *directive* and *non-directive* prompts.

Directive prompts “state explicitly what the user should say” [MCG16, p. 266], for instance, “Please tell me your first and last name”, whereas non-directive prompts “are more open-ended” [MCG16, p. 266], for example “How may I help you?”

Directive prompts are more effective from a usability standpoint, as users are confident in what they are required to say. [MCG16] However, as outlined in Chapter 4 on page 56, directive prompts may induce a less convincing and unnatural-feeling dialog. [KOY09]

The usability of non-directive prompts can be improved by including an example prompt [MCG16], for instance “You can say `record a claim`, `show products`, or `view my policy`”.

“Prompts that present *menu choices* are another design challenge” [MCG16, p. 266], as depicted in the previous section.

Considering the design of *reprompts*, McTear, Callejas, and Griol argue that “if a prompt has to be repeated [because of an invalid user response], it is preferable not to simply repeat the prompt but rather to change it depending on the circumstances. For example, if

the original prompt was unsuccessful in eliciting more than one item of information, the reprompt may be shortened (or tapered) to ask for less information" [MCG16, p. 267]:

```
System: Please tell me your home address, including postal code and city name
User:   (answers, but system fails to understand)
System: Sorry I didn't get that, please repeat your home address
```

They continue to explain another situation in which "it appears that the user does not know what to do or say, in which case an incremental prompt can be used that provides more detailed instructions:"

```
System: how many would you like?
User:   what?
System: how many shares do you want to buy? For example, one hundred
User:   a thousand
System: I'm sorry, I still didn't get that.
        Please state the number of shares you would like
        to buy or enter the number using your keypad
```

[MCG16, p. 267]

3.4 Chatbot Psychology and Affective Computing

> *"When interacting with bots our brain believes it's chatting with a human."*
>
> — Liraz Margalit, *The Psychology of Chatbots*

Bots create a false mental perception of the interaction and "encourage the user to ascribe to the bot other human-like features they do not possess.
This may seem alien, but this attribution of human characteristics to animals, events or even objects is a natural tendency known as *anthropomorphism*." [Mar16]

In other words, our instincts inevitably ascribe human motivations, beliefs, and feelings to inanimate things that behave in intelligent ways, which might happen in a conscious or subconscious manner. This connection effectively works in both directions, which has repercussions in the way that advanced conversational systems can *detect, adapt to, influence, and exploit the user's affective state*. [Sko10]

"Scientists have shown that mechanisms to understand and exhibit emotion and personality in artificial agents (*affective computing*) are essential for human-computer interaction. [...] Affect allows us to fully understand each other, be socially competent, and show that we care." [MCG16]

Nowadays, "there is a very active research community working on affective computing, with several international conferences and journals. This work has demonstrated the many

benefits that emotion can bring to artificial systems, including *increased usability, efficiency, trust, rapport, and improved communication*." [MCG16, p. 323]
Common interaction goals and strategies for conversational agents are e.g. "*making the interaction more fluid and satisfactory*, *fostering the acceptability, perceived social competence and believability of the agent*, *and keeping the user engaged during the interaction*" [MCG16, p. 319], whereas affective computing is also used in commercial applications such as *sentiment analysis* and *opinion mining* for marketing and branding. [MCG16]

Practical Applications in Chatbots

Sentiment as a Trigger for Human Handover According to Dan Norman, we tend to "project human emotions and beliefs into the inanimate computer when interacting with computer interfaces". This means that when the interaction is smooth and enjoyable we associate pleasure and satisfaction directly with the system, but also blame the machine with anger and contempt when things do not work out as we wish.
The latter may induce dissatisfaction, hinder the creativity to explore other ways to solve the problem at hand, or even cause users to stop the conversation altogether.

As noted in the Motivation chapter, one important application of affective computing is therefore handing the conversation over to a (specially trained) human agent once the user's observed distress levels reach a certain threshold, which is detectable by employing modern sentiment analysis techniques.

This can be an alternative or supplementary strategy to the explicit solicitation of human handover via key phrases such as "I want to talk to a human".

Conveying a Personality The design of a fitting personality that resonates with people, or even reacts dynamically to changing user affect, is fundamental. Style, tone, and attitude need to be catered toward the context and depend on the objectives of the system.

"As the personality of a bot has a great influence on affect, one may ask which personality should be conveyed, and what methods can be used to render it? [...] Several studies have shown that *people like personalities that are similar to their own*, a phenomenon known as the *similarity-attraction principle*." [MCG16, p. 316] However, in order to provide a perfect resemblence, very nuanced algorithms are necessary to catch and reproduce every aspect of the user's character such as emotion, writing style (casing preferences, length of sentences), attitude (extraverted, introverted), or adapting to the user's experience with conversational interfaces.

According to the study of Sörensen (details later in Chapter 4 on page 57), a chatbot used within insurance should be personal and human-like, probably since insurance is such a delicate and important area so users feel more comfortable and secure if it feels like they are talking to a human. "However, the chatbot must be to the point and give precise answers." [Sör17, p. 11]

In the study, some participants noted that it was "always important that the chatbot presents itself as a chatbot in order to set up the conditions for the conversation. If the chatbot omits this information the user could get an idea of speaking to a human operator and thereby have the expectations of being able to write in anyway they like and the operator would be able to pick up on the intent and give a correct answer." [Sör17, p. 9]

Sörensen's second version of the same chatbot did not introduce itself fully, and the study participants "perceived difficulties regarding this: 'it didn't really respond to my question like a normal person. She didn't understand if I said something that was a bit off topic'." [Sör17, p. 10]

Personality TEST

The prototype should adhere to the following personality traits:

- Be friendly
- Be humorous
- Adjust dynamically to the correct German (in)formal address ("du" vs. "Sie")
- Make it obvious that it is a chatbot
- Be to the point and utter precise prompts when in the context of recording a claim
- Give congratulations and advises, as this helps make the bot more "easy going and fun" and gives it a more human-like characteristic [Sör17]

It is not necessary to do automatic adjustments to the user's personality, as that is out of scope for a prototype.

The Power of Small Talk for Empathic Engagement Humorous stimuli enhance the positive involvement of users in conversation and increase their intentions to continue using the system. It was also shown that "when users lose interest in the system this has a stronger effect than other negative indicators such as user frustration". [MCG16, p. 322]

"In application domains where the agent is supposed to maintain long-term interactions with their users, a sustained interaction with an agent that always behaves in the same way would be likely to decrease user satisfaction over time." [MCG16, p. 322]

On top of that, agents who deviate from dialog to social talk have shown to be perceived as more trustful and entertaining [PG17] and in particular, humor may be of a great help in achieving these traits [MCG16].

Small Talk USER EXPERIENCE

Since the quality attributes 3.2.2 include the goals of entertaining the user and providing an enjoyable experience, the prototype is coded to be able to deviate to small talk in applicable situations.
However, it urges the user back to the insurance domain by uttering interesting insurance statistics about smartphone damages, so that the user is inclined to record his or her case of damage with the bot the next time.

3.4.1 Managing User Expectations

As Luger and Sellen showed, user expectations in GPAs are dramatically inflated in most aspects of these systems, from known machine intelligence, through system capabilities and goals, to natural language understanding.

They criticized these systems with regards to overpromising system intelligence through the use of familiar 'easter eggs' and humorous trigger responses, belying the true system capabilities. Another point are insufficient measures to counteract the unrealistic expectations set by novice users who anthropomorphise these systems, which they say "should be scaffolded through more *considered revelation* of system intelligence through design." [LS16, p. 9]

They conclude that "whilst CAs offer the promise of an engaging and natural user interface, much design and interaction work is required before this potential is realised." [LS16, p. 10]

However, Luger and Sellen specifically targeted general personal assistants and not specialized digital assistants, meaning that chatbots with more narrow use-cases might generate different expectations.

Sörensen (Chapter 4 on the previous page), on the other hand, studied chatbots in the exact same scope of SDAs as this thesis is dealing with (chatbots in the insurance sector).

She discovered that the phrasing of the welcome message affects how the users formulate themselves in the rest of the conversation, as it gives the user different expectations on the chatbot's abilities.
For instance, the first chatbot in her study formulated its welcome message as an open-ended, non-directive prompt: "Hi, what do you need help with?"
Her participants perceived this system as "giving more opportunities", that it had to be "really clever when the question is *this* open", and that it felt good to reply to an open question by "proper" means (natural language instead of keywords).

Conversely, her second bot opened every conversation with the words "I can help you with all sorts of questions, or just type a word and I will tell you more about it", which was "perceived as more restricted regarding the input format. By informing the user of the expected input format, as well as guiding information of what the chatbot could handle, [a

participant] expressed that it ['felt easier'] and gave comfort in the conversation" [Sör17, p. 9]. These findings correlate with the Section on *Prompting for Information*.

Therefore, "in order for the user to interact effectively with the chatbot, the input language has to be adapted to the chatbots needs, the onboarding is part of this adaption when the user learns the pragmatics of the chatbot." [Sör17, p. 13]

Apart from that, it was also discovered that users are quick in adapting to the system's needs and capabilities by modifying their turn design, i.e. selecting words that they think the bot would understand, judging by their mental model of the system.

When learning to use their CA, all of Sörensen's participants "described making use of a particular economy of language. Dropping words other than 'keywords', removing colloquial or complex words, reducing the number of words used, using more specific terms, altering enunciation, speaking more slowly/clearly and changing accent were the most commonly described tactics." [Sör17, p. 13]

As this learning process takes place mainly in the beginning of the interaction, we can conclude that the design of the onboarding process is of vital importance for setting expectations.

CHAPTER 4

Related Work

This chapter reviews existing literature on conversational interfaces and presents relevant studies with interesting results and their methodologies.

Insurance

Chatbots for delivering Notice and Choice "PriBots: Conversational Privacy with Chatbots" explains how the traditional workflows of informing users "about how websites, devices, apps, or service providers handle their data" (*notice and choice*) have so far "failed to protect users' privacy". Such privacy notices manifested "in several forms, from the (typically lengthy) privacy policies to the (often ambiguous) app permissions". As users were "faced with notice complexity, lack of choices, and notice fatigue" however, they tended to "ignore these notices with time and opt to use the services directly. [With] state-of-the-art approches to improving this model [...] seeing limited spread/usage", Harkous et al. propose to use "Conversational Privacy Bots (PriBots) [as] a new way of delivering notice and choice by creating a two-way dialogue between the user and a computer agent."

In conclusion, they introduce the idea of a question-answering bot which reacts to users' questions about a company's privacy policy.

We can easily see an opportunity for chatbots in the insurance sector here (although they do not mention this use case explicitly), as this question-answering mechanic could be integrated to deliver the policies of insurance products.

Literature on Conversational Interfaces

The Conversational Interface is an in-depth examination and amongst the standard literature to learn about CUIs. It covers most, if not all, aspects of SDS in detail and they claim that there was "no comparable book that [brought] together information on conversational interfaces comprehensively and in a readable style [before]" [MCG16, p. 2].

Although the book is focused on actual voice-driven interfaces and some topics are really only applicable to those, most of the ideas also help with the examination of text-driven CUIs.

Apart from the mandatory technical examination of chatbot components and technologies (dialog manager (DM), NLU or SLU, natural language generation (NLG), text-to-speech synthesis (TTS), etc.), they also cover the range of conversational interfaces like wearables, virtual agents, and robots, and it is a particularly interesting read for studying the intricacies of human-to-human and human-to-bot dialog (emotion, affect, personality, the structure of conversation).

Many of these concepts were borrowed and briefly outlined in the User Interfaces and Experience and Prototype Implementation sections, as was "the cycle of increasing returns" in Motivation.

In his senior thesis [Cah17], Jack Cahn reviews existing chatbot literature for "design choices, architecture, and algorithms". He describes "chatbot function and history", the "methods used to evaluate chatbots", components of a typical chatbot architecture, and performs a case study on the inner workings of the popular IBM Watson chatbot, a "Question-Answer chatbot" which "gained popular recognition for its Jeopardy! win". On top of that, the items and ideas in the History subsection in this thesis were inspired by the information Cahn collected. The reader be especially referred to his paper if interested in the steps involved between receiving a plain text message, natural language processing thereof, and response generation based on rules, statistical models or knowledge bases. That is because he did a good job at summarizing these processes and this thesis blatantly disregards their exact details for the most part. We focus more on the engineering of industrial-grade chatbots based off existing tools, rather than the theoretical backgrounds behind them.

User Experience

Convincing Chatbot-Human Dialog Kirakowski, O'Donnell, and Yiu conducted an experiment where fourteen participants interacted with an "Eliza-style" Chatbot for three minutes each and then highlighted subsections in the transcript of their conversation which seemed either particularly unnatural, or very convincing. The reasoning behind their study was that human-like, intelligent behavior in a powerful enough Chatbot "could offer a hugely attractive form of Human Computer Interaction in that computer use could be mediated by an agent that behaves as though it can understand instructions that are typed

in by human users and respond in kind with natural-seeming utterances" [KOY09, p. 1]. After analyzing the annotated experiment content, they produced a list of themes that might serve as guidelines for the engineering of a convincing dialog:

Unconvincing characteristics:

- "Fails to maintain a theme once initiated. In that, once a theme emerged in the dialogue, the chatbot failed to produce statements relevant to that theme in the following subsection of the dialogue."
- Stiff, formal or unusual treatment of language
- "Failure to respond to a specific question. Users would ask for a specific piece of information, such as asking the chatbot what its favourite film might be, and receiving no answer."
- "Time delay. A fairly cosmetic fault, users felt that the chatbot responded too quickly to a detailed question or too slowly to a courtesy."

For **convincing characteristics**, they name:

- "Greetings. Several participants identified the greeting as a human-seeming characteristic."
- Maintains a theme in consecutive utterances.
- "Damage control. When the chatbot produced a breakdown in communication [...] and then produced a statement that seemed to apologise for the breakdown or seemed to redirect the conversation in a more fruitful direction, users found this a cinvincingly human trait."
- When the chatbot offered a cue for further discussion, such as "What do you want to talk about?" or a range of topics.
- "Language style. Users found conversational or colloquial English to be convincing."
- Personality. Users tend to give the bot a name, independent of it actually having one.

Managing Expectations during the Onbarding Process In [Sör17], Ingrid Sörensen presents a study exposing eight participants to functionally similar chatbots, where one was *human-like* and the other had a more *robotic character*.

These bots were situated in an insurance context with the purpose of signing up for an insurance, cancelling a re-ordered insurance, getting a recommendation, and acting on a push notification. The participants were asked to interact with the two bots in order to complete these tasks.

She evaluated the requirements and expectations of a chatbot of novice users by interviewing them about their thoughts and experiences from using it. While learning to use the chatbots, all participants were observed to *initially use natural language* as though the chatbot were a person. In those cases where the intent of the input was not understood by

the chatbot, usually due to misinterpretations or lack of ability to find a correlating answer on the system side, the upcoming series of *sentences became shorter for each iteration*. Until, in some cases, *only short-typed and essential keywords remained*.

The ramifications and design advices of the results in her study were discussed in more detail in Section 3.4 on page 50.

CHAPTER 5

Challenges

This chapter summarizes the challenges that had to be overcome during engineering of the prototype while taking into consideration some more technical requirements.

Natural Language Understanding

For one, natural language processing is, despite the many improvements in recent years, still not at a stage where it understands any nuanced sentence you throw at it. In addition to that, machine learning algorithms that perform NLU need big amounts of data, which is especially hard to obtain in the insurance sector because of naturally-grown IT infrastructures and the fact that oftentimes there are no textual records of conversations available.

For one, it has taken “decades for scientists, researchers, and practitioners around the world to create algorithms and systems that would allow us to communicate with machines” [MCG16, p. 1].
As these algorithms are very complex [MCG16], it might not be wise to build one up from scratch if no special features are required that differ from available standard solutions. Whereas the internal models of standard solutions are simple to train by adding sentences to a “user says” bank, this process requires large amounts of data that which to be extracted from recordings of conversations.

Insurance companies naturally treat these sensitive user recordings very restrictive however, meaning that we would not get our hands on any data to train the models on. A single person can only think of so many sentences until it exceeds his imagination of possible user inputs.
Therefore, a challenge was the iterative refinement with many test persons in the prototype

that was necessary to get it to a usable state where it understood most of the domain logic people threw at it.

Keeping Track of Context INTEROPERABILITY

Dialogflow itself supports the concept of "input" and "output contexts", which is a way of keeping track of a specific input parameter and handing it over to another intent.
However, the service offers no way to remotely modify these contexts, and arguing that *the prototype should not be locked in to a specific platform*, this capability of the Dialogflow service was not used (as others did no support this feature in the same way). Instead, keeping track of conversation context had to happen in the backend of the bot, and it was decided that a finite state machine should handle the transitions.

Platform-Agnostic Chatbot — Abstracting IO PORTABILITY

Being able to hand a conversation over between different devices and even platforms adds great convenience for the user. As indicated in 2.2.2, the Telegram and Facebook Messengers would be used.
The challenge was to abstract their capabilities well enough, so that adding more platforms to the backend would be as easy as adding a single class without additional changes.
To demonstrate switching of chatbot platforms at runtime, a strongly decoupled integration component had to be built which combines the events from Facebook Messenger, Telegram, or any other messaging platform (SMS was attempted) to a single, unified type of entity ("Event", "Update", "Text Message", respectively, to a single `MyUpdate` containing all the necessary information). Another component also had to allow sending messages, media, and typing indicators in an abstract way.

Simple and lightweight conversational copywriting EXTENSIBILITY

On top of the aforementioned abstraction, a goal was set out to make the work of a "conversational copywriter" (designer of conversational content) working with the system as simple as possible by decoupling authoring from development. In essence, designing the conversation should involve programming to some extent in order to remain flexible, but abstract the underlying architecture away well enough so that a copywriter would be able to focus on the actual content without being distracted by complex data structures or unimportant boilerplate code.
This involved modularizing the core of the application and dedicating the conversational content its own module. Easily adding new states and transitions to the state machine, setting/reading/removing flags or arbitrary information regarding the context of the current user, as well as a sentence composer that

would allow access to the sentence bank by simple means (`composer.say(intent)`, `composer.ask(question_or_intent)`, `composer.send_media(media_id)`) were a subset of convenience features that should be supported.

Response Generation in German language

It was clear before the actual implementation that a holistic response generation (RG) component would not be feasible and way out of scope for this thesis, as linguists already struggle with the task in the English language, and German resides at an even more complicated level. Nonetheless, an initial attempt was made with the idea of providing a canned sentence bank that only contained *stripped-down* snippets, which would then be combined to form an actual sentence. In order to achieve linguistic correctness (i.e. correct grammar) with this approach, it was attempted to build a class that would take two or more snippets as arguments and then use the correct, context sensitive German personal pronoun form such as 1st person singular accusative ("mich").
This experiment failed severely, and the convoluted code was left in the project as a reminder for people not to try this at home.

Eventually, it was settled on a method based on recursive canned text templates with variables that would be inserted at runtime.

Recording Claim Information — FUNCTIONAL REQUIREMENT

The core task of the prototype is recording a damage claim. The necessary and optional data slots for a full claim had to be researched and proper means of requesting them thought of. This included creating a wrapper around generic `Questions` inside of `Questionnaires` that would be extensible just by adding more of said `Questions`.

User Stories — USE CASES

The main goals of users interacting with the prototype are as follows:

Capture a Smartphone Damage Claim — "I want my insurance to compensate for the `[damage_type]` to my phone. I need a way to tell them my insurance ID and enter the cause of loss. They might need other information from me, but I'm not sure what exactly."

Get inspired by the Bot — "I heard that chatbots might improve the customer service of my company. I want to test the bot and see what's possible."

CHAPTER 6

Prototype Implementation

This chapter summarizes the design decisions that went into engineering of the eventual chatbot solution by following the previously collected requirements. In addition, it covers the academic backgrounds and industrial best practices behind the individual components that make up the system, also explaining the considerations for each choice when there are alternatives.

For the system was designed using a top-down approach, the following sections appear in the actual chronological order of tangency.

6.1 Core Architecture

The system was built to be strictly separated into a **reusable core module** and the ad-hoc **domain logic and conversation design**. This was done to ensure that the industry partners would later be able to build their own logic upon the core module seamlessly by working with it as a software library. Python was the programming language of choice throughout the project as it comes with many useful features to support rapid prototyping and a vast number of readily available third-party modules in the Python Package Index (PIP) to further increase velocity of development.

Granted that the following description may be slightly overwhelming in its textual form, two sequence diagrams are provided in Section A.2 on page 97 and Section A.2 on page 98 to illustrate the architectural procedure, and it is advisable to examine them first before diving into the text.

In order to achieve true platform agnosticity, a `BotAPIClient` class serves as the abstract base for all bot API client implementations such as a `TelegramClient`, `FacebookClient`, etc. Apart from abstract methods for registering callbacks with specific filters that are triggered on incoming webhook requests, it defines two important methods that every concrete client implementation needs to override:

- The method `unify_update` is responsible for transforming incoming events from the respective bot platform into a common `Update` type which the whole backend works with. That way, incoming messages have a uniform class signature and can safely be abstracted by an object relational mapper (ORM) and stored in a PostgreSQL database, while still containing all relevant information about an incoming message such as origin, date, text, media, and additional payload. This process involves additional helpers to transcribe voice messages to text form and download media files.
- The second method, `perform_actions`, allows to execute a chain of `ChatAction` objects that each represent an atomic activity in a chat with a specific peer such as sending a text or media message or displaying a typing indicator. `ChatActions` also determine the duration of delays between individual messages, as well as the reply markup to be attached, for example an array of reply buttons when a question allows to restrict the user input to distinct choices. These objects are effectively blueprints of a response to be sent as defined by a `PlanningAgent`, which enables to reference back to what has been said in a conversation.

Furthermore, a `ContextManager` is used to map a `Context` object to every individual user, which holds the current dialog state, a random access key-value store, as well as the incoming and outgoing utterances in their respective object representations (`Update` and `ChatAction`). As this information is persisted in a Redis database, no dialog-related state is lost between restarts of the bot.

On startup of the system, the designated chatbot client implementations are registered in the `DialogManager`, which acts as a central control unit. The dialog manager instructs each of the clients to register event handlers in order to be notified about incoming updates and be able to initiate processing thereof.

Upon receiving an update from one of the clients, the dialog manager uses its abstract `NLUEngine` member for intent determination in order to enrich the update with a `MessageUnderstanding`, so that the incoming message can consequently be referred to by its intent, parameters, and entities. Following that, the context manager is instructed to store the update with regard to the current user.

For the purpose of generating a response for the user, the dialog manager requires an instance of a class implementing an `IPlanningAgent` interface, which consists of a single function: `build_next_actions(context)`. As its name implies, this function takes the context of a user and creates a chain of chat actions by using any strategy the developer wishes to deploy. In this case, a hybrid agent-based dialog management strategy was used to combine the benefits of finite state machines with frame-based data slots. Please refer to

(a) Facebook Messenger **(b)** Telegram

Figure 6.1: The system is mirrored on both the Facebook and Telegram Messengers

section 6.2.2 for details on the applied strategy, and to section 6.2.3 for an explanation of how the response generation component works.

Once the planning agent has generated a response, the resulting chat actions are collected and optionally stored in a `ConversationRecorder` instance for logging and replaying purposes, which uses YAML files to store the recording of a conversation.

Finally, the dialog manager hands the collected chat actions over as parameters to the `perform_actions` method on the original client instance, which then makes the appropriate API calls to ensure that the laid-out plan gets executed.

It is interesting to note that both incoming *and outgoing* utterances are internally treated as intents, meaning that any activity within the system is an abstract representation of a *plan* to be executed, independent of the actual words or sentences contained in it. This opens up possibilities for retrospective planning in the way that previous updates and chat actions may be queried for whether a specific intent had already yielded a response before, which may influence the reasoning behind a choice. This effectively implements the basis for a future information state update (ISU) model (6.2.2) and should also allow for a

straightforward internationalization of the conversational content (since intents are held in english and independent of the corpus language).

Note that the actual implementations of the Telegram and Facebook Messenger clients are intentionally left out from this explanation in order to highlight the fact that these classes are transient and it is possible to add virtually *any* messenger platform. To provide some reference however, the *python-telegram-bot*[1] and *fbmq*[2] Python libraries were built upon to realize the already available clients.

Another point to consider is that although the term "manager" is usually considered a bad practice for the naming of software components in the way of being overused and not conveying much meaning, the `DialogManager`, `ContextManager`, etc. are terms commonly used in the scientific space and offer enough symbolic value to justify their naming.

6.2 Individual Components

In order to get a deeper understanding of the system, this section analyzes the individual subcomponents in greater detail.

6.2.1 Intent Determination

The purpose of a *dialog act recognition* component is to determine the *function of an utterance* in a dialog e.g. whether it is a question, suggestion, or an offer. In academic research, the goal of dialog act recognition is to find a taxonomy of dialog acts (DAs) that can classify the *whole range of possible functions*. [MCG16]

In contrast, commercial applications typically employ an *intent determination* component which uses a narrower range of utterance functions, such as asking a question, issuing a command, or providing information. Combined with the application's domain and tasks, this dialog function constitutes the *intent* of a message (e.g. book a flight, request weather forecast, set an alarm). [MCG16]

Following this first stage of classification, appropriate slot-value pairs (a.k.a. *frames* or *entity-parameter pairs*) can be extracted. For example, when an utterance "My phone fell on the ground" to the chatbot is classified with the intent `phone_broken`, then the slot `damage_type` (entity) might contain `DISPLAY_DAMAGE` (value) with a certain confidence.

A weather bot, as a second example, would expect the slots `location` and `timeframe` to be filled when it detects the intent `weather_forecast`.

Wu et al. found that this two-stage classification helps to constrain the semantic analysis of an utterance, as the relevant set of parameters is different for each identifiable intent.

[1] https://github.com/python-telegram-bot/python-telegram-bot
[2] https://github.com/conbus/fbmq

Service Integration There are a vast number of NLU platforms, including Dialogflow (formerly `api.ai`)[3], Wit.ai[4], Microsoft LUIS[5], and IBM Watson[6], that enable developers to specify and train the intent determination of a conversational interface by defining the intents and entities associated with the utterances that users are likely to say.

In the prototype, the intent determination mechanism is considered a *configurable black box*, accepting a sentence in natural language and yielding its intent, entities and corresponding parameters. This is done to allow focus on more high-level concepts such as user experience, considering that natural language processing is a sizeable discipline which is going to keep computational linguists busy for years or decades to come.

Since the NLU service sits right at the core of the application, a great effort was undertaken to compare features and quality properties, in order to achieve the best possible intent identification for minimal effort.

Table 6.1 shows an evaluation of the aforementioned NLU services, based on the following qualifiers:

Python bindings There exists a library to interact with the service in the Python programming language

German language The service natively supports German

Free of charge The service should be utterly free, or charge rather little for a number of API calls

No credit card required (Not really a knock-out criterion, but being forced to provide credit card details discourages an easy setup)

Remember session state The service has a way to distinguish subsequent requests belonging to the same user (sessions with IDs)

Service bound The platform runs in the cloud and is not self-hosted

Simple training The service should allow simple yet efficient training of its internal model by adding “user says” sentences.

[3] https://dialogflow.com/
[4] https://wit.ai/
[5] https://www.luis.ai/
[6] https://www.ibm.com/watson/services/natural-language-understanding/

Table 6.1: Comparison of Microsoft's LUIS, Google's Dialogflow, Facebook's wit.ai, and IBM's Watson

	LUIS	**Dialogflow**	**Wit.ai**	**Watson**
Python bindings	no	yes	yes	yes
German language	yes	yes	in Beta	yes
Free service	no	yes	yes	no
No credit card	no	yes	yes	yes
Remember state	yes	yes	yes	yes
Service bound	yes	yes	yes	yes
Simple training	with effort	yes	yes	yes

Apart from the fulfillment of all criteria, there were two more points that tipped the scale and eventually solidified Dialogflow as the service of choice:

- Convenient importing of prebuilt agents in many languages for smalltalk, dates, currency conversion, etc. (with intents and corresponding training sentences each)
- Up and download of agents for backups and offline batch modifications

For an exhaustive review of the named and other choices, the reader be referred to *25 Chatbot Platforms: A Comparative Table* [Dav17].

In the prototype, a Dialogflow client wrapper enriches incoming `Updates` from the bot APIs with a `MessageUnderstanding` object that contains information about the determined intent and its parameters (data slots).
Configuration and training of the Dialogflow agent happens in a mature web interface which allows adding "user says" training sentences and to train the agent's model on misunderstood utterances by annotating them with the correct intents.

6.2.2 Dialog Management

Central in the development of a chatbot is designing a proper *dialog management strategy*. A dialog manager (DM) "defines the system's conversational behaviors in response to user utterances and the current state of the system." [MCG16]

In the industry, the DM often consists of a handcrafted set of rules and heuristics, which are tightly coupled to the application domain [MCG16] and improved iteratively by developers and conversation designers to cover more branches and intricate details of the dialog.

One of the obvious problems with handcrafted approaches to DM is that it is challenging to anticipate every possible user input and design appropriate strategies to handle them.
This design process is error-prone and it requires "considerable time and effort" [MCG16, p. 217] to iteratively refine and tune the dialog strategies. However, it offers very fast time to market (TTM), especially in the case when there are very few or no recordings of

conversations available which renders more sophisticated heuristical approaches to dialog management difficult to impossible.

As opposed to the rule-oriented strategies, data-oriented architectures work by using machine learning algorithms that are trained with samples of dialogs in order to reproduce the interactions that are observed in the training data. [Spi05]
These statistical or heuristical approaches to DM can be classified into three main categories:
Dialog modeling based on *reinforcement learning (RL)*, *corpus-based* statistical dialog management, and *example-based* dialog management (simply extracting rules from data instead of manually coding them). [MCG16; Spi05]
Spierling highlights "classical information retrieval algorithms, neural networks (NNs), Hidden-Markov Models (HMM), and Partially Observable Markov Decision Processes (POMDP) [as] technologies on which these systems can be based." [Spi05]

In the case of this thesis, unfortunately there was no way to obtain any conversation recordings of insurances with their customers (mostly because of privacy concerns), so all the strategies and the corpora (sentences for training of machine learning algorithms) for intent classification needed to be forged by iterative refinement of the intent determination component.
Conversely, even if such corpora had actually been available, "the size of currently available dialog corpora [is usually] too small to sufficiently explore the vast space of possible dialog states and strategies", McTear, Callejas, and Griol note in *The Conversational Interface*. They also point out that collecting a corpus with real users and annotating it requires, again, "considerable time and effort."

Following are **common strategies for rule-based dialog management**.

Finite-state-based One of the simpler dialog management strategies is finite-state-based DM, where the interaction model is based on a finite state automaton (FSA) with handcrafted rules. "This approach is usually confined to highly structured tasks in which *system-directed initiative* is used and the user's input is restricted to utterances within the scope of the ASR and NLU components." [MCG16]

Finite-state based dialog managers have been deployed in many applications due to their simplicity and extensibility. [MCG16]

Frame-based Frame-based DM strategies follow no predefined dialog path, but instead allow to gather pieces of information in a frame structure and no specific order. This is done by adding an additional entity-value slot for every piece of information to be collected and annotating the intents in which they might occur. Frames also allow capturing of several data slots at once, as more than one slot can be filled per dialog turn.

"Using frames, it is possible to specify the whole topic of a dialog" [MCG16], which requires a very fine-grained analysis of possible user inputs. The main idea is that the mental state of

humans may change during an interaction, and the frame-based strategy allows to express "whatever comes to mind". As it enables rather unrestricted dialogs, this strategy should be chosen in a context where the interaction strategy is *user-directed* and not much guidance is required from the system.

Information State Update This strategy, also known as *Information State Theory*, is a theoretically motivated attempt at characterizing the dynamics of dialogs.
The strategy represents the information known at a given state in a dialog and updates the internal model each time a participant performs a *dialog move*, such as asking, answering or accumulating information.
The *information state* (or *conversation store*, *discourse context*, *mental state*) may include information about the mental states of the participants (beliefs, desires, intentions, obligations, and commitments) and about the dialog (utterances, generated dialog moves, information that is shared between participants) in abstract representations. [TL03]

Reasoning about the information state, the strategy computes the so-called *update rules*, which are applicability conditions such as `is_valid_move` and effects/consequences of a dialog move. In other words, the main tasks for the dialog manager are to update the information state based on the observed user actions and to select the next system action as specified in the update rules. [TL03]

The ISU dialog management requires very advanced analytical capabilities in many aspects of the system and may yield very good results in long-lasting relationships with users, but it is usually not employed in task-oriented commercial chatbots due to its complexity.

Agent-based A modular agent-based approach to DM makes it possible to combine the benefits of the previous dialog control models. [COHM05]

A Planning Agent for Dialog Control

In the prototype, an agent-based strategy was chosen in order to combine the capabilities of the frame-based entities and parameters in Dialogflow with a custom dialog controller based on predefined rules in a finite state machine.
This FSA allows to define rules that trigger *handlers* and *state transitions* when a specific intent or entity-parameter combination is encountered. That way, both intent and frame processing happen in the same logically encapsulated unit, enabling better maintainability and extensibility.

The rules are instances of a set of `*Handler` classes such as an `IntentHandler` for the aforementioned intent and parameter matching, supplemented by handlers for various purposes and functionalities. These include an `AffirmationHandler` which consolidates different intents that all express a confirmation along the lines of "yes", "okay", "good" and "correct", as well as a `NegationHandler`, a `MediaHandler` (images, videos, stickers, etc.), and an `EmojiSentimentHandler` (to analyze positive, neutral, or negative sentiment of

a message with emojis). As they all inherit from an abstract `BaseHandler`, they must each implement their own `matches(message_understanding)` method which takes a `MessageUnderstanding` object and returns a boolean value determining whether this rule should apply to an incoming message.

Rules (handlers) are added to *one of three collections* in a `Router`, which is effectively a blueprint for the state machine:

1. *Stateless* handlers are always checked first and independent of the current state. For example, a `RegexHandler` rule determines whether the formality of the address towards the user should be changed (German differentiates the informal "du" and the formal "Sie")
2. *Dialog States* is a dictionary that maps each possible state to a list of handlers that are applicable in that state. For instance, when the user has given an answer and the system asks for *explicit confirmation* (Section 3.3 on page 48) of its understanding in a state `USER_CONFIRMING_ANSWER`, then an `AffirmationHandler` and a `NegationHandler` capture "yes" and "no" answers, and a parameterless `IntentHandler` interprets any other utterance as a correction of the understanding by the user (see Listing 6.3 on page 78). The states in the system may be any hashable Python object; tuples of strings were used heavily.
3. *Fallback* handlers are checked if none of the applicable state handlers have yielded a match for an incoming `MessageUnderstanding`. These `fallbacks` include static, predefined responses with lowest priority (e.g. smalltalk), as well as handlers to repair the conversation by bringing the user back on track or changing the topic.

When the planning agent detects that a rule applies to an incoming message, it executes the callback defined by the handler. In order to trigger a state change, the callback function simply returns the new state with a `lifetime` attached to it.

At first, the system had only allowed a single state to be declared at the same time in the router. However, this had quickly proven to be insufficient as users are likely to want to respond or refer not only to the most recent message, but also to previous ones in the chat. With only a single contemporaneous state, the user's next utterance had always been interpreted in exactly that state and if none had matched, the router had jumped directly to the fallbacks. In order to make this model resilient, every state would have needed to incorporate every utterance that the user was likely to say in that context. Adding all these transitions would have been a daunting task as the dialog model grew, so there had to be a better solution to add state handlers that would allow layering transitions on top of each other.

As the initial concept of the system was to have a single monolithic state machine containing all the states and transitions, one idea was to split it up into an arbitrary number of routers that would each handle one branch of the conversation.

The second envisioned solution was to allow multiple simultaneous states whose transitions would all be valid at the same time. This deemed superior to the first idea, as it allowed

to introduce the concept of *state lifetimes* to the system: new states returned by callbacks may have a lifetime that determines the number of dialog moves this state is valid for. On receiving a new message, the planning agent decreases the lifetimes of all current dialog states by one, except for the case of utter non-understanding ("fallback" intent). If a state has exceeded its lifetime, it is removed from the priority queue of current dialog states. As there should always be a state to fall back on, states can have an `infinite` lifetime which is equivalent to having no lifetime returned from the callback. When an `infinite` lifetime state is appended to the queue, the dialog states are cleared beforehand as no other states will be reachable in this case.

This state handling resembles a push down automaton (PDA) with prioritized state transitions, however using a queue instead of a stack to be able to modify elements in the collection of states. The algorithm is actually not hard to grasp once it is encountered "in action", as it replicates our natural intuition of how a dialog works quite well. For example (as seen in Figure 6.2 on the preceding page), in the beginning of a conversation the bot starts out with a single state `SMALLTALK` in the queue. After greeting the user, the bot may ask "how are you" and pushe the tuple state `('asking', 'how_are_you')` on the queue with a lifetime of 2 dialog turns. The user may now respond with an unrelated question "are you a human" to which the bot answers "no, I am a chatbot" and decreases the lifetime of the first state to 1. If the user now decides to answer the initial question with "I'm fine by the way, thank you", the system still knows which utterance the user is referring to and may answer "glad to hear that" accordingly. In the same way, states will be stacked on top the existing ones, causing state handlers to always match the most recently added state before the other ones in the prioritized queue. This enables a more nuanced conversation where it "feels like the bot remembers things".

Figure 6.2: Remembering states

It is fitting that the dialog states of a specific user are stored and persisted in his or her dedicated *Context*, as layered dialog states actually comprise a large part of a conversation's context.

In addition, it would have been possible to interpret *actual* replies in the Telegram interface as responses to specific messages of the bot, although one cannot rely on the user to always make use of this feature reliably and Facebook Messenger does not support the concept of replies yet anyway, which is why this feature was not implemented.

Finally, the deepest step in dialog control are the `handler callbacks`, which comprise reusable definitions of logic and responses such as `ask_to_start` or `ask_next_question` that depend on the `Context`. The Response Generation section examines these callbacks further.

6.2.3 Response Generation

A Response Generation (also referred to as *content determination* or *content realization*) component decides what words to use for a response laid out by the dialog manager.

As explained in Section 3.1.1 on page 40, the automatic generation of responses is a hard task and there are no tools available for the German language as of yet.
Apart from statistical approaches to response generation however, many dialog systems adopt a simple approach in which the system outputs predetermined responses with the help of *canned text* or *templates with variables* to be inserted at runtime. While this approach has a problem with a lack of flexibility since designers have to anticipate all the different contexts possibly occuring in a dialog, it works well in fairly restricted interactions. [MCG16]

As McTear, Callejas, and Griol recommend, "canned text can be used in interactions where the system has to elicit a predetermined set of values from the user", which exactly fits the requirements of the prototype.

Composing Sentences When the planning agent has chosen a callback to execute, it passes on a `ResponseComposer` object and the `Context` of the currently interacting user, producing the following *generic callback function* signature:
`def callback(r: ResponseComposer, c: Context) -> object.`

The response composer is a tool for more convenient conversational copywriting, as it exposes a straightforward API for conversation modeling with methods such as
`say(intent)`,
`ask(intent_or_question, choices)`,
`send_media(media_id)`,
`give_example(question)`,
`implicitly_ground(question, user_answer)`,
and `ask_to_confirm(question, user_answer)`,
which effectively implement the guidelines for conversation design of Section 3.3 on page 45.

The sentence bank consists of an arbitrary number of YAML Ain't a Markup Language (YAML) configuration files and maps all available *outgoing* `intents` to one or more distinct formulations with the same meaning, eagerly loaded at the startup. The individual phrasing choices may have applicability conditions attached that are validated at runtime to determine which of the available sentences should be used. Generating a response for the outgoing intent `ok_thank_you` for instance, it would not be wise to pick the wording "okey-dokey, thanks!" if the user has recently expressed the intent `user.sad` or when the degree of formality in the conversation with this user is supposed to be high. In this case, the applicability condition could be realized as
`not user_recent('user.sad') and user.informal.`

For handling of the case when multiple templates have valid conditions, any implementation of an abstract base class `TemplateSelector` may be injected (defaulting to a `RandomTemplateSelector`) to select from the choices, whereas a `LeastRecentlyUsedSelector` was deployed to ensure a deterministic sentence selection behavior where the least used template is picked first.

In order to combine invidividual snippets of sentences and to insert the values of variables at runtime, the sentence bank uses the *Jinja2 Text Templating Library*[7] which has gained popularity in the Python community where it is used for text templating in the *Flask Microframework*[8] and *Ansible*[9]. Its syntax allows to define arbitrarily complex code inside of `{{expressions}}`, which are surrounded by curly braces and efficiently expanded after an initial compilation step.

The planning agent is responsible for creating a `shared_environment` (essentially a dictonary) of convenience accessors to relevant data and functionality, such as the previously introduced `user_recent` function which allows searching for a specific intent in the collection of incoming messages, and a `has_answered(question) -> bool` function. In addition, the Jinja2 rendering environment contains attributes concerning the current user, for instance `current_question: Question`, `current_questionnaire: Questionnaire`, `questionnaire_completion: float`, etc. in order to simplify the design of all templates.

Listing 6.1 on the facing page shows an excerpt (partly translated from German) of the outgoing intent `ok_thank_you` – which finds use in many occasions in the project – with its corresponding phrasing choices and conditions, as well as a `what_i_can_do` template that is used in the `intro` callback of Listing 6.2. Note the placeholder `:white_check_mark:` that will be replaced with a checkmark emoji to depict that an action has been completed successfully.

[7] http://jinja.pocoo.org/
[8] http://flask.pocoo.org/
[9] https://www.ansible.com/

```yaml
ok_thank_you:
  choices:
    - ":white_check_mark: Okay, thank you{{' ' + user.name if user.name is
    not none}}!"
    - text: ":white_check_mark: Perfect, got it."
      condition: {{not user_recent('user.angry')}}
    - text: "okey-dokey, thanks!"
      condition: {{not user_recent('user.sad') and user.informal}}

what_i_can_do:  # Left in German to demonstrate proper formal address
  "Ich kann {{'Ihnen' if user.formal else 'dir'}} helfen,"
  "{{render('dmg_type_snippet')}}"  # e.g. 'den Wasserschaden'
  "an {{'Ihrem' if user.formal else 'deinem'}} "
  "Smartphone zu melden."
```

Listing 6.1: Jinja2 Response Template inside a YAML file

Bringing it all together, a typical callback handler might look as follows (see also Figure 6.3):

```python
def intro(r, c):
    if not c.has_outgoing_intent("what i can do", age_limit=10):
        # Intent is normalized to what_i_can_do
        r.say("What I can do")

    r.say("what you can say")
    return "explained_usage", 1  # lifetime of 1 dialog move
```

Listing 6.2: A handler function explaining how to use the bot

When the `intro` callback is triggered, the context (`c`) is queried for whether the bot itself sent the intent `what_i_can_do` (normalized) within the last 10 messages of the conversation and, if not, instructs the referenced response composer (`r`) to create a new `ChatAction` with the same intent. In any case, the intent `what_you_can_say` is added aswell, with the response composer looking up both intents in the sentence bank and rendering the Jinja2 templates. The function then returns a new state `explained_usage` with a lifetime of one dialog turn, meaning that the next incoming message is going to be looked up in that particular state in the router.

Figure 6.3: Visualization of the *intro* callback

As the `ResponseComposer` is smart enough to combine individual intents (i.e. text snippets) to a larger sentence by adhering to linguistically correct grammatical punctuation and casing, it accepts any number of intents as `*arguments` and, if given, merges them together (an example of this behavior can be seen on Line 3 in Listing 6.4 on the next page .

It became apparent that designing a conversation using this model is quite straightforward and without hurdles, as all components are well-encapsulated and adhere to the separation of concerns which in turn ensures testability. Adding more context sensitive tools is also simple and eventually aids the goal of making the logic of the callback handlers as readable and comprehensible as possible, also for copywriters from a less technical background.

6.2.4 Claim Recording

In order to elicit the necessary information to record a claim from the user, an abstraction was built to define the required data slots as **`Questions`** arranged in **`Questionnaires`**, configured inside of YAML files.

Questions may contain information such as their ID, name, a regular expression for verification, and a flag denoting whether this question is required or optional. On top of that, they include the eventual user-facing response snippets such as title, a hint for clarification, and an example. Just like the RG snippets, these are allowed to include Jinja2 templates for dynamic content generation at runtime.

Listing 6.3 on the following page is an excerpt from the application router that shows the *generic* states and transitions to handle an arbitrary number of questions, meaning that requesting another data slot from users is a simple matter of adding another question to the YAML configuration and optionally adding a function to properly interpret user inputs. In order to check an answer for validity, as a first instance the question's regular expression is consulted. If none is set, there is an additional Python module `answercheckers` that includes specific algorithms to determine and transform the user's input, for instance datetime and name inputs, and a phone model identifier that accesses a database of smartphone models and yields back a number of keyboard button choices when the input is ambiguous, as displayed in Figure 6.4.

Listing 6.4 on the following page and Figure 6.5 on page 79 show an example of how the `skip_question` callback decides the next actions in case the user does not want to give an answer to a question.

Figure 6.4: Reply keyboard buttons in case of ambiguous user input

```
"dialog_states": {
    ⋮
    States.ASKING_QUESTION: [
        # Handles responses to a question
        IntentHandler(send_hint, intents=['can_you_help', 'clarify']),
        IntentHandler(static_smalltalk_response[6.2.2/3], intents='thank_you'),
        IntentHandler(send_example, intents='example'),
        IntentHandler(skip_question[6.4], intents='skip'),
        NegationHandler(skip_question),
        MediaHandler(check_answer),  # images, videos
        IntentHandler(check_answer),  # catches all other messages
    ],
    States.USER_CONFIRMING_ANSWER: [
        # Handles responses to explicit confirmation prompt
        AffirmationHandler(store_answer),
        NegationHandler(repeat_question),
        IntentHandler(check_answer)  # treat other messages as correction
    ],
    ⋮
}
```

Listing 6.3: State machine for controlling claim questionnaires

```
def skip_question(r, c):
    if c.current_question.is_required:
        r.say("sorry", "but", "cannot skip this question")
        r.ask("continue anyway", choices=("affirm_yes", "negate_no"))
        return "ask_continue_despite_no_skipping", 2
    else:
        r.say(
            "skipping this question",
            parameters={'question': c.current_question}
        )

        c.add_answer_to_question(c.current_question, UserAnswers.NO_ANSWER)
        return ask_next_question(r, c)
```

Listing 6.4: Exemplary action handler for skipping a question

(a) Skipping an optional question **(b)** Abort prompt when a question is mandatory

Figure 6.5: Paths of the function to skip a question

CHAPTER 7

Evaluation and Discussion

After the successful implementation of the prototype, it was necessary to evaluate whether the solution actually fulfilled the main goal of creating a conversational user interface with a positive user experience.

In order to verify that, the previously gathered requirements were each reviewed regarding two aspects:

1. Whether the requirement was fully implemented in the prototype.
2. Whether the requirement was actually expedient and sufficient regarding the objectives.

As it turned out, most of the gathered requirements were satisfied in the solution and made sense in ways of achieving a better user experience or assuring software quality. Nonetheless, some issues require further work. Thus, only the requirements that were *failed* to fulfill are analyzed hereupon.

In terms of methodology, the logs of user interactions created by the `ConversationRecorder` allowed to make objective judgements about the state of the system, such as the duration of the dialog, determined intents and parameters, system state transitions, etc.
Although these metrics were useful to iteratively refine the system over the course of many interactions, they do not provide much value in terms of evaluating user experience, where subjective metrics should be used to elicit the opinions of users about some aspect of quality. The use of appropriate degrees of formality is an example of one of these aspects, as it is inherently subjective. As quality involves users making comparisons of the perceived qualities of a system against its desired qualities, quality can only be measured by taking into account the opinions of users [MCG16]. Therefore, a user survey was conducted

with consideration of the quality attributes of Section 3.2.2 on page 43 and its results are presented here.

7.1 Satisfaction and Adequacy of Requirements

7.1.1 Survey Results (non-functional Requirements)

The evaluation was conducted as an online survey with acquaintances and Fraunhofer employees, whose task was to imagine they had a broken smartphone and now needed to report the damage to their insurance. No further instructions or help were provided so that every participant had to figure out for him or herself how to best interact with the prototype in order to achieve this goal. Unfortunately, Facebook's review process for new bots was suspended at that time, which meant that even though the Messenger bot was fully functional, all participants had to use Telegram to interact with the bot (not making a big difference as it behaves identically on both platforms).

As depicted in Figure A.1 on page 91, fourteen participants partook in the study out of whom 35.7% reported to use chatbots on a regular basis, 57.1% used them occasionally, and it was the first encounter with a chatbot for one participant (7.1%). Surprisingly enough, all fourteen participants managed to fulfill the task and submitted their contrived claim without problems in an average time span of approximately 4 minutes, meaning that the task completion was 100%. Intuitively, the task completion metric quantifies the capability of the chatbot to collect sufficient information for the dialog to succeed and to receive all necessary information [MCG16], hence it can be concluded that the system-directed interaction strategy paid off in this regard (while it has problems in other areas as described below).

Contingent upon the remote character of the survey, the questions regarding user experience must be considered as remembered experience (Section 3.2 on page 42), which is not ideal, as metrics for expected and momentary experience would have yielded additional clues to the mental state of the participants before and during the interaction. However, the `ConversationRecorder` picked up on the bi-directional utterances and the internal state transitions of the system, which allowed to deduce some insights into the thought processes of users.

Quality Attributes: Functionality TEST

- **Use appropriate degrees of formality** (Figure A.3 on page 92)

$\longrightarrow$ With 8.3 points on average, the impressions of the study participants were divided into a larger portion that were completely satisfied with the formal and informal speech, and two persons who rated this survey question with 1 and 2 out of 10 points. One of them reported that he was anxious about the bot repeatedly calling him by his first name once he had been asked for it. What other problems arose for

these two participants to rate the degree of formality this weakly, remains unclear. Future iterations should take the first-name-address concern into consideration and possibly implement a broader approach to formality detection than simple regular expressions.

- **General ease of use** (Figure A.5 on page 93)

$\longrightarrow$ Certainly, a mean of 8 points indicates a rather good usability, for general ease of use being one of the main goals of the bot however, the variance of 2.46 points in the participants' answers is a little too high. The experience across different user clientele should be more consistent, although none of the participants rated the experience with less than 5 points, proving that the interaction is definitely above average.

Quality Attributes: Humanity TEST

- **Convincing, satisfying, & natural interaction** (Figure A.7 on page 93)

$\longrightarrow$ At a mean of 7.9 points, the study participants' interactions had been fairly convincing, but not entirely. One point to note is that the system-directed interaction strategy that was employed to gather answers to questions is always more restrictive than true user-directed input which gives more freedom. This might have played a crucial role in the rating of this quality attribute. In the future, it could be argued that more of the questions should be collected via user-directed or a mixed input rather than system-directed prompts in order to improve the rating here.

- **Able to respond to specific questions** (Figure A.8 on page 94)

Note: The metric to this quality attribute was altered from the objective "% of successes" of Radziwill and Benton to a subjective "0..10" user rating due to the inability of the bot to answer many domain-specific questions. Instead, it had a larger bank of sentences for answering questions in the context of smalltalk, which is measured better by subjective than objective metrics.
$\longrightarrow$ At an average of 7.6 points, the ability to answer questions was probably rated better than desserved since the system-directed interaction strategy guides the user from the point of starting the claim onwards, not allowing the user to ask many questions in the process. Explicit question-answering is something that should definitely be included in a production-level version of the prototype, including insurance domain knowledgebases that had not been available for this thesis.

Personality TEST

- **Convey a personality** (Figure A.10 on page 94)

⟶ With a mean rating of 5.2 points to the survey question "It felt like the bot had a personality", the challenge of conveying a specific personality was not accomplished. The personality traits of Requirement 3.4 were apparently not enough to convince the study participants of the bot having any prominent quirks that they would identify as personality traits. It must be said however, that little effort went into the task at hand and a professional conversational copywriter should be able to invent a much more convincing character whose personality is built up like that of a protagonist in a novel.
One idea that comes to mind about why the personality was perceived so badly, is that the bot does not introduce itself with a name. Instead of trying to keep the conversation as clinical and professional as possible, the bot should include at least a first name in its personality.

Quality Attributes: Affect TEST

- **Make tasks more fun and interesting** (Figure A.11 on page 95)

⟶ In 7.2 points on average, the task of recording a claim was perceived as fairly fun and interesting, although there is still room for improvement. Again this goes back to the question of system-directed vs. user-directed input, where a more open but nonetheless guided interaction strategy would be the ideal case from the user perspective. The user should be able to say anything with the bot responding, instead of being forced into a workflow that the bot prescribes, which is partly the case for the prototype. Smalltalk is still possible when a question has been asked, but the bot is incapable of deviating from the predefined questionnaire path, which should be improved in the future.

- **Entertain and/or enable participant to enjoy the interaction** (Figure A.12 on page 95)

⟶ Just like the challenge of making tasks fun and interesting, creating an enjoyable conversation is the daily bread of a conversational copywriter. With 7.7 points on average, it became apparent that the prototype does some things right, but still lacks the ability to really entertain. It is capable of telling insurance jokes and uttering a couple of rather funny sentences, but the overall entertainment factor should be improved by more comedic authoring and a more holistic approach than just jokes.

7.1.2 Functional Requirements

Quality Attributes: Performance TEST

- Provide appropriate escalation channels to humans

⟶ A Human Handover component was not implemented due to time constraints.

Usability Heuristics BEST PRACTICES

3. **User control and freedom & Error Prevention** — "Get confirmation from the user at critical points, and provide escape hatches for multi step interactions."
4. **Flexibility and efficiency of use**

⟶ While efficiency is certainly ensured, the system lacks user freedom and the ability to escape from the multi step interaction when recording the claim. As soon as the system asks a question, it expects an answer and does not yet allow to return to a previous question or correct answers to a question that is not currently under prompt. While skipping questions is possible, the sequential order of the questionnaires is predetermined and hinders flexibility of user-directed input.

7.2 Future Work

Since a future version of the prototypical solution might at some point be used in a productive environment, some words should be said about improvements to the system that would yield considerable benefits.

First of all, it should be fairly easy to add additional features due to the modular architecture, whereas the domain logic (i.e. conversation design) had grown over time and should probably revisited or reworked with proper content. For instance, an insurance vendor may want to add a knowledgebase for common FAQs that the bot would be able to answer, together with more informational content from the insurance domain for more fitting smalltalk.

The most important issue to address in terms of user experience however, as mentioned a couple of times in Chapter 7 on page 81, is to hand over more control in the dialog to users instead of enforcing a specific claims workflow.
Contrary to the sequential "interrogation" procedure as seen in Figure 6.4 on page 77, a more user-directed approach would allow eliciting several data items in a single message and conveying a *higher level of system intelligence* by stating more open-ended prompts. On top of that, it would allow eliciting information from users whenever it is suitable for them.

```
System: What can you tell me about the damage?
User:   Yesterday, my friend pushed me over, causing me to
        drop my iphone 6 to the floor and now the display
        is shattered. Can you help me?
```

This paradigm presents two challenges:

1. **Detection of the individual data slots**. All of the entity-parameter pairs the system should be able to differentiate would need to be added to the NLU engine, requiring big sets of conversational data for training of the underlying model. In addition, since Dialogflow is not capable of handling multiple intents in a single utterance (and neither are most of its competitors), the backend would be responsible for tokenizing the message into chunks that each contain a intent, for instance `phone_broken` and `agent.can_you_help` in the previous example, which would then be sent to the NLU service individually and yield multiple intents. A naïve approach to tokenization would be to split the message at the sentence markers (`.`), although this approach would already have failed with one participant in the survey who had a habit of using "," instead of "." for punctuation, which goes to say that incoherent patterns like this are not uncommon in 21th century messaging.

2. **Presenting implicit and explicit confirmation and applying the `answerchecker` functions to every detected data slot individually**. In above dialog, the utterance "yesterday" is not sufficient and needs refinement by providing an exact daytime, and the phone model parameter "iphone 6" is ambiguous, thus requiring the system to present a set of reply buttons to select choices from as in Figure 6.4 on page 77. The challenge is to come up with a good strategy and implementation for handling these confirmations that will not overwhelm the user, e.g. by requesting them in consecutive dialog turns:

   ```
   System: Oh no, I'm sorry to hear that your phone has a display damage...
           I understood that the incident happened yesterday,
           at what time exactly?
   User:   13:00
   System: Thanks, and which model is it again?
           [Presents choices as reply buttons]
   User:   [Apple iPhone 6s]
   System: Alright, so you mentioned that another person was involved in
           the incident. Could you tell me more about them?
   ...
   ```

   The `questionnaire` logic in the solution is already organized in a way that questions are not asked when an answer has already been collected previously. This means that the user-directed interaction capabilities could be integrated on top of a predetermined path in order to guide users when they fail to provide relevant information by themselves.

A dialog structure resembling this paradigm would also be ideal for recovery mechanisms, as it would create a generic structure for eliciting (and thus also *correcting*) arbitrary information at any moment in the conversation. However, it cannot be overstated that this approach really is the holy grail of conversational interfaces and extremely challenging to implement under consideration of border cases and the intricacies of natural language. As a result, the paradigm was not implemented due to the restricted amount of time and resources available for this thesis, although it became apparent that the current system-directed design works well enough to yield above average satisfaction scores (Figure A.2 on page 92) and a perfect task completion ratio.

More Ideas for Future Improvement

- By exploiting the layered states with lifetimes and continuous persistence of dialog utterances, it would be possible to implement the **memento pattern** for modifying and removing answers to previous questions. In any state, the user should be able to refer back to previous answers by terms such as “I made a mistake, this was not correct” or “I forgot to add something” and the system should be flexible enough to handle these corrections by remembering the recent interaction.
- Formal and casual address of the user are implemented in German language to demonstrate slight altering of sentences in order to respect the user’s preferred interaction style. If future iterations on the system require to incorporate more sophisticated sentiment analysis techniques to **account for the emotional state of peers**, the existing implementation may serve as a basis for selection and alteration of utterances based on conditions and Jinja2 templating.
- As addressed in Chapter 7 on page 81, a component for **handover to human agents** was not yet added. For handling cases where the bot fails to understand or properly respond to users, a handover component could be implemented to interpose a human in the conversation until the case is resolved. The designated behavior of this component depends on a couple of factors that should be considered, such as whether to trigger handover on a sentiment threshold (user is angry, gloomy about something) vs. per explicit elicitation of a human agent (“I want to speak to a human”), as well as the actual means of interaction i.e. whether the bot’s interactions should be suspended while the human agent is active or the agent should assume control over the bot’s capabilities in a dedicated administration interface so that visual features such as reply buttons and media still work.
- The bot is currently designed to deal with consumer requests (i.e. claims) as quickly and efficiently as possible, which makes sense when the goal is just to provide a convenient claims process. Companies may however want to incorporate branding and conversational commerce in order to provide relevant content, strategical company values and product advertisements, so it may be profitable to keep the user in the loop for longer to provide these commercial services and as a tool for customer

engagement. A combination of push messages and pattern matching or machine learning algorithms may be useful to target customers in favorable situations.

Legal Considerations

For chatbots in the insurance domain, it will be mandatory to consult with legal entities in order to ascertain the general liabilities inside of chat interfaces. Questions to ask may be whether statements customers make in messengers may be considered as legally binding and they may be held accountable for information provided to a chatbot, as well as the extent to which prevailing data privacy laws must be obided in production.
As these points were not examined in this work, they require further research before market maturity.

CHAPTER 8

Conclusion

In brief, the research questions posed in this thesis were **1)** in what ways would the German insurance sector benefit from chatbot technology in the present and future, **2)** what are constituents of conversational interfaces that ensure a positive user experience, and **3)** the practical application of these principles in a prototypical implementation.

Hence, the work started by introducing the problems and opportunities of the German insurance sector in an advent of changing consumer expectations towards more prominent user-centric and on-demand services. It was explained how insurers traditionally face rather poor customer engagement due to very few touch points with their clients and how these reasons, bundled with faltering economic conditions, lead to a strong pressure to change.

In order to address changing customer expectations and combat poor engagement, the prospect of employing chatbots as a first line of support was put forward and backed by an evidentiary abundance of data in insurance, advancements in the disciplines of data analysis and artificial intelligence, as well as the current landscape of messaging services.

Moreover, the benefits and possible applications of chatbots were suggested, in that they might aid insurers in faster, cheaper, and more convenient claims processing; while touching on underwriting, conversational commerce, and fraud detection as promising additional branches for AI.

Contingent upon the few and brief touch points with customers, chatbots in insurance need to make the most out of every interaction and deliver a pleasing experience at the first impression. For this reason, the chatbot prototype developed for the Fraunhofer IAO was designed after researching indicators and expedient practices for a positive user experience, which subsequently served as requirements, integrated in the prototypical solution.

These requirements originated from review of scientific and industrial literature in the field of HCI, namely affective computing, usability engineering, conversation design, as well as resources about software engineering and software quality assurance, in order to gather desirable characteristics, use cases, best practices, evaluation strategies, metrics for classification of CUIs, and QA quality attributes.

On the basis of these specifications, the chatbot was engineered and its core architecture and individual components presented with consideration of the academic backgrounds. It was thereafter evaluated by means of user opinions and fulfillment of requirements in order to suggest future improvements that indicated added value in terms of user experience and conversation design.

For the greater part, the developed solution was received positively with regards to the initial goals and may thus be considered a success. Although some issues emerged concerning personality, entertainment, and interaction strategy, the evaluation unveiled no cases of failure that a future maintainer would not be able to fix easily or with some monetary investment in subsequent iterations of engineering.

In addition to a public display of the solution in a demo booth at the Fraunhofer grounds, further evaluations will be performed with insurance representatives in a few weeks after the time of writing. This is going to yield more valuable insights into applicability of the prototype in a productive environment, and eventually decide its fate for the future.

APPENDIX A

Charts and Diagrams

A.1 Survey Results

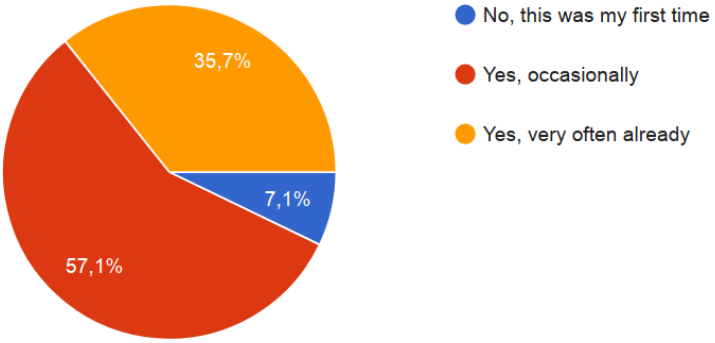


Figure A.1: “Have you interacted with a chatbot before?”

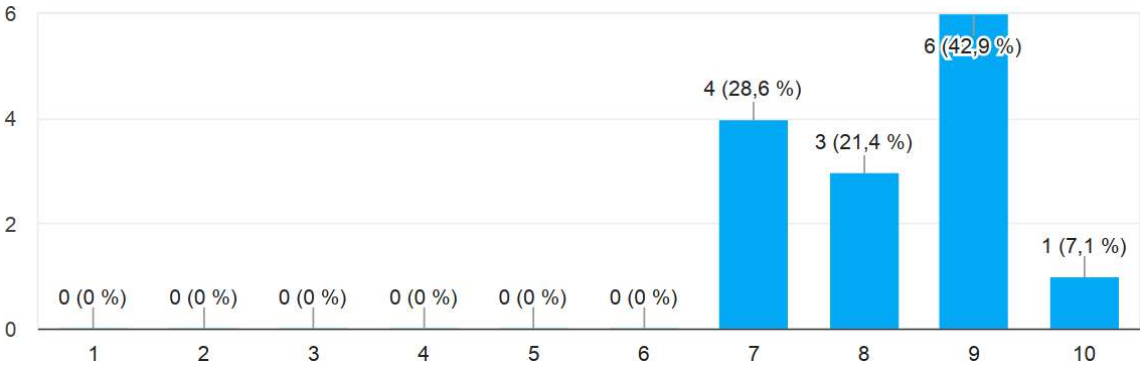


Figure A.2: Overall Experience (0..10)

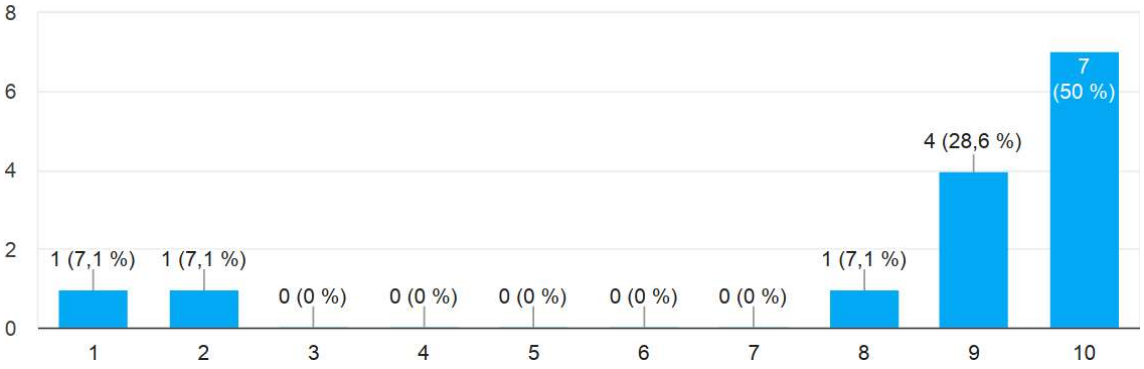


Figure A.3: Use appropriate degrees of formality (0..10)

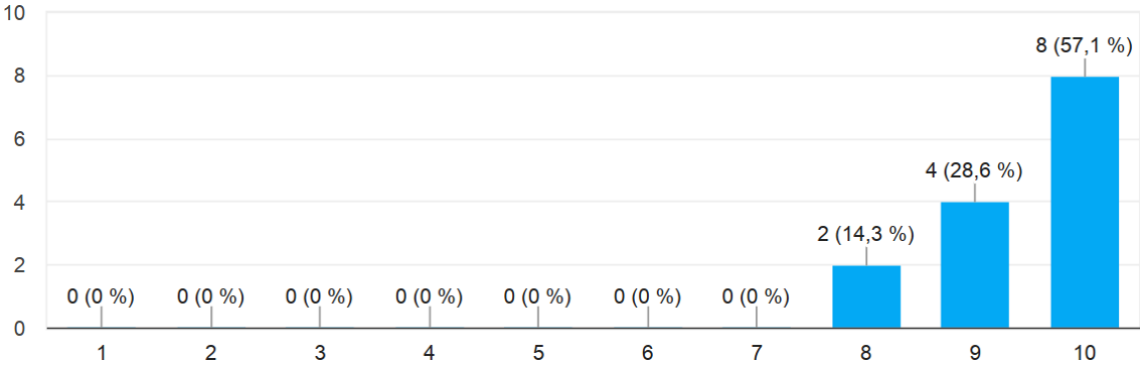


Figure A.4: Linguistic accuracy of outputs (0..10)

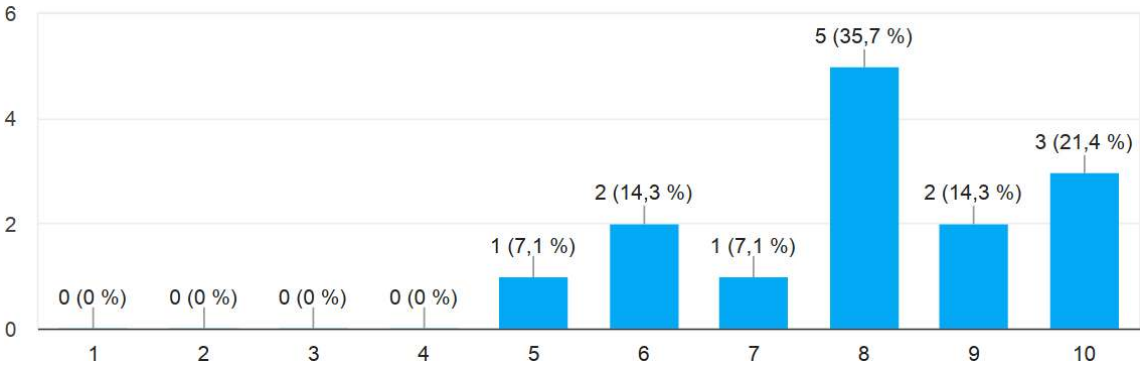


Figure A.5: General ease of use (0..10)

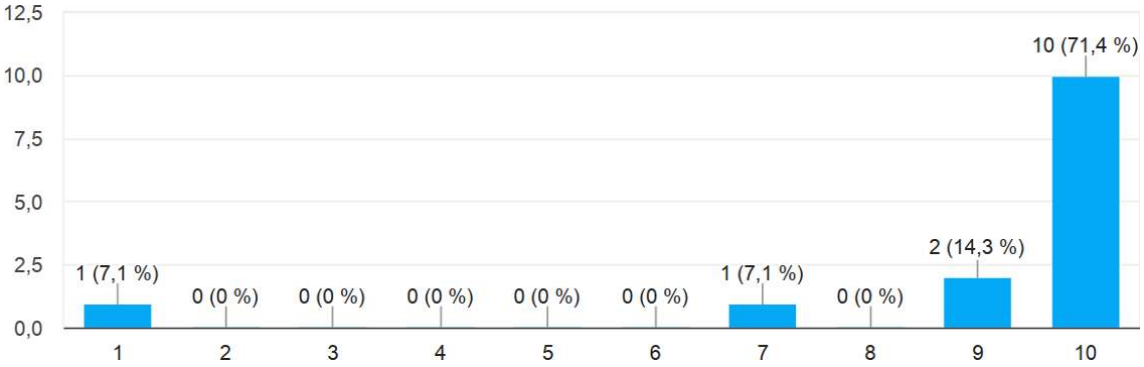


Figure A.6: Transparency to inspection – "It was easy to tell that I was talking to a bot" (0..10)

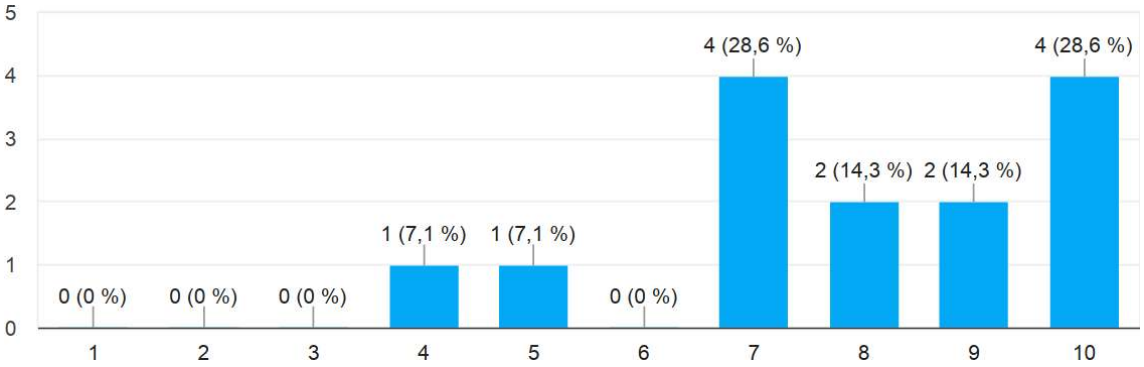


Figure A.7: Convincing, satisfying, & natural interaction (0..10)

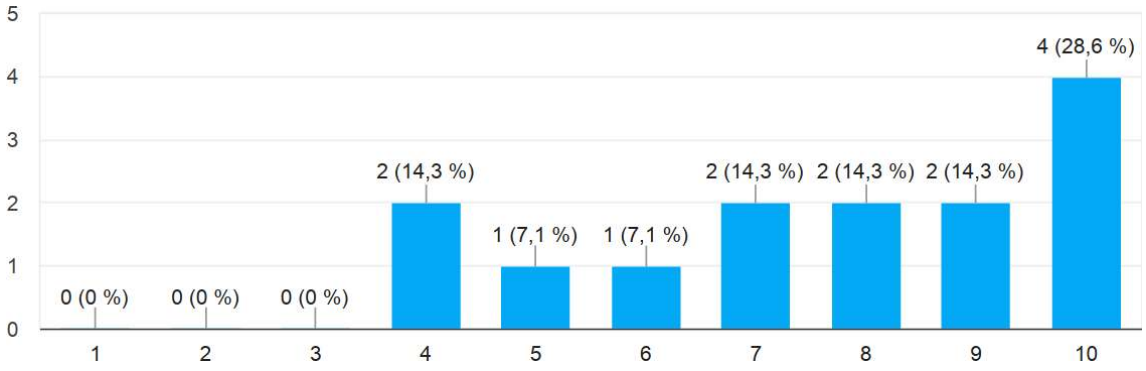


Figure A.8: Able to respond to specific questions (0..10)

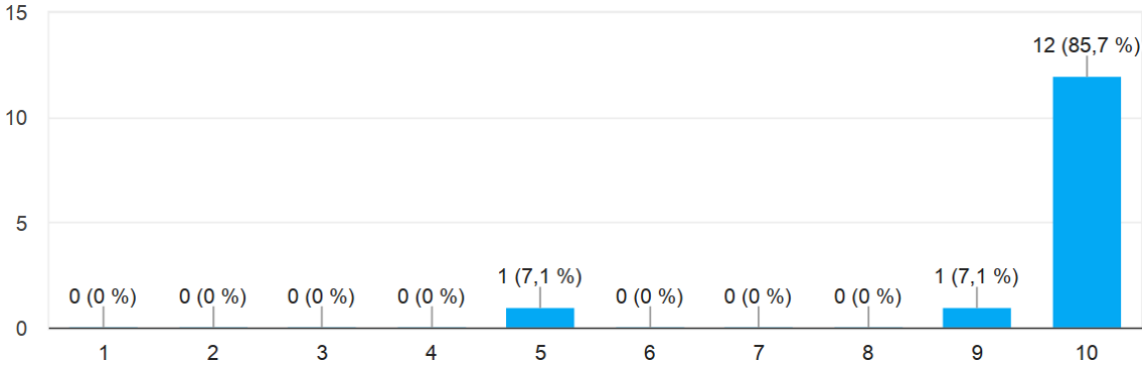


Figure A.9: Provide greetings (0..10)

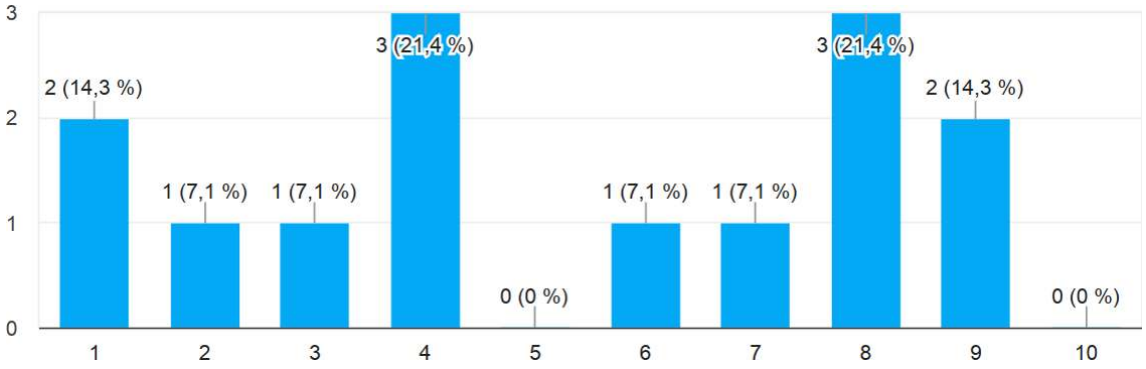


Figure A.10: “It felt like the bot had a personality” (0..10)

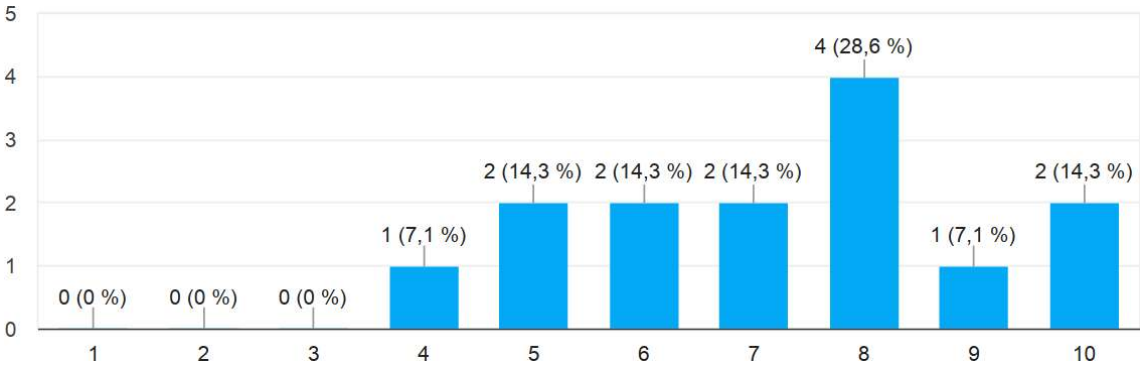


Figure A.11: Tasks (recording claim) were fun and interesting (0..10)

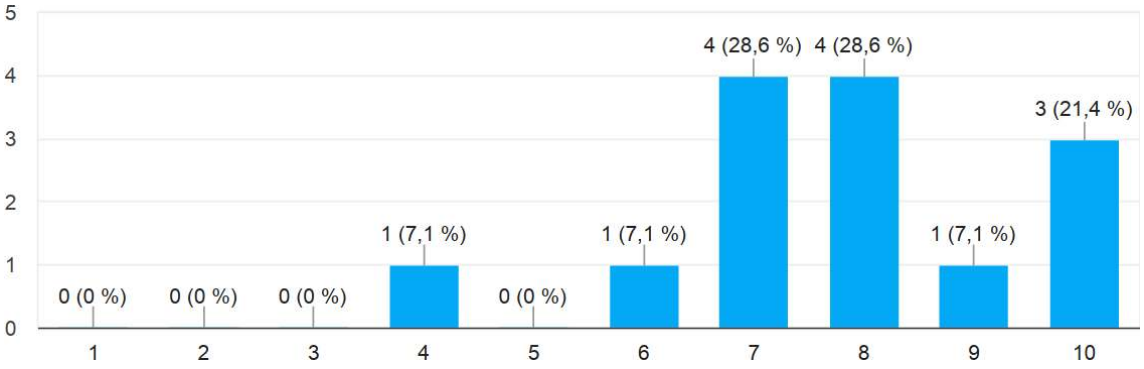


Figure A.12: The interaction was entertaining (0..10)

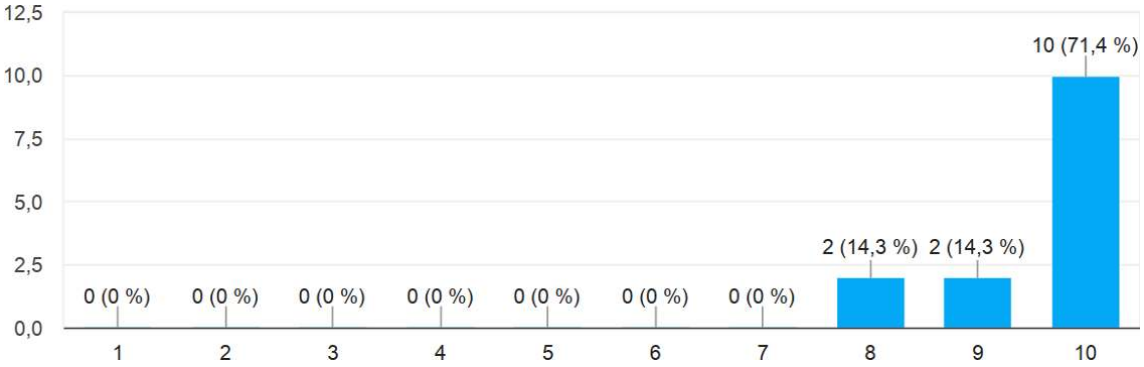


Figure A.13: “It felt like the bot did not try to deceive me” (0..10)

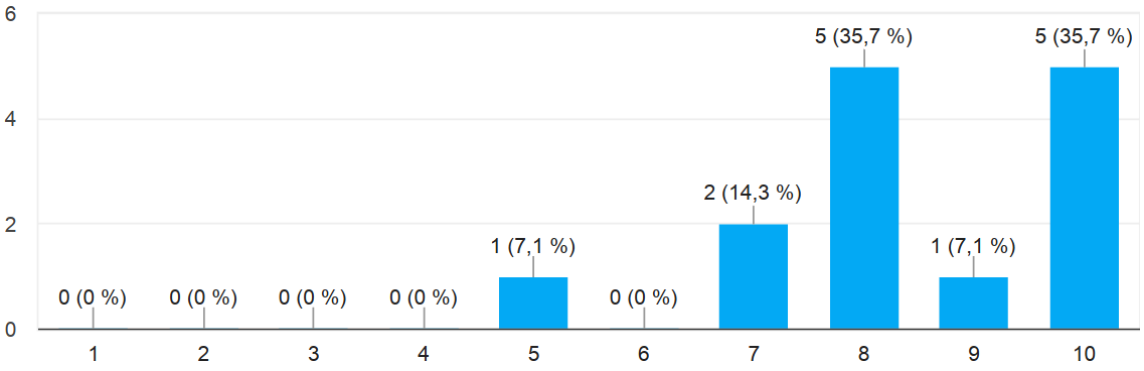


Figure A.14: Trustworthiness (0..10)

A.2 Model Diagrams

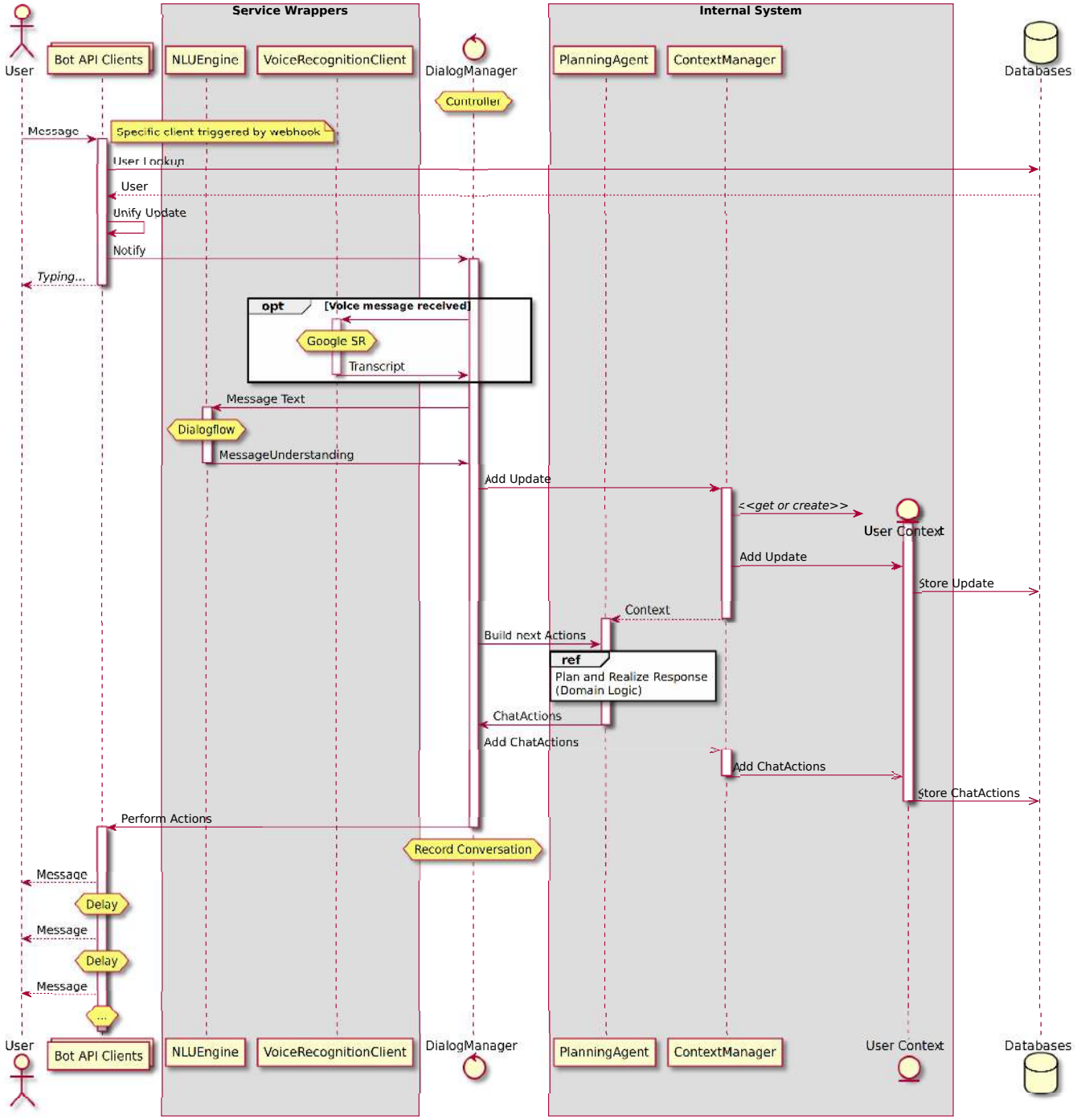


Figure A.15: Sequence diagram of core architecture

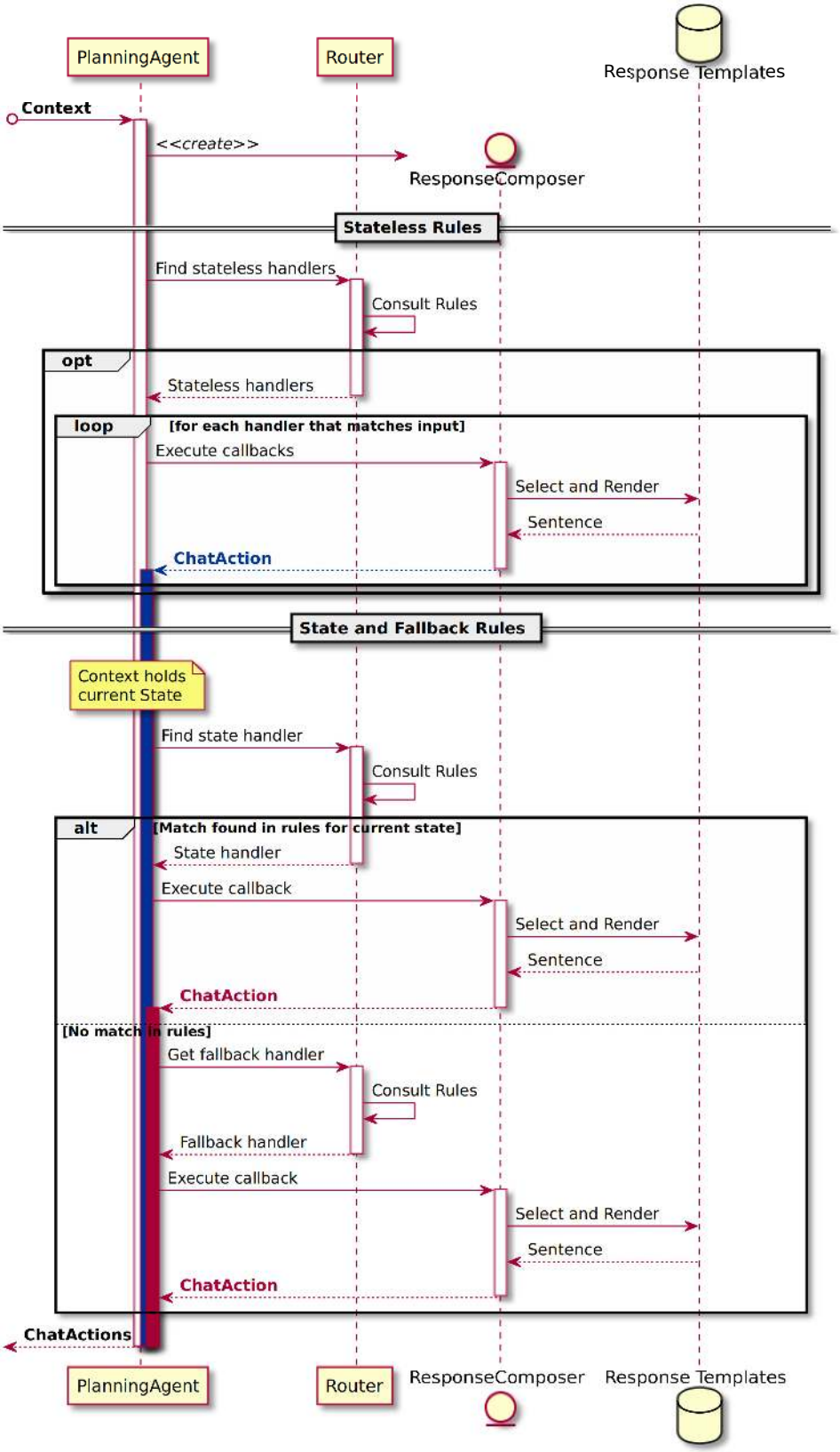


Figure A.16: Sequence diagram of the planning agent's domain logic

Bibliography

[16] *Digital Transformation of Industries*. Tech. rep. World Economic Forum and Accenture, 2016 (cit. on pp. 17, 18).

[17a] *Technology Vision for Insurance report*. Tech. rep. Accenture, 2017 (cit. on p. 18).

[17b] *Touchless Claims - 2017 Future of Claims Study*. Tech. rep. LexisNexis, 2017 (cit. on pp. 18, 21, 22).

[17c] *UX Analytics for Chatbots: metrics, method and case study*. Tech. rep. uxprobe, 2017 (cit. on p. 24).

[18] *Wie die Mafia auch Versicherungen betrügt*. 2018. URL: https://www.gdv.de/de/themen/news/wie-die-mafia-auch-versicherungen-betruegt-29910 (cit. on p. 23).

[AKSS03] A. Abran, A. Khelifi, W. Suryn, A. Seffah. "Usability meanings and interpretations in ISO standards." In: *Software quality journal* 11.4 (2003), pp. 325–338 (cit. on p. 42).

[al12] K. et al. *Projekt openXchange - Servicenetzwerk zur effizienten Abwicklung und Optimierung von Regulierungsprozessen bei Sachschäden*. 2012 (cit. on p. 22).

[BCS+17] M. G. de Bayser, P. Cavalin, R. Souza, A. Braz, H. Candello, C. Pinhanez, J.-P. Briot. "A Hybrid Architecture for Multi-Party Conversational Systems." In: *arXiv preprint arXiv:1705.01214* (2017) (cit. on p. 36).

[Bii13] P. Bii. "Chatbot technology: A possible means of unlocking student potential to learn how to learn." In: *Educational Research* 4.2 (2013), pp. 218–221 (cit. on p. 33).

[Cah17] J. Cahn. "CHATBOT: Architecture, Design, & Development." PhD thesis. University of Pennsylvania, 2017 (cit. on pp. 31, 32, 56).

[Cam17] C. Campbell. *Why Chatbots Are Taking over the Insurance Industry*. 2017. URL: https://masterofcode.com/blog/insurance-industry (cit. on p. 23).

[CMR+08] R. S. Cooper, J. F. McElroy, W. Rolandi, D. Sanders, R. M. Ulmer, E. Peebles. *Personal virtual assistant*. US Patent 7,415,100. Aug. 2008 (cit. on p. 34).

[COHM05] S.-W. Chu, I. O'Neill, P. Hanna, M. McTear. "An approach to multi-strategy dialogue management." In: *Ninth European Conference on Speech Communication and Technology*. 2005, pp. 865–868 (cit. on p. 70).

[Dal16] R. Dale. "Industry Watch: The Return of the Chatbots." In: *Natural Language Engineering* 22.5 (2016), pp. 811–817 (cit. on pp. 19, 20, 27, 32, 34).

[Dav17] O. Davydova. *25 Chatbot Platforms: A Comparative Table*. 2017. URL: https://chatbotsjournal.com/25-chatbot-platforms-a-comparative-table-aeefc932eaff (visited on 03/29/2018) (cit. on p. 68).

[Der17] R. Derler. *Chatbot vs. App vs. Website – Chatbots Magazine*. Dec. 2017. URL: https://chatbotsmagazine.com/chatbot-vs-app-vs-website-en-e0027e46c983 (visited on 03/29/2018) (cit. on p. 24).

[Eeu17] M. Eeuwen. "Mobile conversational commerce: messenger chatbots as the next interface between businesses and consumers." MA thesis. University of Twente, 2017 (cit. on pp. 19, 23, 24, 49).

[Gar15] Gartner. *Market Trends: Mobile App Adoption Matures as Usage Mellows*. 2015. URL: https://www.gartner.com/newsroom/id/3018618 (visited on 03/29/2018) (cit. on p. 25).

[Gib13] J. J. Gibson. *The ecological approach to visual perception*. Psychology Press, 2013 (cit. on p. 41).

[GP16] I. Guzmán, A. Pathania. *Chatbots in Customer Service*. Tech. rep. Accenture, 2016 (cit. on pp. 19–21, 23, 31, 32, 36).

[Has10] M. Hassenzahl. "Experience design: Technology for all the right reasons." In: *Synthesis Lectures on Human-Centered Informatics* 3.1 (2010), p. 95 (cit. on p. 42).

[HFSA16] H. Harkous, K. Fawaz, K. G. Shin, K. Aberer. "PriBots: Conversational Privacy with Chatbots." In: *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*. Denver, CO: USENIX Association, 2016. URL: https://www.usenix.org/conference/soups2016/workshop-program/wfpn/presentation/harkous (cit. on pp. 19, 55).

[Hol16] C. Holotescu. "MOOCBuddy: a Chatbot for personalized learning with MOOCs." In: *RoCHI*. 2016, pp. 91–94 (cit. on p. 33).

[Kap13] R. Kaplan. *Beyond the GUI: It's time for a Conversational User Interface*. 2013. URL: https://www.wired.com/2013/03/conversational-user-interface/ (visited on 03/29/2018) (cit. on pp. 39, 41).

[KNF17] P. Kashfi, A. Nilsson, R. Feldt. "Integrating User eXperience practices into software development processes: implications of the UX characteristics." In: *PeerJ Computer Science* 3 (2017), e130 (cit. on p. 42).

[KOY09] J. Kirakowski, P. O'Donnell, A. Yiu. "Establishing the Hallmarks of a Convincing Chatbot-Human Dialogue." In: *Human-Computer Interaction*. Ed. by I. Maurtua. Rijeka: InTech, 2009. Chap. 09. DOI: 10.5772/7741. URL: http://dx.doi.org/10.5772/7741 (cit. on pp. 49, 56, 57).

[KV17] V. K., V. Vlad. *How Much Does it Cost to Build a Chatbot?* 2017. URL: https://rubygarage.org/blog/how-much-does-it-cost-to-build-a-chatbot (visited on 03/29/2018) (cit. on p. 27).

[Les16] S. Lessin. *Bots, Conversational Apps and Fin*. 2016. URL: https://www.theinformation.com/articles/on-bots-conversational-apps-and-fin (visited on 03/29/2018) (cit. on p. 24).

[LM14] P. Lison, R. Meena. "Spoken dialogue systems: the new frontier in human-computer interaction." In: *XRDS: Crossroads, The ACM Magazine for Students* 21.1 (2014), pp. 46–51 (cit. on p. 40).

[LS16] E. Luger, A. Sellen. "Like having a really bad PA: The Gulf between User Expectation and Experience of Conversational Agents." In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM. 2016, pp. 5286–5297 (cit. on p. 53).

[Mar16] L. Margalit. *The Psychology of Chatbots*. 2016. URL: https://www.psychologytoday.com/us/blog/behind-online-behavior/201607/the-psychology-chatbots (cit. on p. 50).

[MCG16] M. McTear, Z. Callejas, D. Griol. *The Conversational Interface*. Vol. 6. 94. Springer, 2016, p. 102 (cit. on pp. 17, 19, 20, 32, 33, 35, 40, 41, 45–53, 56, 59, 66, 68, 69, 73, 81, 82).

[MJDV16] A. Murgia, D. Janssens, S. Demeyer, B. Vasilescu. "Among The Machines: Human-Bot Interaction On Social Q&A Websites." In: (2016) (cit. on pp. 32, 35).

[Nao09] E. Naone. "TR10: Intelligent Software Assistant." In: *Mar.-Apr* (2009) (cit. on p. 34).

[NBGC14] M. Niddam, N. Barsley, J.-C. Gard, U. Cotroneo. *Evolution and Revolution: How Insurers Stay Relevant in a Digital Future*. 2014. URL: https://www.bcg.com/publications/2014/insurance-technology-strategy-evolution-revolution-how-insurers-stay-relevant-digital-world.aspx (visited on 03/29/2018) (cit. on p. 18).

[NDH+17] E. Nordman, K. DeFrain, S. (Hall, D. Karapiperis, A. Obersteadt. *How Artificial Intelligence Is Changing the Insurance Industry*. 2017 (cit. on pp. 17–20, 23).

[New17] M. Newlands. *10 Ways AI and Chatbots Reduce Business Risks*. 2017. URL: https://www.entrepreneur.com/article/305073 (cit. on p. 21).

[NM90] J. Nielsen, R. Molich. "Heuristic evaluation of user interfaces." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 1990, pp. 249–256 (cit. on p. 44).

[Nor04] D. Norman. "Emotional design: Why we love (or hate) everyday things." In: *The Journal of American Culture* 27.2 (2004), pp. 234–234 (cit. on p. 51).

[PG17] M. Portela, C. Granell-Canut. *A new friend in our Smartphone? Observing Interactions with Chatbots in the search of emotional engagement*. 2017 (cit. on pp. 19, 53).

[Phi18] C. Phillips. *How to Balance AI Capabilities with UX Design when Building Chatbots*. 2018. URL: https://chatbotsmagazine.com/how-to-balance-ai-capabilities-with-ux-design-when-building-chatbots-6c0e2567cdec (visited on 03/28/2018) (cit. on p. 49).

[RB17] N. Radziwill, M. Benton. "Evaluating Quality of Chatbots and Intelligent Conversational Agents." In: *arXiv preprint arXiv:1704.04579* (2017) (cit. on pp. 34, 43, 83).

[SA03] B. A. Shawar, E. Atwell. "Using dialogue corpora to train a chatbot." In: *Proceedings of the Corpus Linguistics 2003 conference*. 2003, pp. 681–690 (cit. on p. 41).

[Sac16] C. Sackmann. *Deutsche betrügen massenhaft ihre Versicherungen - doch die sind selber schuld*. 2016. URL: https://www.focus.de/finanzen/versicherungen/jeder-zehnte-fall-ist-gar-keiner-deutsche-betruegen-ihre-versicherungen-wo-sie-nur-koennen-und_id_5246090.html (cit. on p. 23).

[Sät17] A. Sätterkvist. "Visualizing conversational data in virtual reality." In: (2017) (cit. on p. 39).

[Sco16] K. Scott. *Usability Heuristics for Bots*. 2016. URL: https://chatbotsmagazine.com/usability-heuristics-for-bots-7075132d2c92 (visited on 03/30/2018) (cit. on pp. 42, 44).

[Sko10] M. Skowron. "Affect listeners: Acquisition of affective states by means of conversational systems." In: *Development of Multimodal Interfaces: Active Listening and Synchrony*. Springer, 2010, pp. 169–181 (cit. on p. 50).

[Sör17] I. Sörensen. "Expectations on Chatbots among Novice Users during the Onboarding Process." In: (2017) (cit. on pp. 39, 40, 45, 52–54, 57).

[Spi05] U. Spierling. *Interactive digital storytelling: towards a hybrid conceptual approach*. 2005 (cit. on pp. 33, 69).

[Sum] J. Summerfield. *Mobile Website vs. Mobile App (Application) – Which is Best for Your Organization?* URL: https://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/ (visited on 03/29/2018) (cit. on p. 25).

[TL03] D. R. Traum, S. Larsson. "The information state approach to dialogue management." In: *Current and new directions in discourse and dialogue*. Springer, 2003, pp. 325–353 (cit. on p. 70).

[Wei66] J. Weizenbaum. "ELIZA&Mdash;a Computer Program for the Study of Natural Language Communication Between Man and Machine." In: *Commun. ACM* 9.1 (Jan. 1966), pp. 36–45. ISSN: 0001-0782. DOI: 10.1145/365153.365168. URL: http://doi.acm.org/10.1145/365153.365168 (cit. on p. 31).

[Wil10] Y. Wilks. "Is a Companion a distinctive kind of relationship with a machine?" In: *Proceedings of the 2010 Workshop on Companionable Dialogue Systems*. Association for Computational Linguistics. 2010, pp. 13–18 (cit. on p. 33).

[WWLL08] Y. Wu, G. Wang, W. Li, Z. Li. “Automatic chatbot knowledge acquisition from online forum via rough set and ensemble learning.” In: *Network and Parallel Computing, 2008. NPC 2008. IFIP International Conference on*. IEEE. 2008, pp. 242–246 (cit. on p. 66).

[ZR15] G. Zimmermann, S.-L. Richter. “Gründe für die Veränderungsaversion deutscher Versicherungsunternehmen.” In: *Change Management in Versicherungsunternehmen*. Springer, 2015, pp. 11–35 (cit. on pp. 17, 18).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature